**HiPay Integration**

Version 18.1.0

****

Table of Contents

[1. Summary 1-4](#_Toc497987329)

[2. Component Overview 2-5](#_Toc497987330)

[2.1 Functional Overview 2-5](#_Toc497987331)

[2.1.1 Frontend and Business Manager Functionalities 2-5](#_Toc497987332)

[2.1.2 Security Features and Functionalities 2-6](#_Toc497987333)

[2.1.3 Payment Configurations 2-6](#_Toc497987334)

[2.2 Use Cases 2-7](#_Toc497987335)

[2.3 Limitations, Constraints 2-11](#_Toc497987336)

[2.4 Compatibility 2-12](#_Toc497987337)

[2.5 Privacy, Payment 2-12](#_Toc497987338)

[3. Implementation Guide 3-13](#_Toc497987339)

[3.1 Integration Efforts 3-13](#_Toc497987340)

[3.2 Setup 3-13](#_Toc497987341)

[3.2.1 Installation 3-13](#_Toc497987342)

[3.2.2 Metadata Import 3-13](#_Toc497987343)

[3.3 Configuration 3-14](#_Toc497987344)

[3.4 Custom Code 3-16](#_Toc497987345)

[3.4.1 HiPay Multi-account 3-44](#_Toc497987346)

[3.5 External Interfaces 3-45](#_Toc497987347)

[3.6 Testing 3-45](#_Toc497987348)

[4. Operations, Maintenance 3-46](#_Toc497987349)

[4.1 Data Storage 3-46](#_Toc497987350)

[4.1.1 Orders 3-46](#_Toc497987351)

[4.1.2 ClearHungOrders job 3-47](#_Toc497987352)

[4.2 Availability 3-47](#_Toc497987353)

[4.3 Support 3-48](#_Toc497987354)

[5. User Guide 5-49](#_Toc497987355)

[5.1 Roles, Responsibilities 5-49](#_Toc497987356)

[5.2 Business Manager 5-49](#_Toc497987357)

[5.2.1 HiPay business module 5-49](#_Toc497987358)

[5.2.2 HiPay Site Preferences 5-52](#_Toc497987359)

[5.2.3 Services 5-54](#_Toc497987360)

[5.2.4 Schedules 5-56](#_Toc497987361)

[5.2.5 Payment Processors 5-56](#_Toc497987362)

[5.2.6 Payment Methods 5-56](#_Toc497987363)

[5.2.7 Logs 5-57](#_Toc497987364)

[5.2.8 Notification url 5-57](#_Toc497987365)

[5.2.9 Custom CSS configuration 5-58](#_Toc497987366)

[5.3 Storefront Functionality 5-59](#_Toc497987367)

[5.3.1 Merchant Payment API - Credit Cards 5-59](#_Toc497987368)

[5.3.2 Payment API - Merchant Link Credit Cards (with 3DSecure) 5-59](#_Toc497987369)

[5.3.3 Hosted Merchant Link 5-60](#_Toc497987370)

[5.3.4 Hosted Merchant Link Credit Cards 5-60](#_Toc497987371)

[5.3.5 Hosted Merchant iFrame 5-61](#_Toc497987372)

[6. Known Issues 6-63](#_Toc497987373)

[7. Release History 7-63](#_Toc497987374)

# Summary

HiPay Fullservice offers tailored payment solutions for retailers, with functionalities adapted to today’s e-commerce landscape. HiPay provides a single integration solution for the most relevant domestic and international payment methods in each market.

The HiPay cartridge enables merchants to use HiPay full service as a payment option on their Salesforce Commerce Cloud storefronts. The cartridge provides several options for entering payment information (credit card or debit card information) during the checkout process: through a hosted page that is, external to the Salesforce Commerce Cloud platform or directly on the merchant’s Salesforce Commerce Cloud storefront, through API module calls. Beside this, a full range of other payment methods can be supported.

This document fully describes how to install the cartridge and integrate it into the online store. It is required for the merchant to first contact HiPay and request an account for the integration to work properly. HiPay shall provide a testing and production accounts for the merchant.

There are also configurations to be made in both Salesforce Commerce Cloud Business Manager and HiPay back office.

