**Global Payments SG.**

Logo

Description automatically generated

Global Payments Inc. is an American company that provides payment technology and services to merchants, issuers and consumers. The company processes payments made through credit cards, debit cards and digital and contactless payments. In June 2021, the company was named to the Fortune 500.

Salesforce B2C LINK Cartridge

System requirements

* **LINK Cartridge version**: 22.3.0
* **API Fleet and version: Globalpay** version 2021-03-22.
* **Commerce Platform Version:** This package was implemented against SF B2C version 22.5 and tested against compatibility mode 21.7. This package was developed against SFRA versions 6.0.0.
* The cartridge only supports SFRA.

Before you begin

Before you can install the Salesforce B2C LINK Cartridge, you'll need to contact Globalpay, salesforce@globalpay.com to request credentials, configure the available payment options within Drop-in, and receive your LINK key. Additionally: Use of the Globalpay cartridge requires credentials and keys from Globalpay (contact Globalpay at salesforce@globalpay.com). The cartridge is designed for the US locale and is compatible with any locale if you add the localization strings.

Architecture and Sequence diagram

Table

Description automatically generated with low confidence

Sequence Diagram.

A picture containing table

Description automatically generated

Diagram

Description automatically generated Diagram, table

Description automatically generated

Diagram, table

Description automatically generated

**Install the cartridge and import the metadata.**

Step 1: Install the cartridge

1. Create a folder “Globalpay” in your workspace.

Download and place the latest SG base(app\_storefront\_core, app\_storefront\_controllers )

and GlobalPay (int\_ globalpay, int\_ globalpay \_controllers) cartridge from Marketplace.

## Step 2: Install Node modules

From your project's root directory, install Node modules using your command line:

npm install

## Step 3: Import the metadata

To add new configuration items, import the predefined metadata:

1. Download the [installation package](https://github.com/globalpayments/globalpayments-salesforce-commerce-cloud-b2c/tree/main/Link_globalPay) on our GitHub page.
2. Open the **package/metadata/site\_import/sites/SiteGenesis** folder.
3. Rename the **yourSite\_Id**folder to the ID of your site in the Business Manager.
4. Zip the **site\_import** folder.
5. In the Business Manager, go to **Administration** > **Site Development** > **Site Import & Export** and import the zipped file.

On successful import, it creates following highlighted metadata,

1. custom preferences(gp\_globalpay)

Graphical user interface, application, email

Description automatically generated

2. Payment Methods (GP\_DW\_PAYPAL,GP\_DW\_CREDIT\_CARD, GP\_DW\_GOOGLE\_PAY,GP\_DW\_APPLE\_PAY)

Graphical user interface, text, application, email

Description automatically generated

3.Payment Processor(GLOBALPAY\_CREDITCARD ,GLOBALPAY\_PAYPAL,GOOGLE\_PAY , GLOBAL\_APPLE\_PAY)

Graphical user interface, text, application

Description automatically generated

4.Service (int\_globalpay.http.generic)

Graphical user interface, text, application

Description automatically generated

# Set up the cartridge and Workspace

1. [Create a test account with Globalpay](https://developer.globalpay.com/). With this, you will be able to access the [test Customer Area](https://developer.globalpay.com/resources/test-card-numbers), and apply for a live account. While you wait for your live account credentials, you can start integrating by using your test account.
2. Finish the [metadata import](https://docs.adyen.com/plugins/salesforce-commerce-cloud/install-the-cartridge-and-import-the-metadata).

## Step 1: Set up the cartridge path

1. Go to **Administration** > **Sites** > **Manage Sites**>**[yourSite]**

Graphical user interface, application

Description automatically generated

1. Go to Setting tab In the**Cartridges**box:
   * [your custom SG cartridge]**int\_globalpay\_controllers:int\_globalpay:app\_storefront\_controllers:app\_storefront\_core**

Graphical user interface, text, application

Description automatically generated

1. Select **Apply**.

## Step 2: Set up the Business Manager

1. In the Business Manager, go to Merchant tools > Site Preferences > Custom Preferences > GlobalPay.
2. In the merchantCode box, enter the name of your Globalpay merchant account for which the payments will be processed.
3. Set Test/Production mode to (TEST).
4. Select Save in the upper-right corner of the screen.

