

Version 22.0.0 for Salesforce Commerce Cloud - SiteGenesis

# 

# 

# 

# Summary

This guide describes how to integrate Clearpay Payment Processor Version 22.0.0 in the SFCC reference application SiteGenesis V2.

Clearpay’s Version 22.0.0 cartridge provides the following features:

● Provides the Clearpay payment method during checkout

● Supports Clearpay’s v1.0 and v2.0 API

# Table of Contents

[Summary](#_jt10ax3c2qs7)

[Table of Contents](#_jhv9b7538x31)

[Component Overview](#_ebz76u79rfqt)

[Integration Guide for New Clearpay Merchants](#_q2pitw96wfa1)

[Testing](#_33ui88pawxbm)

[Features and Usage](#_bprujxyisxyo)

[Operations / Maintenance](#_f5vulortpxbu)

[Guide](#_ume50h2ezvz3)

[Release History](#_yx7nwbp2l7tr)

# 

# Component Overview

## Functional Overview

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

## Use Cases

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

## Limitations

The following Clearpay APIs are supported.

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

The following Clearpay 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.

# 

# 

# 

# Integration Guide for New Clearpay Merchants

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

## Download the v22.0.0 cartridge

The current version of the Clearpay cartridge is available directly from Clearpay.

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

## Import the v22.0.0 cartridge

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

### Import the Clearpay 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\_clearpay, int\_clearpay\_sg, and int\_clearpay\_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.

## 2. Import XML files for the site metadata

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

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

## 3. Custom Code Changes

In order to integrate the new Clearpay 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/\_clearpay.scss**

Copy the **\_clearpay.scss** file from the **reference** folder of the cartridge into **app\_storefront\_core/cartridge/scss/default/\_clearpay.scss**

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

Add to the top of the file import statement highlighted above:

|  |
| --- |
| var app = require('\*/cartridge/scripts/app'); var guard = require('\*/cartridge/scripts/guard'); var ClearpayUtilities = require('\*/cartridge/scripts/util/clearpayUtilities'); |

In the method resetPaymentForms(), add the command for the three conditions:

|  |
| --- |
| cart.removePaymentInstruments(cart.getPaymentInstruments('CLEARPAY')); |

|  |
| --- |
| 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));  cart.removePaymentInstruments(cart.getPaymentInstruments('CLEARPAY'));  } else if (app.getForm('billing').object.paymentMethods.selectedPaymentMethodID.value.equals(PaymentInstrument.METHOD\_CREDIT\_CARD)) {  app.getForm('billing').object.paymentMethods.bml.clearFormElement();    cart.removePaymentInstruments(cart.getPaymentInstruments(PaymentInstrument.METHOD\_BML));  cart.removePaymentInstruments(cart.getPaymentInstruments('PayPal'));  cart.removePaymentInstruments(cart.getPaymentInstruments('CLEARPAY'));  } else if (app.getForm('billing').object.paymentMethods.selectedPaymentMethodID.value.equals(PaymentInstrument.METHOD\_BML)) {  app.getForm('billing').object.paymentMethods.creditCard.clearFormElement();    if (!app.getForm('billing').object.paymentMethods.bml.ssn.valid) {  return false;  }    cart.removePaymentInstruments(cart.getPaymentInstruments(PaymentInstrument.METHOD\_CREDIT\_CARD));  cart.removePaymentInstruments(cart.getPaymentInstruments('PayPal'));  cart.removePaymentInstruments(cart.getPaymentInstruments('CLEARPAY'));  }  return true;  });    return status; } |

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

|  |
| --- |
| function handlePaymentSelection(cart) {  var clearpayBrandUtilities = ClearpayUtilities.brandUtilities;  var selectedPaymentMethodID = app.getForm('billing').object.paymentMethods.selectedPaymentMethodID.value;  var result;    if (empty(selectedPaymentMethodID)) {  if (cart.getTotalGrossPrice() > 0) {  result = {  error: true  };  } else {  result = {  ok: true  };  }  }    if ((selectedPaymentMethodID == 'CLEARPAY') && !clearpayBrandUtilities.isClearpayApplicable()) {  result = {  error: true  };  }    // skip the payment handling if the whole payment was made using gift cert  if (selectedPaymentMethodID.equals(PaymentInstrument.METHOD\_GIFT\_CERTIFICATE)) {  result = {  ok: true  };  }    if (empty(PaymentMgr.getPaymentMethod(selectedPaymentMethodID).paymentProcessor)) {  result = {  error: true,  MissingPaymentProcessor: true  };  }  if (!result) {  result = app.getModel('PaymentProcessor').handle(cart.object, selectedPaymentMethodID);  }  return result; } |

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

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

#### **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:

|  |
| --- |
| require("\*/cartridge/scripts/util/ClearpayUtilities.js").disableSummaryForClearpay(cart, viewContext); |

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

In the handlePayments() function, Replace the code block

|  |
| --- |
| if (authorizationResult.not\_supported || authorizationResult.error) {  return {  error: true  }; } |

With :

|  |
| --- |
| if (authorizationResult.not\_supported || authorizationResult.error) {  return {  authorizeError : authorizationResult.authorizationResponse,  error: true  }; } |

Then, near the bottom of the start() function, replace the code block

|  |
| --- |
| if (handlePaymentsResult.error) {  return Transaction.wrap(function () {  OrderMgr.failOrder(order);  return {  error: true,  PlaceOrderError: new Status(Status.ERROR, 'confirm.error.technical')  };  }); |

With :

|  |
| --- |
| if (handlePaymentsResult.error) {  return Transaction.wrap(function () {  OrderMgr.failOrder(order);  return {  error: true,  clearpayOrderAuthorizeError: handlePaymentsResult.authorizeError,  PlaceOrderError: new Status(Status.ERROR, 'confirm.error.technical')  };  }); |