The integration consists of an archive, which contains the following contents:

* cartridge called ‘int\_hipay\_controllers’ – core integration cartridge
* cartridge called ‘bm\_hipay\_controllers’ - extension for the cartridge within Business Manager
* ‘site\_template’ folder with metadata

The integration is based on the Site Genesis demo store provided by Salesforce Commerce Cloud.

# Component Overview

## Functional Overview

This section describes the functionalities that HiPay cartridge offers for the merchant.

### Frontend and Business Manager Functionalities

**Business Manager Extension for HiPay**

Merchants can configure the HiPay cartridge features from the Salesforce Commerce Cloud BM. On the Business Manager page each merchant can configure the connection to HiPay. The merchant will have to enter his credentials to create a connection to HiPay.

**Hosted Pages**

The cartridge supports the hosted payment page approach of entering payment information directly on pages that are hosted and provided by HiPay. During the checkout process, when the customer is asked to enter the payment details, he/she will be redirected to the HiPay hosted payment pages where he/she can select the preferred payment method. Each merchant has the ability to choose what payment methods shall be offered to the customer. The hosted payment pages can be customised in order to match the merchant storefront style.

**iFrame Integration**

The cartridge enables the merchant to use the HiPay payment solution via an iFrame that is integrated into his own Salesforce Commerce Cloud implementation.

**Multiple Payment Methods**

The cartridge allows users to complete the checkout process by using a wide variety of payment methods.

**Direct Payment API**

The cartridge offers to customers the possibility of filling in the payment information directly on the merchant’s website; in this case, the merchant will have to be PCI compliant. The transaction will be validated through the module called HiPay Fullservice API.

The cartridge can provide several options for entering payment details (credit card or debit card information) during the checkout process: through a hosted page, external to Salesforce Commerce Cloud platform or directly on the merchants’ Salesforce Commerce Cloud storefront, through API module calls. Additionally, a full range of other payment methods can be supported.

### Security Features and Functionalities

**3D Secure**

The cartridge supports 3D secure rules and activation where it is available on the card used. The 3D secure authentication process redirects the customer to a special page that is hosted by the card company that issued the customers’ credit or debit card. The cartridge enables the redirection to that page. After the authentication is completed, the cartridge has to ensure that the result of the authentication process is transferred back to Salesforce Commerce Cloud, and after the process is completed, the right message will be displayed on the frontend and the status of the order is set accordingly.

When using 3D Secure the merchant can set specific rules. These rules can be created, edited or deleted in the configuration parameter named “rules 3d secure”.

**Device Fingerprint**

The cartridge supports device fingerprint information by sourcing the dynamically generated JavaScript from HiPay TPP. The JavaScript determines the available information and creates or generates a so called “black box”. For the usage of this functionality custom configurations are needed which require individual implementation for each merchant, as they are not part of the standard development.

**Signature Verification**

A signature verification mechanism is available for verifying the content of requests and redirections between the merchant site and the HiPay pages. The first step is for the merchant to set a secret password or phrase in the HiPay backend. The secret password or phrase will be used to generate a unique string that will be hashed with a SHA algorithm.

**Automatic Cancel**

Pending orders are automatically cancelled if they are not completed within 30 minutes.

### Payment Configurations

These cartridge supports both authorization mode and direct capture mode.

**Direct Capture**

In this case the capture is requested automatically after the authorization. If the capture is not successful, the customer (the buyer) is redirected to an error page. If the capture is successful, the customer is redirected to a success page and completes the checkout process.

**Authorization**

In this case the customer (the buyer) is not charged automatically or direct after the order is placed. The merchant has a defined amount of time to capture the amount money from the customer. This can be configured manually from Business Manager > HiPay Integration > Order capture; partial or full capture action can be done; Order Cleanup Time can be configured from **Hung Order Cleanup Time** configuration option.

If the transaction fails, the customer is redirected to an error page. If the transaction is successful, the customer is redirected to a success page. The authorization can be used within an interval of 7 days otherwise it will be cancelled.