## Step 3: Set up an API key

To connect your Commerce Cloud store to the Globalpay API, you need to get an API key from Globalpay need to update in to the Business Manager site preferences.

Generate an API key in the Globalpay Customer Area

1. Log in to your [Customer Area](https://developer.globalpay.com/), if you do not have account you can create the account in development portal.
2. Click on following profile iCone.

Graphical user interface, application

Description automatically generated

Click on ‘GP API Apps’, you can get merchant id and api key and Account & Permissions.

Graphical user interface, text, application

Description automatically generated

Graphical user interface, text, application

Description automatically generated

3.Copy the API key and Merchant key to the Business Manager site preference.

## Step 4: Set up workspace with Visual Studio code.

1. Create a folder “Globalpay” in your workspace and place the cartridge (int\_globalpay\_controller and int\_globalpay) downloaded from GitHub.
2. If you have a different project set-up, you will need to open the file ‘/package.json’ and modify the paths.base value to point to your ‘app\_storefront\_base’ cartridge. This path is used by the JS and SCSS build scripts.
3. If using VSCode, install the extension Prophet Debugger link or any other SFCC extension and include below in dw.json ().
4. {
5. "hostname": "your sandbox name",
6. "username": "sandbox username",
7. "password": " sandbox password",
8. "code-version": "code version"
10. }

NOTE: If you are using different IDE, refer respective developer guide to setup the workspace.

Step 3: Build and Upload the code Prerequisite:

Install node under "Link\_globalpay" folder.

Compile the Code

Compile JS and SCSS with following command

npm run compile:js && npm run compile:scss

Upload the code

npm run uploadCartridge

# Setting up the Payment Methods

## Payment methods

After the [metadata import](https://docs.adyen.com/plugins/salesforce-commerce-cloud/install-the-cartridge-and-import-the-metadata), Globalpay has been added as the payment processor for credit cards and for the local payment methods that have been [set up for your merchant account](https://docs.adyen.com/payment-methods#add-payment-methods-to-your-account) in the Adyen [Customer Area](https://ca-test.adyen.com/).

To make sure that this has been set up correctly:

1. In the Business Manager, go to **Merchant tools** > **Ordering** > **Payment Methods**.
2. In the **ID** column, select CREDIT*\_CARD*  and check that **Payment Processor** is set to **GLOBALPAY\_CREDITCARD**.

### Credit card Hosted fields.

Credit card payment optionally available in some countries. To learn more about their availability, see [Credit card Hosted fields](https://developer.globalpay.com/ecommerce/hosted-fields-guide).

STEP 1: Set up Payment Processor

1. To configure credit card payment: In the Business Manager, go to **Merchant tools** > **Ordering** > **Payment Methods**.
2. Select the ‘CREDIT\_CARD’id and enable or disable
3. In the Business Manager, go to **Merchant tools** > **Ordering** > **Payment Methods**.
4. In the **ID** column, select *GP\_DW\_GOOGLE\_PAY* and check that **Payment Processor** is set to **GLOBALPAY\_CREDITCARD**

Graphical user interface, text, application

Description automatically generated

STEP 2: Setup custom preferences

Go to Merchant Tools > Custom Preferences > gp\_globalpay and set values for the parameter:

|  |  |
| --- | --- |
| FIELD | DESCRIPTION |
| gp\_app\_id | A Global Payments created unique identifier to identify the application. |
| GP App Key | A string used to securely create access tokens. It must be securely stored and never be divulged to unauthorized third parties. |
| Gp MerchantId |  |
| Gp MerchantName |  |
| Gp google Env |  |

We are including globalpaycreditcards.isml in paymentmethods.isml (OOO file), on storefront we can see the screenshot as below which is iFrame, it is rendering from Globalpay. All card validations are happing through Globalpay libraries from Globalpay end.

Text

Description automatically generated

Storefront CreditCard form.

Graphical user interface, application

Description automatically generated

After entering credit card details (card#, exp date, CVV) in checkout page in payment form, we can see the call(payment-methods) in browser console in network section.

With help of GlobalPay plugin(globalpayments-3ds.js) we are invoking below 2 SFCC calls.

* Globalpay-Authentication
* Globalpay-Initiation

3DS feature:

3ds popup will appear only for specific cards. Please referee below url for more details. If the card is 3ds card, then on checkout page following screen and popup will appear. Before appearing the pop up we can see following calls in browser network console.