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

Add the highlighted code:

|  |
| --- |
| function setSessionCurrency() {  var clearpayUtilities = require('\*/cartridge/scripts/util/clearpayUtilities');  var currencyMnemonic = request.httpParameterMap.currencyMnemonic.value;  var locale = request.httpParameterMap.locale.value;  var Response = require('\*/cartridge/scripts/util/Response');  var currency;    if (currencyMnemonic) {  currency = Currency.getCurrency(currencyMnemonic);  if (currency) {  session.setCurrency(currency);    Transaction.wrap(function () {  var currentCart = Cart.get();  if (currentCart) {  currentCart.updateCurrency();  currentCart.calculate();  }  });  }  }    if (clearpayUtilities.sitePreferencesUtilities.isClearpayEnabled() && locale) {  clearpayUtilities.brandUtilities.initBrand(locale);  }    Response.renderJSON({  success: true  }); } |

#### **6.** **File: app\_storefront\_core/cartridge/templates/default/ components/header/htmlhead\_UI.isml**

Locate the

|  |
| --- |
| <!--[if lt IE 9]> <script src="${URLUtils.staticURL('/js/lib/html5.js')}"></script> <![endif]--> |

and paste the code below:

|  |
| --- |
| <isinclude url="${URLUtils.url('Clearpay-IncludeClearpayLibrary')}" /> |

#### **7.** **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:

|  |
| --- |
| <isset  name="clearpayError"  value="${  !empty(pdict.ClearpayApiError) ? pdict.ClearpayApiError :  (!empty(request.httpParameterMap.get('clearpay')).stringValue ? request.httpParameterMap.get('clearpay').stringValue : null)  }"  scope="page" /> |

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

After the line:

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

Paste the following code:

|  |
| --- |
| <isscript>  var ClearpayUtilities = require("\*/cartridge/scripts/util/clearpayUtilities.js");  var brandUtilities = ClearpayUtilities.brandUtilities;    brandUtilities.initBrand(pdict.CurrentRequest.locale);    var clearpayApplicable = brandUtilities.isClearpayApplicable();  var apBrand = brandUtilities.getBrand();  var basketObject = dw.order.BasketMgr.getCurrentBasket();  var orderGrandTotal = basketObject.totalGrossPrice;  var thresholdUtilities = apBrandUtilities.getApiVersionDependentClass('\*/cartridge/scripts/util/thresholdUtilities');  var withinThreshold = thresholdUtilities.checkThreshold(orderGrandTotal);    clearpayApplicable = clearpayApplicable && withinThreshold.status && !withinThreshold.belowThreshold;    var clearpayPaymentName = ''; </isscript> |

After the line:

|  |
| --- |
| <isif condition="${paymentMethodType.value.equals(dw.order.PaymentInstrument.METHOD\_GIFT\_CERTIFICATE)}"><iscontinue/></isif> |

Paste the following:

|  |
| --- |
| <iscomment>Ignore the payment method CLEARPAY\_PBI if Clearpay payment method is disabled in configuration</iscomment> <isif condition="${(paymentMethodType.value.equals('CLEARPAY')) && !ClearpayApplicable}">  <iscontinue/> </isif>   <isif condition="${(paymentMethodType.value.equals('CLEARPAY'))}">  <isset name="clearpayPayment" value="${paymentMethodType.value}" scope="page"/> </isif> |

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

And replace with the following:

|  |
| --- |
| <div class="form-row label-inline">  <isset name="radioID" value="${paymentMethodType.value}" scope="page"/>  <div class="field-wrapper">  <input  id="is-${radioID}"  type="radio"  class="input-radio"  name="${pdict.CurrentForms.billing.paymentMethods.selectedPaymentMethodID.htmlName}"  value="${paymentMethodType.htmlValue}"  <isif condition="${paymentMethodType.value == pdict.CurrentForms.billing.paymentMethods.selectedPaymentMethodID.htmlValue}">  checked="checked"  </isif>  />  </div>  <label for="is-${radioID}">  <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', apBrand, null)}"/>  </isif>  </label> </div> |

Locate the following block of code:

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

And paste the following below that code block:

|  |
| --- |
| <iscomment>  Clearpay Pay Over Time  -------------------------------------------------------------- </iscomment> <isif condition="${clearpayApplicable && !empty(clearpayPayment)}">  <div class="payment-method <isif condition="${!empty(pdict.selectedPaymentID) && pdict.selectedPaymentID == clearpayPayment}">payment-method-expanded</isif>" data-method="${clearpayPayment}">  <isclearpaymessage  classname="checkout-clearpay-message"  totalprice="${orderGrandTotal}"  clearpaybrand="${apBrand}"  />  </div> </isif> |

#### 

#### **9.** **File: app\_storefront\_core/cartridge/templates/default/product/components/pricing.isml**

Locate this code:

|  |
| --- |
| <isinclude template="product/components/standardprice"/> |

And paste the code below:

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

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

|  |
| --- |
| <isset  name="isQuickView"  value="${  pdict.CurrentHttpParameterMap.source.stringValue == 'quickview' ||  pdict.CurrentHttpParameterMap.source.stringValue == 'cart' ||  pdict.CurrentHttpParameterMap.source.stringValue == 'giftregistry' ||  pdict.CurrentHttpParameterMap.source.stringValue == 'wishlist' ||  ''  }"  scope="page" />   <isscript>  var priceForAp;    if (pdict.ProductSetStandardPrice || pdict.ProductSetSalesPrice) {  priceForAp = pdict.ProductSetSalesPrice < pdict.ProductSetStandardPrice ? pdict.ProductSetSalesPrice : pdict.ProductSetStandardPrice;  } else {  priceForAp = pdict.Product.priceModel.isPriceRange() > 0 ? pdict.Product.priceModel.minPrice : (SalesPrice ? SalesPrice : StandardPrice);  } </isscript>   <isinclude url="${URLUtils.url('Clearpay-RenderMessage',  'totalprice', priceForAp,  'classname', 'pdp-clearpay-message',  'quickview', isQuickView,  'clearpayBrand', pdict.CurrentSession.privacy.clearpayBrand  )}" /> |

#### **10.** **File: app\_storefront\_core/cartridge/templates/default/product/producttile.isml**

Locate this code at the top of file:

|  |
| --- |
| <iscontent type="text/html" charset="UTF-8" compact="true"/> <isif condition="${(pdict.cache != null) ? pdict.cache : true}">  <iscache type="relative" hour="24" varyby="price\_promotion"/> </isif> |

Add this after:

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

Around line 239, comment out the following code:

|  |
| --- |
| prices.push(price); if (extraPrice) {prices.push(extraPrice);} |

Replace it by pasting in the following:

|  |
| --- |
| if (price && price.value) {  prices.push(price); } if (extraPrice && extraPrice.value) {  prices.push(extraPrice); } |

Around line 248, right before the line:

|  |
| --- |
| <iscomment>Promotion</iscomment> |

Paste in the following before:

|  |
| --- |
| <isscript>  var ClearpayUtilities = require('\*/cartridge/scripts/util/clearpayUtilities');  var BrandUtilities = ClearpayUtilities.brandUtilities;  var priceForAp = pdict.CurrentHttpParameterMap.pricerange.stringValue == 'true'? pdict.CurrentHttpParameterMap.minprice : prices[prices.length-1].value;    BrandUtilities.initBrand(pdict.CurrentRequest.locale); </isscript>   <isinclude url="${URLUtils.url('Clearpay-RenderMessage',  'totalprice', priceForAp,  'classname', 'plp-clearpay-message',  'clearpayBrand', pdict.CurrentSession.privacy.clearpayBrand  )}" /> |

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

Locate the div

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

And add the below mentioned code before the div:

|  |
| --- |
| <iscomment>Clearpay message</iscomment> <isif condition="${pdict.Basket.totalGrossPrice.available}">  <isset name="orderTotalValue" value="${pdict.Basket.totalGrossPrice}" scope="page"/> <iselse/>  <isset name="orderTotalValue" value="${pdict.Basket.getAdjustedMerchandizeTotalPrice(true).add(pdict.Basket.giftCertificateTotalPrice)}" scope="page"/> </isif>   <isscript>  var ClearpayUtilities = require('\*/cartridge/scripts/util/clearpayUtilities');  var BrandUtilities = ClearpayUtilities.brandUtilities;    BrandUtilities.initBrand(pdict.CurrentRequest.locale); </isscript>   <div class="learn-more">  <isinclude url="${URLUtils.url('Clearpay-RenderMessage',  'totalprice', orderTotalValue,  'classname', 'cart-clearpay-message',  'clearpayBrand', pdict.CurrentSession.privacy.clearpayBrand  )}"  /> </div> |

#### **12.** **File: app\_storefront\_core/cartridge/templates/default/util/modules.isml**

At the very bottom of the file, paste the following:

|  |
| --- |
| <iscomment> Render the Clearpay modules </iscomment> <isinclude template="util/modulesclearpay" /> |

#### **13.** **File: app\_storefront\_core/cartridge/scripts/util/Resource.ds**

Find the line:

|  |
| --- |
| ResourceHelper.getPreferences = function(pageContext) { |

Add the highlighted code inside the method:

|  |
| --- |
| ResourceHelper.getPreferences = function(pageContext) {  var cookieHintAsset = ContentMgr.getContent('cookie\_hint');  var consentTrackingHintAsset = ContentMgr.getContent('consent\_tracking\_hint');  var ClearpayUtilities = require('\*/cartridge/scripts/util/clearpayUtilities.js');  var brandUtilities = ClearpayUtilities.brandUtilities;  return {  LISTING\_INFINITE\_SCROLL: (Site.getCurrent().getCustomPreferenceValue('enableInfiniteScroll') ? true : false),  LISTING\_REFINE\_SORT: true,  STORE\_PICKUP: Site.getCurrent().getCustomPreferenceValue('enableStorePickUp'),  COOKIE\_HINT: (cookieHintAsset && cookieHintAsset.online) || false,  CONSENT\_TRACKING\_HINT: (consentTrackingHintAsset && consentTrackingHintAsset.online) || false,  CHECK\_TLS: Site.getCurrent().getCustomPreferenceValue('checkTLS'),  CLEARPAY\_PORTAL: brandUtilities.getBrandSettings().javaScriptUrl  }; } |

#### **14.** **File: app\_storefront\_core/cartridge/js/app.js**

Find the below mentioned code

|  |
| --- |
| $('.user-account').on('click', function (e) { |

And add this snippet after the method :

|  |
| --- |
| $(document).on('click', '.clearpay-link', function (e) {  e.preventDefault();  dialog.open({  url: $(e.target).attr('href')  });  }); |

#### **15.** **File: app\_storefront\_core/cartridge/js/bonus-products-view.js**

Add the following code:

|  |
| --- |
| function createOrder() {  var url = util.ajaxUrl(Urls.createClearpayOrder);    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.CLEARPAY\_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"]');    $btnContinue.on('click', function(e) {  var methods = ['CLEARPAY'];  if (methods.indexOf($paymentMethods.filter(':checked').val()) !== -1) {  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();  initSubmitOrderEvent(); };   module.exports.initSubmitOrderEvent = function () {  initSubmitOrderEvent(); }; |