## Use Cases

**Short description of the HiPay payment system**

HiPay generally can receive payments in 2 ways:

* via the HiPay Direct Payment API. A result is returned right away. This can only be used for Credit Cards.
* via the 'redirect model' where a form containing certain fields is posted to HiPay and the shopper is redirected to HiPay's Hosted Payment Page (HPP). After completing the payment, the shopper is returned to the shop's resultURL. This resultURL can be configured and contains the result of the payment (success or fail). The redirect model can also be implemented in an iFrame solution where the merchants’ customer is redirected to HiPay pages, but this pages render within a frame that is inserted inside the merchant’s page. This approach gives the merchant’s customer the feeling that he has not left the merchant’s site.

Compared to the API implementation, the redirect model supports more payment methods as well as some other functionality.

Payment results should be sent asynchronously to a 'notification URL' at the Merchant (DW) site. Payments can change status over the time (Authorised, Refunded, Cancelled and others) on the HiPay side, and these statuses are updated automatically on the Salesforce Commerce Cloud side through the sending update notifications. HiPay Fullservice is sending notifications to Salesforce Commerce Cloud for each event occurred/ transaction changed. In order to handle all notifications, notes are added to the order.

In both cases, the 'two phase ordering' approach is used. The order is first created (but only 'placed', status is “Created”) and after the payment has been authorized (which is now standard behaviour in Site Genesis) the order can be captured and its status will change to „Paid”. In cases where payment cannot be authorized, the order will fail and its status will change to “Failed”.

Registered and unregistered shoppers will be able to follow the standard Site Genesis flow with the following changes:

HiPay redirect (hosted payment pages, iFrame integration and multiple payment methods) and the HiPay Direct Payment API will be the only payment option.

**If HiPay redirect is enabled**

* when the user selects this payment method, no credit card details will be entered on the Billing page (this is done on HiPay hosted payment pages)
* the possibility to choose one of the predefined payment methods on the Billing page with further redirection to HiPay hosted payment pages, if Directory Lookup is enabled and the user selected HiPay payment method
* a redirect to HiPay hosted payment pages after clicking the ‘SUBMIT ORDER’ button on the Order Confirmation page
* a return to the DW Order Summary page after successful payment authorization
* a return to the DW Order Confirmation page, if the user cancelled the payment on HiPay hosted payment pages
* a return to the DW Order Confirmation page with an error message displayed, if the payment was refused

**If HiPay API payments are enabled**

* credit card details are entered and stored in your DW shop
* for the other payment methods, you can still redirect to the HiPay hosted payment pages, but can disable credit cards via the configuration of your skin

**Sample of testing scenarios**

* All test scenarios involve successful and declined payments
* Test scenarios should be done with every available payment method
* The used currency should be EUR, USD has only credit cards enabled
* If for a payment method on a hosted page there is a emulation page, all options (e.g. cancel, exception, decline) should be used as an outcome to test how the system will behave
* When making a payment a Salesforce Commerce Cloud Order object changes its status as follows
  + CREATED - the IFrame or Hosted page has been shown – the Basket has been cleared – the order may stay/hang in this status if the process did not complete properly (bug) or the user abandons the Iframe or Hosted page and does not complete the payment
  + NEW – the Iframe or Hosted page has been closed with an accept – the Order changed it’s status
  + OPEN – the Order has been viewed in DW Business Manager after it was in status NEW
  + FAILED – all cases of cancel and decline

| **UC - 1** | **HiPay Hosted Page** |
| --- | --- |
| This use case describes the high level steps in which a registered/guest customer successfully creates an order and the order with HiPay Hosted Page. | Preconditions   * Go to Merchant Tools > Site Preferences > Custom Preferences > HiPay Settings * For HiPay Operation Mode, select hosted (Hosted Page) and click on Apply  1. A registered customer navigates to the site, an item, add it to the cart and proceeds to the cart page. 2. The customer clicks on the **Checkout** button and fills in the shipping form requirements. 3. The customer clicks on the **Continue** button and proceeds to fill in the billing form requirements and for payment method, select “**HiPay Hosted**”. 4. The customer clicks on the **Continue** button, proceeds to the **Payment** page and clicks on the **Place order** button. 5. A HiPay hosted page is opened and the customer enters the card details. 6. After successful payment the customer gets redirected to the Summary page and the thank you message is successfully loaded. 7. The customer can verify Order Summary information 8. Merchant can verify Order status in DW BM 9. Merchant can verify information in HIPay Fullservice account   **Note that a similar flow can be done for guest checkout** |