Graphical user interface, application

Description automatically generated

If the specified card is 3ds card, then we are making additional api(https://apis.sandbox.globalpay.com/ucp/authentications/<AuthId> /result) call to Globalpay.

Globalpay test cards:  
https://developer.globalpay.com/resources/test-card-numbers

Api url:  
<https://documenter.getpostman.com/view/12989945/UVJeGwg4>

If the specified card is 3ds 1 card, then the popup looks like below.

Graphical user interface, application

Description automatically generated

Globalpay test cards:  
https://developer.globalpay.com/resources/test-card-numbers

Api url:  
<https://documenter.getpostman.com/view/12989945/UVJeGwg4>

### Google Pay

Prerequisite:

Before proceeding to BM configuration, a Merchant Account needs to be created with Google. Please follow the [Link](https://support.google.com/paymentscenter/answer/7161426?hl=en) to create the merchant account with google.

To enable Google Pay we need to do the Payment Processor configuration as mentioned below,

1. In the Business Manager, go to **Merchant tools** > **Ordering** > **Payment Methods**.
2. Select the ‘GP\_DW\_GOOGLE\_PAY’ id and enable or disable
3. In the Business Manager, go to **Merchant tools** > **Ordering** > **Payment Methods**.
4. In the **ID** column, select *GP\_DW\_GOOGLE\_PAY* and check that **Payment Processor** is set to **GOOGLE\_PAY**

Graphical user interface, text, application, email

Description automatically generated

Go to Merchant Tools > Custom Preferences > gp\_globalpay and set values for the parameter

|  |  |
| --- | --- |
| Field | Description |
| enableGooglePay | Enable or Disable Google Pay Service |
| gp\_gpayMerchantId | Global pay Merchant account ID |
| gp\_gpayMerchantName | Matching setting on Google Account |
| gp\_gpayEnv | Test/Prod |
| gp\_gatewayMerchantId |  |

Once you complete the above setup you can see the below Google pay payment option on billing page

Graphical user interface, website

Description automatically generated

onclick of google pay button you will see different payment method option on google pay popup,

Graphical user interface, text, application

Description automatically generated

On successful completion of payment you will be redirecting to confirmation page where you can see the “Google Pay” as payment method.

Graphical user interface, text, application, email

Description automatically generated

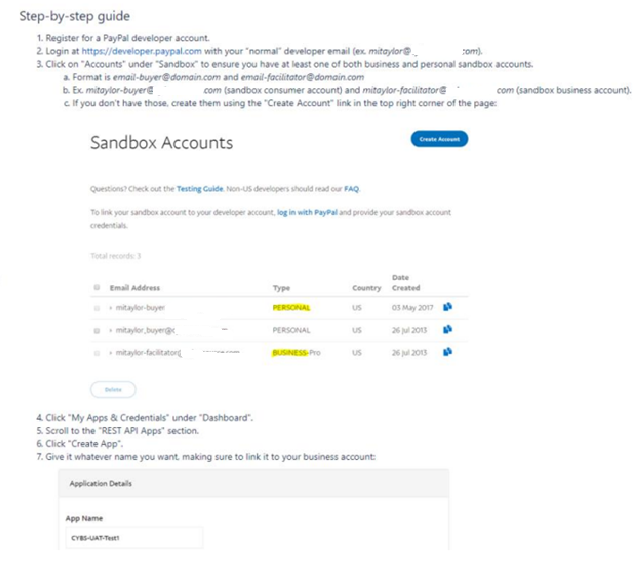
PayPal

Prerequisite

Prior to development phase, there are a generic set of configurations that a development team

needs to account for. These configurations include:

1. PayPal developer account
2. PayPal sandbox account Screenshot of the detailed set of configurations for #1 & #2.



3. Linking developer and sandbox account. On creating a PayPal developer account, get in

touch with the GlobalPay team, share the developer account details and get the

developers’ details configured on GlobalPay (BackOffice Configuration tool). Share

the following keys with GlobalPay:

### In the Business Manager, go to **Merchant tools** > **Ordering** > **Payment Methods**.

1. In the Business Manager, go to **Merchant tools** > **Ordering** > **Payment Methods**.
2. Select the ‘GP\_DW\_PAYPAL’ id and enable or disable
3. In the Business Manager, go to **Merchant tools** > **Ordering** > **Payment Methods**.
4. In the **ID** column, select *GP\_DW\_PAYPAL* and check that **Payment Processor** is set to **GLOBALPAY\_PAYPAL**

Graphical user interface, text, application, email

Description automatically generated

Once you complete the above setup you can see the below “Buy with PayPal” payment option on billing page

Graphical user interface, application

Description automatically generated

On click of Buy with PayPal you can see the below PayPal Login screen

Graphical user interface, application

Description automatically generated

On Click of Cancel your order will be cancelled and you will be redirected to cart page.