#### **16.** **File: app\_storefront\_core/cartridge/js/countries.js**

Add the highlighted code :

|  |
| --- |
| use strict'; exports.init = function init () {  $('.country-selector .current-country').on('click', function () {  $('.country-selector .selector').toggleClass('active');  $(this).toggleClass('selector-active');  });  // set currency first before reload  $('.country-selector .selector .locale').on('click', function (e) {  e.preventDefault();  var url = this.href;  var currency = this.getAttribute('data-currency');  var locale = this.getAttribute('data-locale');  $.ajax({  dataType: 'json',  url: Urls.setSessionCurrency,  data: {  format: 'ajax',  currencyMnemonic: currency,  locale: locale  }  })  .done(function (response) {  if (!response.success) {  throw new Error('Unable to set currency');  }  window.location.href = url;  });  }); }; |
|  |

#### **17.** **File: app\_storefront\_core/cartridge/scss/default/style.scss**

At the end of the file, paste the following:

|  |
| --- |
| @import "clearpay"; |

## 4. 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 |

## 5. Upload Modified Cartridge

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

## 6. Activate the Cartridge In Business Manager

### Set Cartridge Path

Before the Clearpay functionality can become available to SiteGenesis, the cartridges have to be added to the cartridge path of the Site in question. In order to do this, follow the following instructions:

1. Log into Business Manager
2. Navigate to Administration > Sites > Manage Sites.
3. Click on the site name and on the next page go to the Settings tab.
4. In the textbox Cartridges add “:**int\_clearpay\_sg:int\_clearpay\_core**” in front of Sitegenesis base cartridges.s
5. Click Apply.
6. Repeat steps 3 to 6 for each site that is to use Clearpay.
7. Go to “Manage the Business Manager site”
8. In the textbox Cartridges add “:bm\_clearpay: int\_clearpay\_core”.

### 

### Set Clearpay 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\_Clearpay is available. Please select it and edit the attributes according to your Clearpay account data and the data shown below.

|  |  |  |
| --- | --- | --- |
| Site Preference | Description | Default |
| Enable Clearpay  **(enableClearpay)** | The field determines if the Clearpay is enabled / disabled in the cartridge. | False |
| Clearpay 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. |
| 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 |
| 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 |  |
| Brand settings **(apBrandSettings)** | Clearpay Brand related settings  in JSON format where  ● brand – clearpay, depending on region;  ● service – the field is used to identify the service with Clearpay.  ● versionApi – 1/2 API version. Please make sure that URLs in service corresponds to the chosen API version  ● javaScriptUrl – The endpoint where the javascript library is obtained from.    **Sandbox**:  UK  <https://portal.sandbox.afterpay.com/afterpay.js>  Europe  <https://merchant-tools.sandbox.clearpay.com/clearpay.js>    **Production**:  UK<https://portal.afterpay.com/afterpay.js>  Europe  <https://merchant-tools.clearpay.com/clearpay.js>  ● learnMoreUrl – the URL that links to more information about Clearpay that users can click to view. | <default value for RefArch site. Other countries examples could be found in RefArchGlobal site meta>  {  "GB": {  "brand": "clearpay",  "service": " clearpay.service.GB.v2",  "versionApi": "2",  " javaScriptUrl": "https://portal.sandbox.clearpay.co.uk/afterpay.js",  "learnMoreUrl": "https://www.clearpay.co.uk/en-GB/terms-of-service"  }  } |

### 

### Set Clearpay Payment Methods

Check that the payment method with the name CLEARPAY is available.

### 

### Set Clearpay Payment Processors

Check that a payment processor with the name CLEARPAY\_CREDIT is available that was imported.

### 

### Set SFCC(Demandware) Service

Services for GB, IT, ES and FR countries are available. Select the Credentials tab and select required service and enter the account details.

Administration > Operations > Services > Credentials select required service and enter the account details.

|  |  |
| --- | --- |
| URL | This is the Clearpay API endpoints for:  United Kingdom:  Sandbox  <https://api.eu-sandbox.afterpay.com/v1/>  <https://api.eu-sandbox.afterpay.com/v2/>  Production  <https://api.eu.afterpay.com/v1/>  <https://api.eu.afterpay.com/v2/>  EU:  Sandbox  <https://api.sandbox.clearpay.com/v1/>  Production  <https://api.clearpay.com/v1/> |
| User | Enter the Merchant ID provided by Clearpay |
| Password | Enter the Secret Key provided by Clearpay |

### Add Clearpay Image to Payment Method

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

● Merchant Tools > Ordering > Payment Methods

● Select the payment method with ID CLEARPAY and locate the image attribute and upload the image included in the cartridge (**int\_clearpay\_core/cartridge/static/default/images/clearpay-logo.png**).

### 

### Modify Content

Job

This cartridge supports an Clearpay order status of PENDING by running a Job Schedule against all Clearpay Orders that are in a PENDING state. This is not required at this point as Clearpay will only respond with APPROVED or DECLINED and if the order is DECLINED the checkout cannot complete.

By default Job Step is set for SiteGenesis site. You may change this to your site.