| **UC - 2** | **HiPay Hosted Page with 3-D Secure** |
| --- | --- |
| This use case describes the high level steps in which a registered/guest customer successfully creates an order and the order with HiPay Hosted Page with 3-D Secure. | Preconditions   * Go to Merchant Tools > Site Preferences > Custom Preferences > HiPay Settings * For HiPay Operation Mode, select hosted (Hosted Page) and click on Apply * For 3-D Secure, select “1 (3-D Secure authentication if available)” and click on “Apply”  1. A registered customer navigates to the site, an item, add it to the cart and proceeds to the cart page. 2. The customer clicks on the **Checkout** button and fills in the shipping form requirements. 3. The customer clicks on the **Continue** button and proceeds to fill in the billing form requirements and for payment method, select “**HiPay Hosted**”. 4. The customer clicks on the **Continue** button, proceeds to the **Payment** page and clicks on the **Place order** button. 5. A HiPay hosted page is opened and the customer enters the 3-D Secure card details. 6. The customer is redirected to the card vendor’s site and it enters the 3-D Secure password. 7. After successful identification the customer clicks on “Back to Payment”. 8. After successful payment the customer gets redirected to the Summary page and the thank you page is successfully loaded. 9. The customer can verify Order Summary information 10. Merchant can verify Order status in DW BM 11. Merchant can verify information in HIPay Fullservice account   **Note that a similar flow can be done for guest checkout** |

| **UC - 3** | **HiPay API Integration** |
| --- | --- |
| This use case describes the high level steps in which a registered/guest customer successfully creates an order and the order with HiPay API Integration. | Preconditions   * Go to Merchant Tools > Site Preferences > Custom Preferences > HiPay Settings * For HiPay Operation Mode, select api (API) and click on Apply  1. A registered customer navigates to the site, an item, add it to the cart and proceeds to the cart page. 2. The customer clicks on the **Checkout** button and fills in the shipping form requirements. 3. The customer clicks on the **Continue** button and proceeds to fill in the billing form requirements and for payment method, select “iDEAL” 4. The customer selects the desired issuing bank and clicks on the **Place order** button. 5. The customer is redirected to the iDEAL vendor site. 6. The customer enters the payment details. 7. After successful payment the customer gets redirected to the Summary page and the thank you message is successfully loaded. 8. The customer can verify Order Summary information 9. Merchant can verify Order status in DW BM – in Order screen, open Payment tab to verify the payment information. 10. Merchant can verify information in HIPay Fullservice account   **Note that a similar flow can be done for guest checkout** |

## Limitations, Constraints

* The merchant needs a configured HiPay account
* The merchant can style the hosted pages to make it matching the look and feel of their store.
* The following features are out of scope:
  + Cancelled Order support
  + Refund support
  + One-Click Payment (credit card only)

## Compatibility

The HiPay integration cartridge is compatible with the latest Salesforce Commerce Cloud API version, currently 17.8 and SG version 103.1.11

## Privacy, Payment

When the redirect method is used, all payment data is entered into the HiPay hosted pages by the customer and no Credit Card data is stored in Salesforce Commerce Cloud (other than the brand of the used card).

Examples:

HiPay Product Name: hosted

HiPay Payment Product List: bcmc,cb,maestro,mastercard,visa,american-express

HiPay Payment Category List: credit-card,debit-card

HiPay Product Name: dexia-directnet

HiPay Payment Product List: dexia-directnet

HiPay Payment Category List: realtime-banking

# Implementation Guide

## Integration Efforts

The following steps are needed to complete the integration:

* Install the cartridge
* Change and import ‘site-template.zip’ (system objects)
* Do the required code changes
* Configure your merchant's HiPay Account in the HiPay back office
* Configure HiPay parameters in Salesforce Commerce Cloud BM
* Test

## Setup

### Installation

Install the “int\_hipay\_controllers”, “bm\_hipay\_controllers” cartridges from the distributive zip-archive in the standard way using the Salesforce Commerce Cloud UX-studio.