On Click Of Login you can see the below screen to pay with PayPal.

Graphical user interface, text, application

Description automatically generated

On successful completion of payment, you will be redirecting to confirmation page where you can see the “Pay Pal” as payment method.

Graphical user interface, text, application, email

Description automatically generated

### Apple Pay

1. In the Business Manager, go to **Merchant tools** > **Ordering** > **Payment Methods**.
2. In the **ID** column, select GP*\_DW\_APPLE\_PAY* and check that **Payment Processor** is set to **GOOGLE\_PAY**

Graphical user interface, text, application, email

Description automatically generated

Order Management

Refund Procedure:

1. In the Business Manager, go to **Merchant tools > Site Preferences > Custom Preference > Globalpay**
2. We can select captureMode as either ‘Auto’ or ‘Later’.
3. If the captureMode is ‘Auto’, and we place the order through storefront,  
   then the order payment status would be ‘paid’ otherwise payment status will be ‘not paid’.
4. If you refund the amount, then we can refund the with following url. Once the refund completes then the order status will be updated as ‘cancelled’. We can refund the amount only for the order, which has the payment status as paid and order status as ‘open’.

Graphical user interface, text, application, email

Description automatically generated

Graphical user interface, text, application, email

Description automatically generated

Capture Procedure:

1. In the Business Manager, go to **Merchant tools > Site Preferences > Custom Preference > Globalpay**
2. We can select captureMode as either ‘Auto’ or ‘Later’.
3. If the captureMode is ‘Later, and we place the order through storefront,  
   then order payment status as ‘not paid’.
4. Through the 3rd party we can capture the amount. Below is the url for capturing the amount.

Graphical user interface, text, application, email

Description automatically generated

Apple Pay with GlobalPay

1. Description

This document details the configurations required for Apple Pay through SFCC Business Manager and Apple Merchant Account. It will also brief

The setup required on Apple devices to use Apple Pay

2. Supported Devices

iPhone: iPhone SE, iPhone 6, iPhone 6Plus & higher with iOS 10 or later

iPad: iPad Pro, iPad Air 2, iPad mini 4, iPad mini 3 & higher with iOS 10 or later

Macbook: macOS v. 10.12. The user must have an iPhone or Apple Watch that can authorize the payment

Note: To troubleshoot issues from other regions, change the region setting on the devidde to a region where Apple Pay is supported

3. Apple Pay Requirements

You must have an Apple Developer Account (enrolled for development)

All paged that incorporate Apple Pay must be served over HTTPS

Your website must comply with the Apple Pay guidelines. For more information see Apple Pay on the Web Acceptable Use Guideline

Setting up Sandbox Domain with Apple Pay Merchant

Below are the steps mentioned to configure your SFCC Sandbox and Apple Pay account along with Payment Processor

4.1. Setting up Apple Pay Merchant

1. Login to Apple Merchant Center with registered Developer Account
2. In Member Center, select Certificates, Identifiers & Profiles
3. Under Identifiers, select Merchant IDs
4. Click on "+" sign on the Right Top corner to add new Merchant ID
5. Enter Merchant ID Description and Identifier on Registering a Merchant ID Page
   1. Don't use special characters like @, &, \*, ', "
   2. Click on Next
   3. Click Register on subsequent page

Graphical user interface, text, application, email

Description automatically generated

1. Select your Merchant ID created from the list and click Edit
2. In the Apple Pay on the Web Section, click on Add Domain
3. Enter your fully qualified domain name and click on Continue
4. This creates a file and makes it available to download and host it at the provided location i.e. https://<domain>/.well-known/apple-developer-merchantid-domain-associatio