# 

# 

# 

# 

# 

# 

# 

# 

# Features and Usage

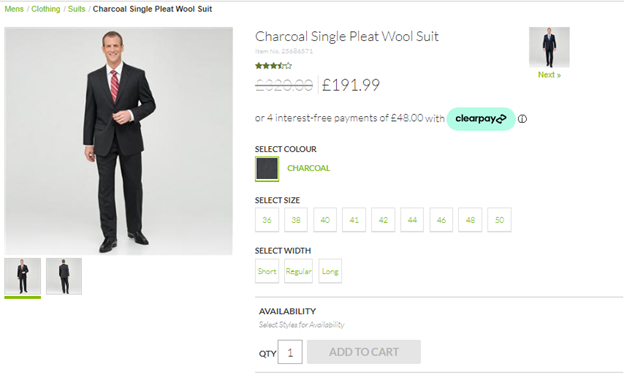
## Clearpay Payment on Checkout

When the shopper selects the Clearpay payment method at checkout and clicks the **Place Order** button, they will be redirected to Clearpay 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

## Clearpay Product Messaging

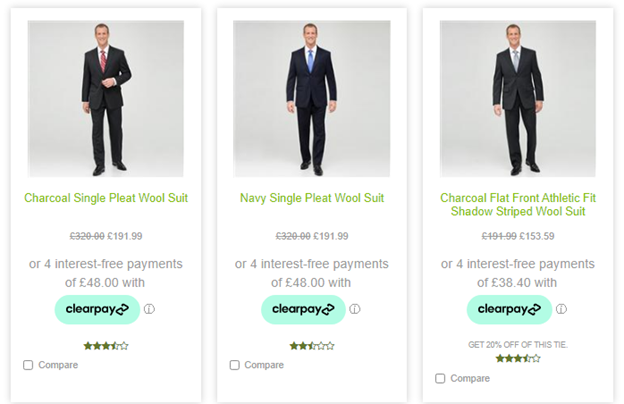
Clearpay adds messaging on various pages in the store.

### Messaging on Product Details Page:

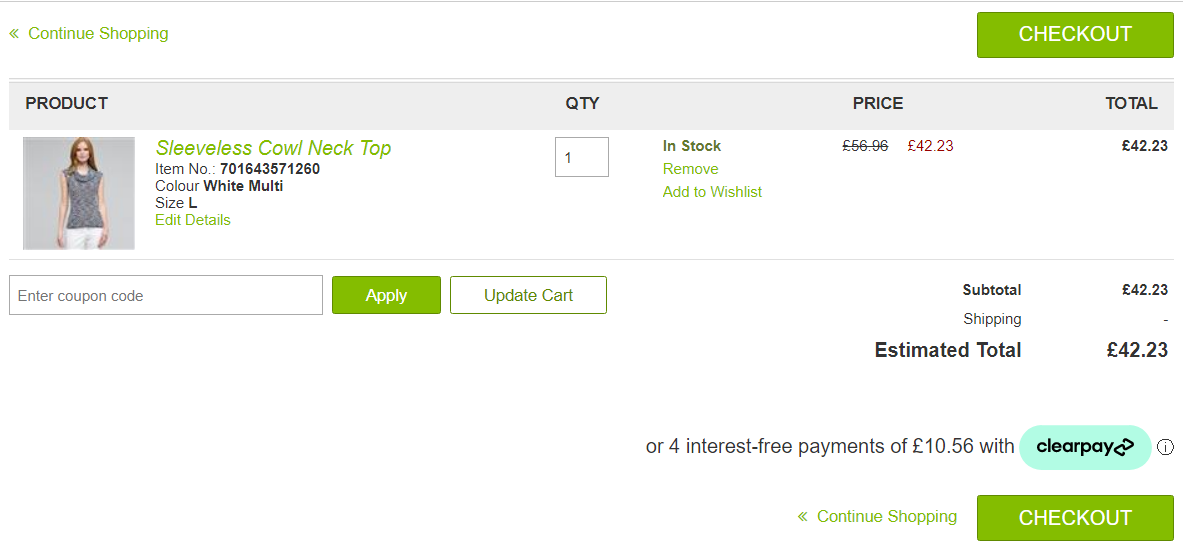


### 

### Messaging on Product List Page:



### Messaging on Cart Page:



# 

# Testing

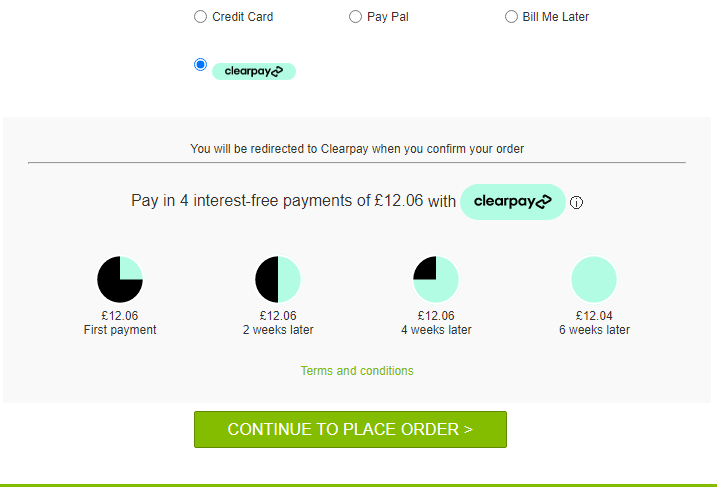
In Business Manager, navigate to the SiteGenesis Site > Site Preferences > Custom Preferences. A custom site preference group with the ID Integration\_Clearpay is available.

Please select it and add the respective data (obtained from Clearpay). The service credentials also need to be updated with an Account ID and Key provided by Clearpay. The Service URL needs to be set to the relevant region and environment.