Graphical user interface, text, application, email

Description automatically generated

Graphical user interface, application

Description automatically generated

**4.2. Adding Verification File on Domain**

**4.2.1. Adding verification file in Catalog**

1. In SFCC Business Manager, go to Merchant Tools > Products & Catalogs > Catalogs
2. Select storefront catalog
3. Create Category structure .well-known > apple-developer-merchantid-domain-association under root category
4. Use xml file similar to the one attached here:
5. Use a Category level custom attribute to add data present in the domain verification file downloaded from Apple Merchant Account mentioned in above steps
6. Sample rendering template attached here: apple-domain-verification.isml
7. Use a rendering template to include content from the above mentioned attribute on category landing page for .well-known/apple-developer-merchantid-domain-association

**4.2.2. Creating Alias to register domain**

1. Go to Merchant Tools > Site URLs > Aliases
2. Add alias as below in SFCC Business Manager

{

"\_\_version":"1",

"settings":{

"http-host":"development-pentland-ecommera.demandware.net",

"https-host":"development-pentland-ecommera.demandware.net"

},

"development-pentland-ecommera.demandware.net":[

{

"pipeline" : "Home-Show"

}

]

}

**4.2.3. Updating URL Rules**

* 1. Go to Merchant Tools > Site URLs > URL Rules > Catalog URLs
  2. Add rules as in below screenshot:

Graphical user interface, text, application, email

Description automatically generated

**4.2.5. Verifying domain**

* 1. Hit site url as https://<domain>/.well-known/apple-developer-merchantid-domain-association and check if content from domain verification file are getting loaded with a 200 OK status
  2. To check the network response, use one of the below:
     1. Use Browser extension for Link Redirect Trace:http://lrt.li/rtlogoappchrome
     2. Check online at http://www.redirect-checker.org/index.php
  3. Once this is done, go back to Apple Merchant Account and click on Verify

1. **Domain Registration in SFCC Business Manager**

1. Go to: “Merchant Tools > Site Preferences > Apple Pay

2. Under Domain Registration section

a. Click on Register Apple Sandbox under Apple Sandbox section for registering SFCC to Apple Sandbox account. 16 Visa Confidential

b. Click on Register Apple Production under Apple Production section for registering SFCC to Apple Production account.

Graphical user interface, text, application, email

Description automatically generated

1. **Configure Apple Pay in SFCC Business Manager Business Manager Configuration**

1. Go to: “Merchant Tools > Site Preferences > Apple pay

2. Check “Apple Pay Enabled?”

3. Fill in the “Onboarding” form: • Ensure “Apple Merchant ID” and “Apple Merchant Name” match settings in your Apple account

4. Fill in the “Storefront Injection” form:

• Selects where Apple Pay buttons should be displayed on your site.

5. Fill in “Payment Integration” form:

• Fill in Payment Provider URL

• Fill in Payment Provider User

• Fill in Payment Provider Password(some random Value)

• Ensure “Use Basic Authorization” is checked

6. Click "Submit"

Graphical user interface, text

Description automatically generated

Graphical user interface, text, application

Description automatically generated

1. Apple Pay on Demandware Storefront

To test ApplePay on Demandware site, following files need to be updated:

* + 1. Script – applepay.js
       - Update the file with below changes
* var Status = require('dw/system/Status');
* var server = require('server');
* var Transaction = require('dw/system/Transaction');
* var ApplePayHookResult = require('dw/extensions/applepay/ApplePayHookResult');
* var paymentMethodID = 'DW\_APPLE\_PAY';
* /\*\*
* \* @function getRequest hook is called whenever there is a new request on the site
* \*/
* exports.getRequest = function (basket, response) {
* session.custom.applepaysession = 'yes';   // eslint-disable-line
* var status = new Status(Status.OK   );
* var result = new ApplePayHookResult(status, null);
* return result;
* };

exports.authorizeOrderPayment = function (order, responseData) {

    var status = Status.ERROR;

    var authResponseStatus;

    var paymentMethod = require('dw/order/PaymentMgr').getPaymentMethod(paymentMethodID);

    setBillingAddress(responseData.payment.billingContact);

    setShippingAddress(responseData.payment.shippingContact);

    // eslint-disable-next-line

    Transaction.wrap(function () {

        //  lineItemCtnr.paymentInstrument field is deprecated.  Get default payment method.

        var paymentInstrument = null;