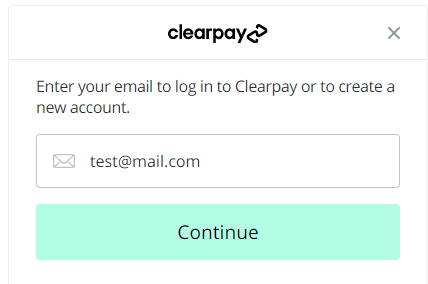
## Checkout Flow

Once the cartridge is installed and integrated based on the instructions, please try to place an order on your sandbox to test the functionality.

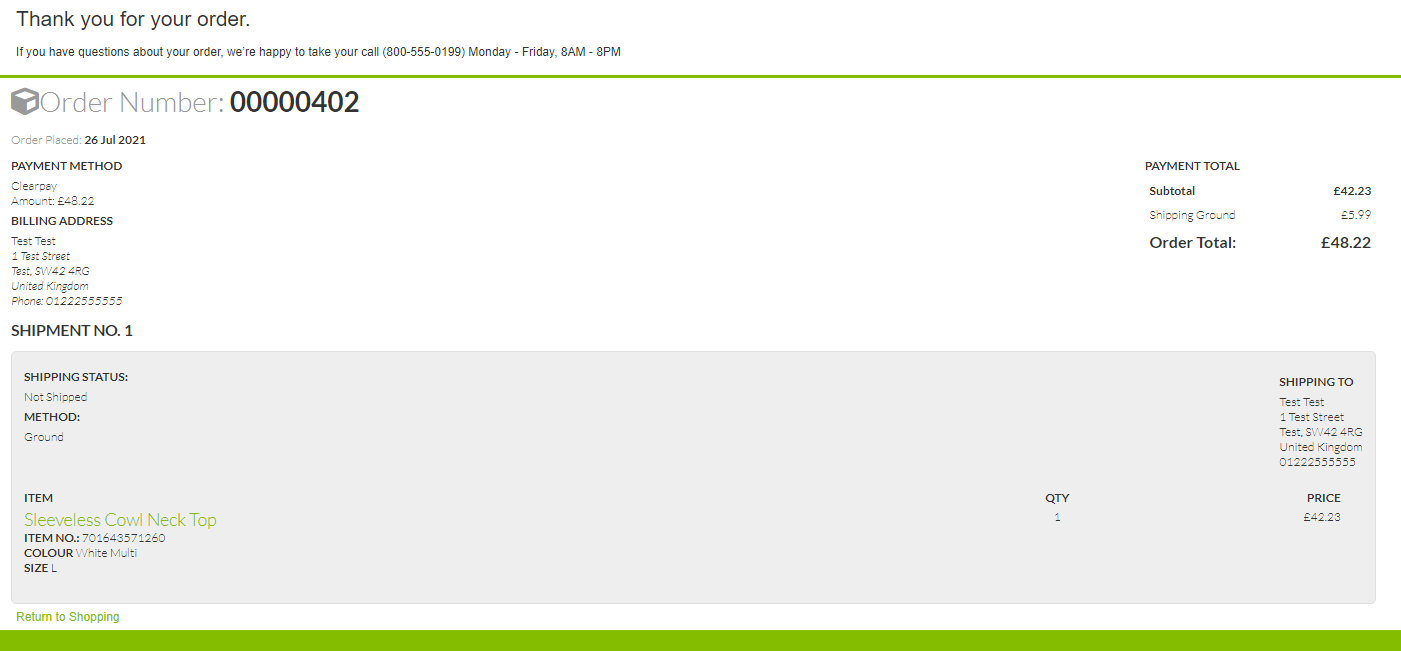
On billing page, you should see following option of Clearpay:



After you click on Place order then you should be redirected to Clearpay Payment Page. If you are not registered then you will be asked to create an account otherwise you see the following login page:

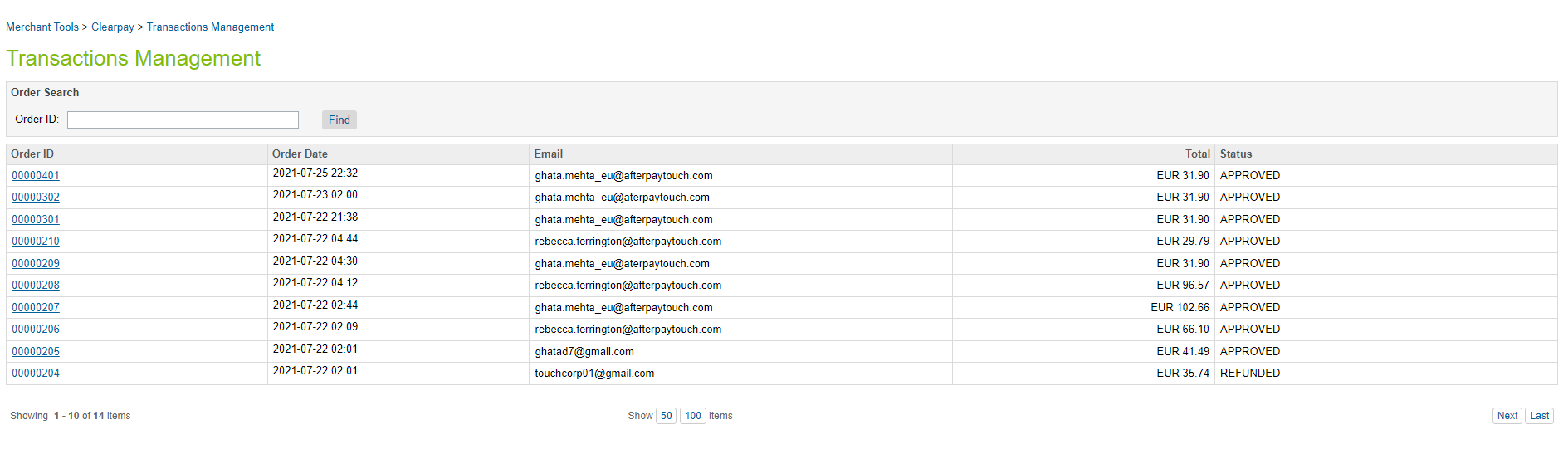


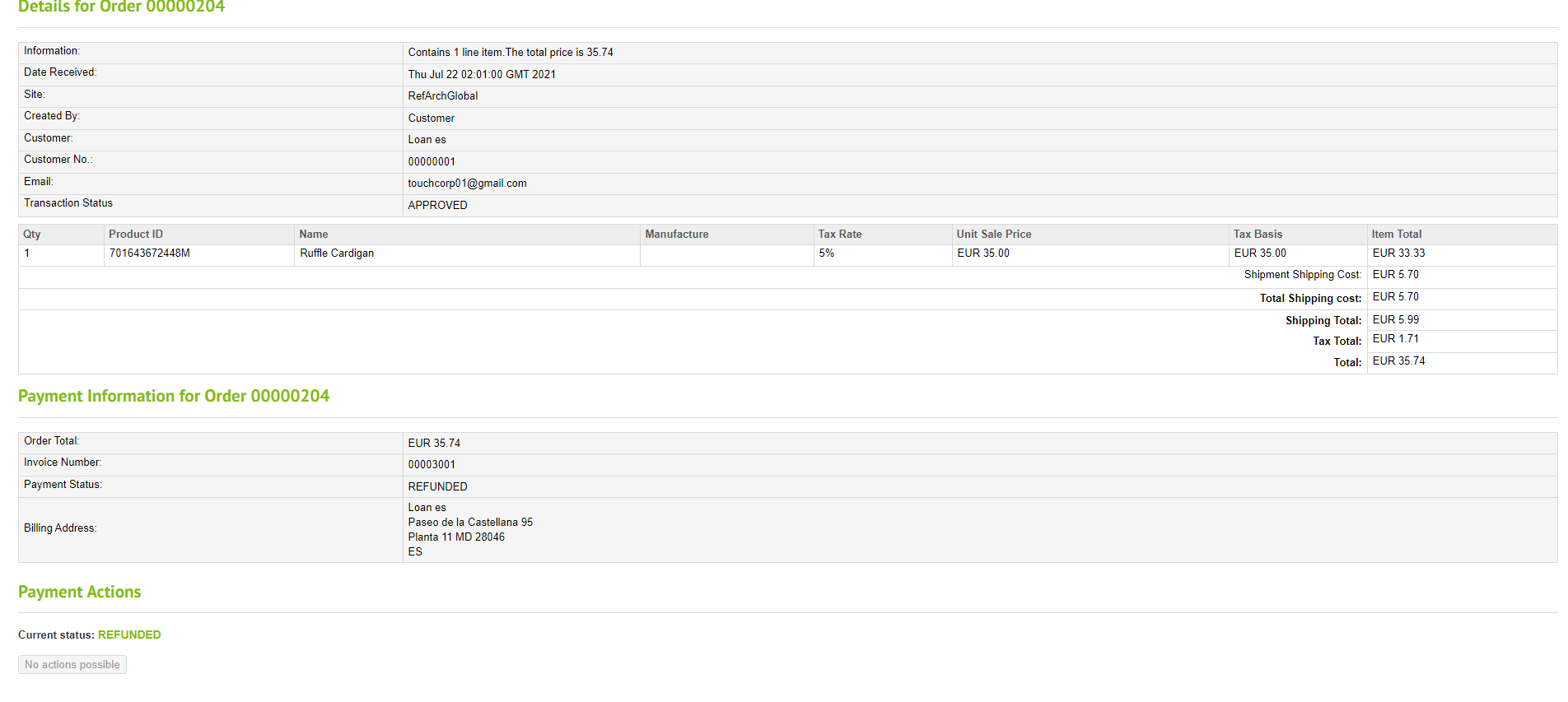
Once you logged in successfully, you will be asked to enter payment details. After payment you will be redirected to Commerce Cloud storefront confirmation page like one below:



Refund Transaction

Merchant tools > Clearpay > Transactions Management > Click on the “APPROVED “ Order ID for which you want to refund the transaction.





# 

# 

# Guide

## Roles, Responsibilities

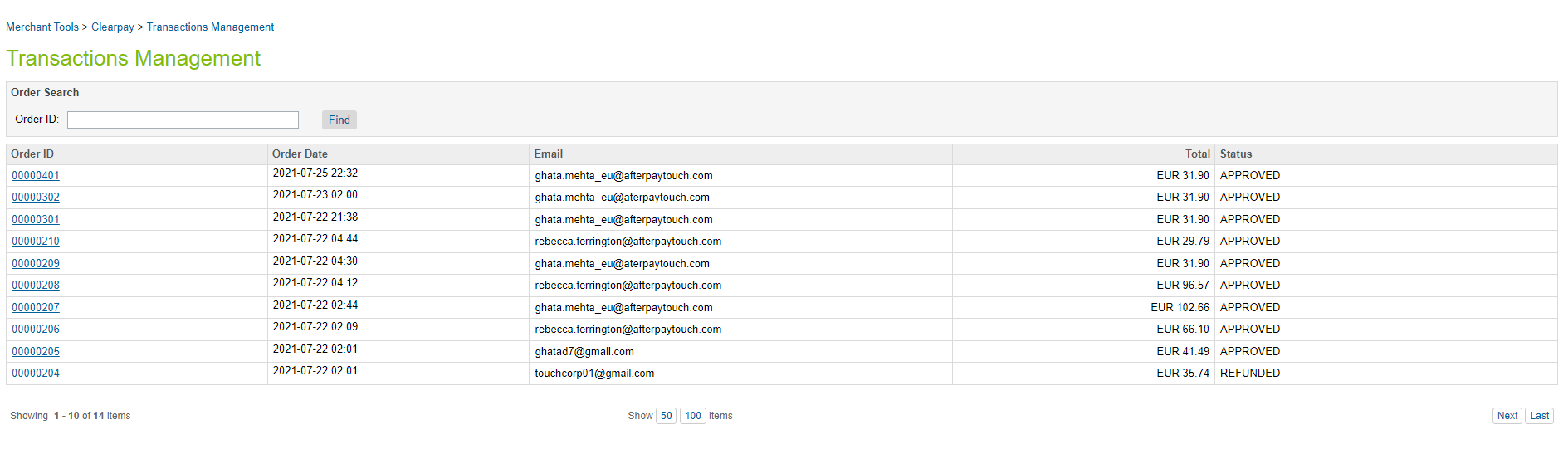
Integration of this cartridge will typically be done by a SFCC(B2C Commerce) developer.