        // eslint-disable-next-line

        if (!empty(order.getPaymentInstruments())) {

            paymentInstrument = order.getPaymentInstruments()[0];

            paymentInstrument.paymentTransaction.paymentProcessor = paymentMethod.getPaymentProcessor();

        } else {

            return new Status(status);

        }

        paymentInstrument.paymentTransaction.paymentProcessor = paymentMethod.getPaymentProcessor();

    });

   // service logic import

  var globalpayAuthorization = require('../globalpayapplepay');

  var token = responseData.payment.token.paymentData;

  authResponseStatus= globalpayAuthorization.Authorize( order ,token);

    if (authResponseStatus) {

        status = Status.OK;

    }

    return new Status(status);

};

* 1. hooks.json Add hook for applepay at the end of file present at /cartridge/script
     + - {
       - "name": "dw.extensions.applepay.getRequest",
       - "script": "./cartridge/scripts/hooks/payment/processor/applepay"
       - },
       - {
       - "name": "dw.extensions.applepay.paymentAuthorized.authorizeOrderPayment",
       - "script": "./cartridge/scripts/hooks/payment/processor/applepay"
       - }
  2. Controller – GlobalPay-Authorization.js
* /\*\*
* \* GlobalPay-Authorization : The GlobalPay-Authorization endpoint invokes authorization call from applepay
* \* @name Base/GlobalPay-Authorization
* \* @function
* \* @memberof GlobalPay
* \*/
* server.post('Authorization', function (req, res) {
* //Returning Success in the basic Auth method
* return { success: true }

});

* 1. Actual Authorization Using GlobalPay API:
     1. Update the below file :
* /\*\*
* \* Authorizes a payment using a apple pay.
* \* @param {number} orderNumber - The current order'\
* \* s number
* \* @param {dw.order.PaymentInstrument} paymentInstrument -  The payment instrument to authorize
* \* @param {dw.order.PaymentProcessor} paymentProcessor -  The payment processor of the current
* \*      payment method
* \* @return {Object} returns an error object
* \*/
* function Authorize(order, paymentdata) {
* var globalpayconstants = require('\*/cartridge/scripts/constants/globalpayconstants');
* var globalPayPreferences = require('\*/cartridge/scripts/helpers/globalPayPreferences');
* var globalPayHelper = require('\*/cartridge/scripts/helpers/globalPayHelper');
* var URLUtils = require('dw/web/URLUtils');
* var BasketMgr = require('dw/order/BasketMgr');
* var currentBasket = BasketMgr.getCurrentBasket();
* var preferences = globalPayPreferences.getPreferences();
* var captureMode = preferences.captureMode;
* var HookManager = require('dw/system/HookMgr');
* var Locale = require('dw/util/Locale');
* var serverErrors = [];
* var applePayData = {
* account\_name: globalpayconstants.applePay.account\_name,
* channel: globalpayconstants.applePay.channel,
* type: globalpayconstants.applePay.type,
* capture\_mode: captureMode.value,
* amount: (order.totalGrossPrice) \* 100,
* currency: order.currencyCode,
* reference: order.orderNo,
* country: 'US',
* payment\_method: {
* name: order.customerName.replace(' ', ''),
* entry\_mode: globalpayconstants.applePay.entryMode,
* digital\_wallet: {
* provider: globalpayconstants.applePay.provider,
* //need to be removed once we get the solution for payment token
* payment\_token: {
* version: paymentdata.version,
* data: paymentdata.data,
* header: {
* ephemeralPublicKey: paymentdata.header.ephemeralPublicKey,
* transactionId: paymentdata.header.transactionId,
* publicKeyHash: paymentdata.header.publicKeyHash
* }
* }
* }
* }
* }
* var globalPayHelper = require('\*/cartridge/scripts/helpers/globalPayHelper');
* var PaymentInstrumentUtils = require('\*/cartridge/scripts/utils/PaymentInstrumentUtils');
* var applePayresp = globalPayHelper.applePay(applePayData);
* var orderUpdateResult = PaymentInstrumentUtils.ApplePaymentOrderUpdate(order, applePayresp);
* if (!orderUpdateResult) {
* var error = true;
* serverErrors.push(
* Resource.msg('error.technical', 'checkout', null)
* );
* }
* return orderUpdateResult;
* }
* exports.Authorize = Authorize;
  1. Now Applepay Can be tested in the DW storefront.