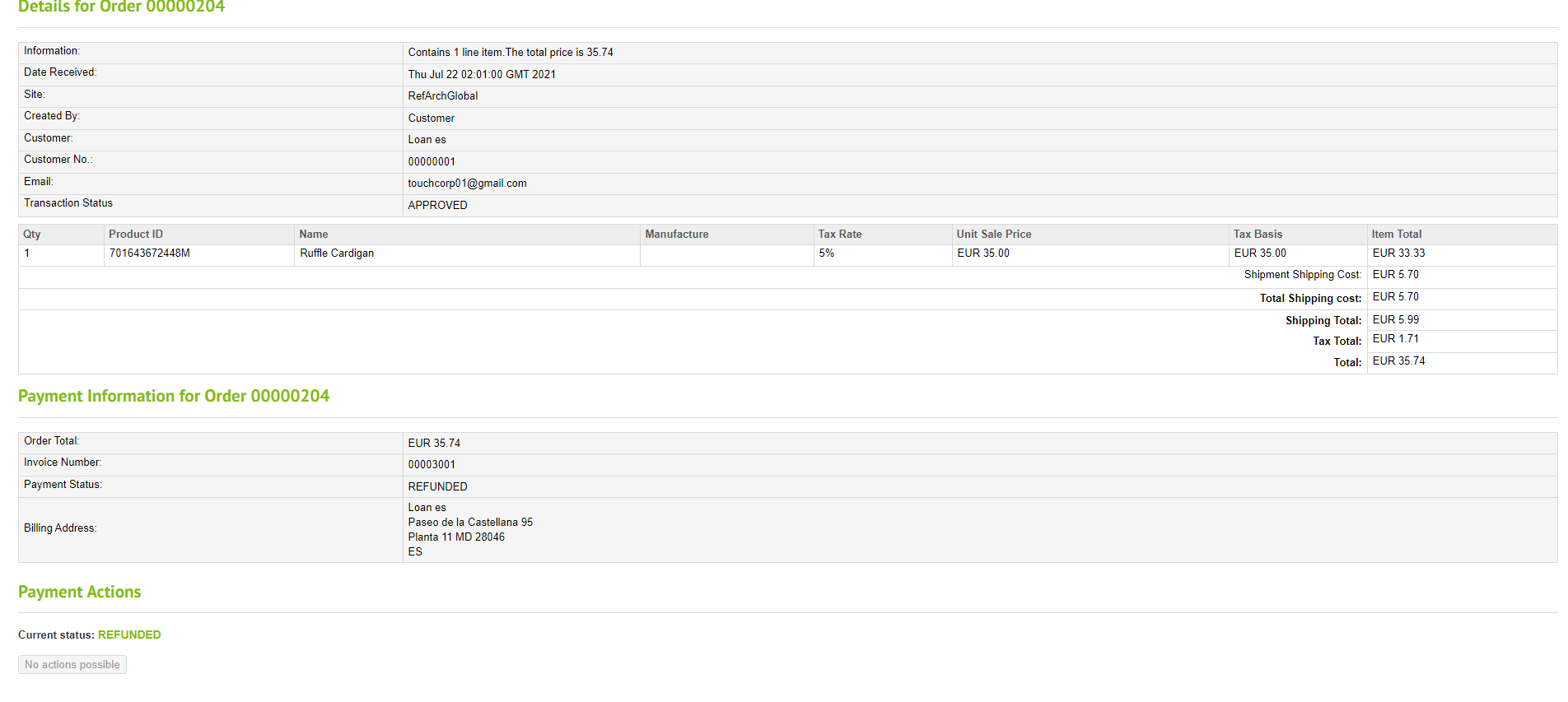
Clearpay will provide access keys to be used with the API.

## Business Manager

For the Clearpay integration to work, the following needs to be configured in the Business manager.

1. Upload the cartridge **bm\_clearpay** and **int\_clearpay\_core** into the server.
2. Go to B2C Commerce > Administration > Manage Sites > Business Manager > Settings > Cartridges input field add cartridge name “**bm\_clearpay: int\_clearpay\_core**”.
3. Administration > Organization > Roles > Administrator > Business manager modules > Select context(select your site ) > enable the Clearpay transactions management .
4. Merchant tools > Clearpay > transactions Management > Click on the “APPROVED” order ID for which you want to refund the transaction.





# 

# 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

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

Availability

If the Clearpay 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 Clearpay account contact.

# 

# Release History

|  |  |  |
| --- | --- | --- |
| Version | Date | Changes |
| 22.0.0 |  | Initial Release |

### 