### Metadata Import

Find the folder ‘site-template’ in the installation package, which will contain metadata files.

Please review the ‘site\_template’ folder, do the necessary modifications, archive the ‘site\_template’ folder (into the site\_template.zip) and import it through BM Administration > Site Development > Site Import & Export section. The only required modification is the renaming of the “SiteGenesis” folder (under the site\_template/sites folder) to the ID of your site. For multiple sites integration, you will also need to clone the SiteGenesis folder and rename it with the ID of your site. After import you will also need to check the ClearHungOrder schedule and Site Assignment section, see [5.2.4 Schedules.](#_Schedules)

Please note that the ‘site\_template’ folder also contains the ‘pin-price-lists.xml’ with pricebook. If you don’t need it, you may just remove this file and the ‘pricebooks’ folder before archiving the ‘site\_template’ folder.

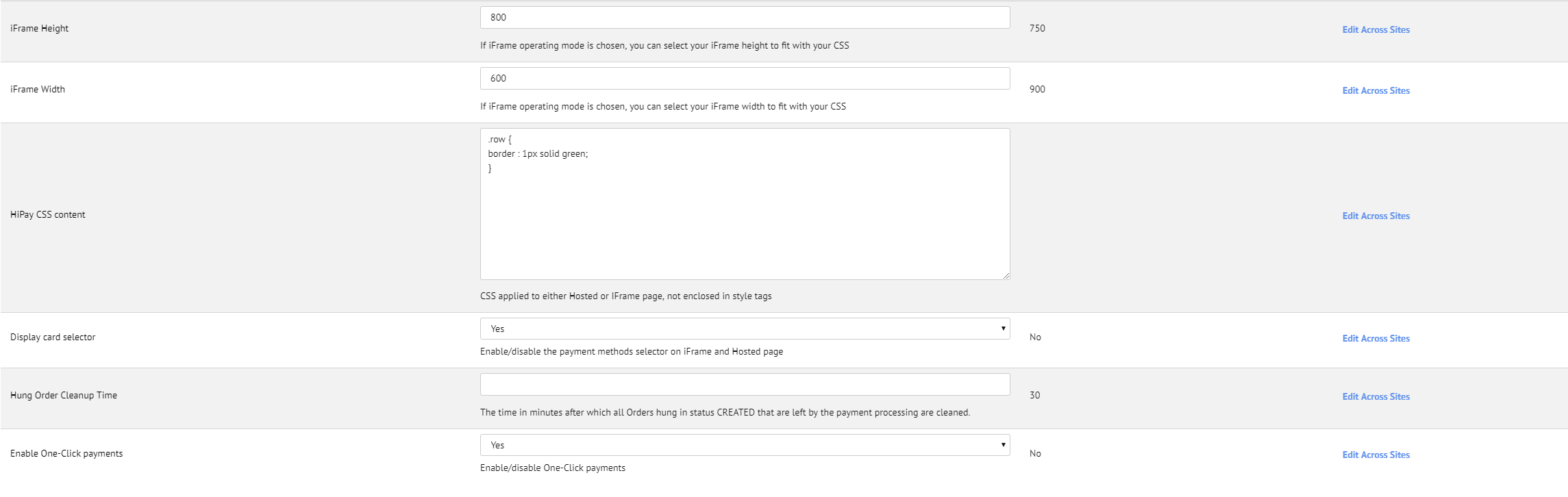
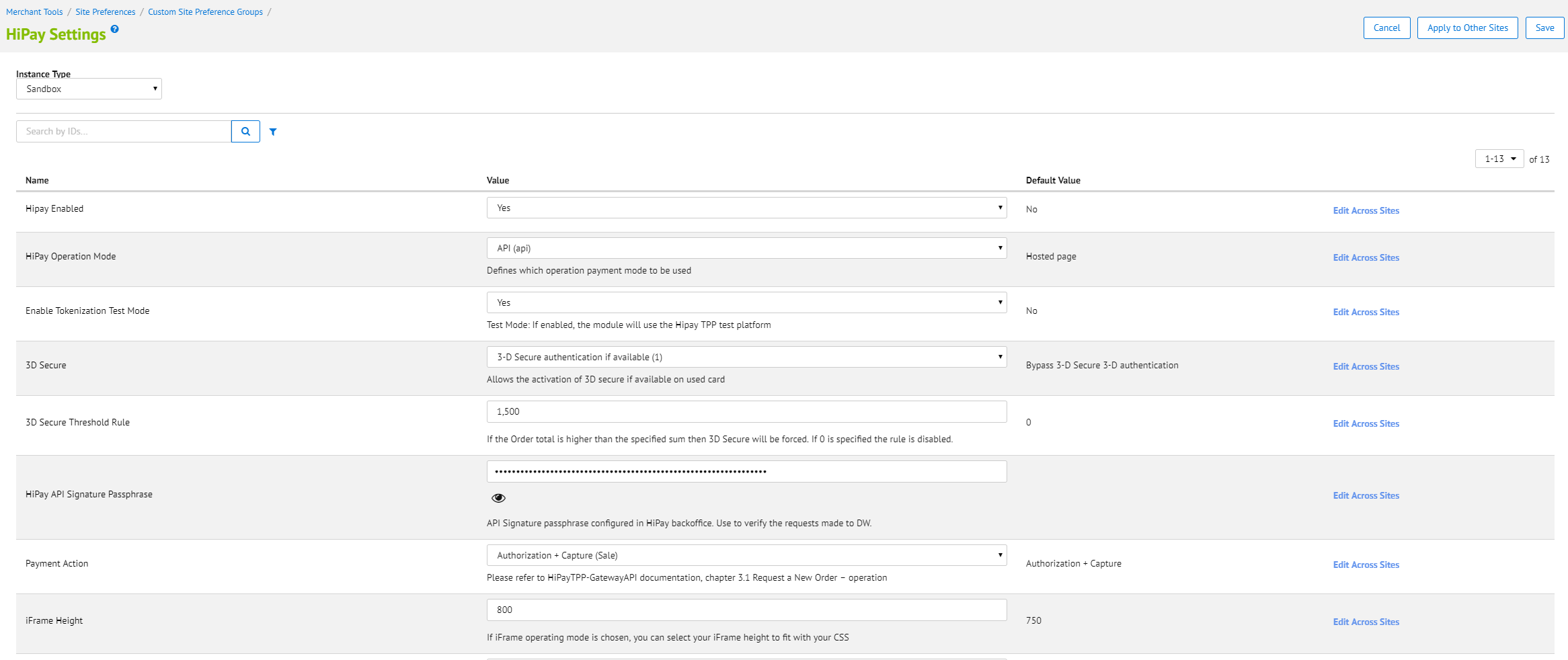
Before importing the ‘site-template.zip’, check and save the cartridge paths of your site and Business Manager, because these will be modified. By saving them elsewhere you will be able to configure them manually if something goes wrong. Also, check the fields which will be updated after importing the file ‘site-template.zip’, and be sure that there will be no conflict with any existing fields.

## Configuration

* Add an int\_hipay\_controllers to the cartridge site path:
  + For storefront controller version:  
    int\_hipay\_controllers:app\_storefront\_controllers:app\_storefront\_core
  + For storefront pipeline version:

int\_hipay\_controllers:app\_storefront\_pipelines:app\_storefront\_core

* Add a bm\_hipay\_controllers to the cartridge BM path:   
  bm\_hipay\_controllers:int\_hipay\_controllers
* Go to BM [**Administration**](https://osfglobal09.alliance-prtnr-na02.dw.demandware.net/on/demandware.store/Sites-Site/default/ViewApplication-ExpandMenuGroup?MenuGroupID=AdministrationMenu&OverviewPage=SiteNavigationBar-ShowAdministrationOverview) > [**Site Development**](https://osfglobal09.alliance-prtnr-na02.dw.demandware.net/on/demandware.store/Sites-Site/default/SiteNavigationBar-ShowMenuitemOverview?CurrentMenuItemId=studio) > Site Import & Export:  
  and import the site\_template.zip (make a zip file)
* Custom site preferences (HiPay preferences):  
  [**Merchant Tools**](https://osfglobal09.alliance-prtnr-na02.dw.demandware.net/on/demandware.store/Sites-Site/default/ViewApplication-SelectSite?MenuGroupID=ChannelMenu&ChannelID=bc2EUiaagRjE6aaadl7Bold27B&SelectedSiteID=bc2EUiaagRjE6aaadl7Bold27B) > [**Site Preferences**](https://osfglobal09.alliance-prtnr-na02.dw.demandware.net/on/demandware.store/Sites-Site/default/SiteNavigationBar-ShowMenuitemOverview?CurrentMenuItemId=site-prefs) > [**Custom Site Preferences**](https://osfglobal09.alliance-prtnr-na02.dw.demandware.net/on/demandware.store/Sites-Site/default/CustomPreferences-View?PreferenceType=SITE) > HiPay Settings:
  + **HiPay API Signature Passphrase -** API Signature passphrase configured in HiPay backoffice. Use to verify the requests made to DW.
  + **HiPay CSS content -** CSS applied to either Hosted or IFrame page, not enclosed in style tags.
  + **Display card selector -** Enable/disable the payment methods selector on iFrame and Hosted page.
  + **3D Secure -** Allows the activation of 3D secure if available on used card.
  + **3D Secure Threshold Rule -** If the Order total is higher than the specified sum then 3D Secure will be forced. If 0 is specified, the rule is disabled.
  + **Enable Tokenization Test Mode -** Test Mode: If enabled, the module will use the HiPay TPP test platform.
  + **Hipay Enabled –** Indicates whether the HiPay code will be exacuted or not.
  + **Enable One-Click payments –** Enable/disable One-Click payments.
  + **Hung Order Cleanup Time -** The time in minutes after which all Orders hung in status CREATED that are left by the payment processing are cleaned.
  + **iFrame Height -** If iFrame operating mode is chosen, you can select your iFrame height to fit with your CSS.
  + **iFrame Width -** If iFrame operating mode is chosen, you can select your iFrame width to fit with your CSS.
  + **HiPay Operation Mode -** defines which operation payment mode to be used – these are HOSTED/IFRAME/API. When selecting the API mode then the HIPAY\_HOSTED payment method will be hidden from Billing page. If HOSTED or IFRAME is selected, then all API payment methods are hidden from the Billing page.
  + **Payment Action -** Please refer to HiPayTPP-GatewayAPI documentation, chapter 3.1 Request a New Order – operation



* Service configuration:  
  [**Administration**](https://osfglobal09.alliance-prtnr-na02.dw.demandware.net/on/demandware.store/Sites-Site/default/ViewApplication-ExpandMenuGroup?MenuGroupID=AdministrationMenu&OverviewPage=SiteNavigationBar-ShowAdministrationOverview) **>** [**Operations**](https://osfglobal09.alliance-prtnr-na02.dw.demandware.net/on/demandware.store/Sites-Site/default/SiteNavigationBar-ShowMenuitemOverview?CurrentMenuItemId=operations) **> Services > hipay.{type}.cred Details**  
  + Update User ID and password for hipay credentials: hipay.hosted.cred, hipay.maintence.cred, hipay.order.cred, hipay.token.cred. User ID and password you can take from hipay dashboard intergration.  
    - URL: <https://stage-secure-gateway.hipay-tpp.com>- User: enter user name for example (\*\*\*\*\*\*\*\*.api.hipay-tpp.com)  
    - Password: enter your password
  + For multiple hipay accounts configuration read [HiPay Multi-account](#_HiPay_Multi-account).
    - HiPay payment methods can be enabled/disabled from the **Payment methods** section. There are two methods – **HiPay Hosted** and **HiPay**, responsible for hosted solution and for the integrated API respectively.
    - HiPay predefined Log category

Go to: **Administration > Operations > Custom Log Settings**

The HiPay log category is called „HIPAY”. It can be added or removed from the **Info** and **Debug** configuration.

## Custom Code

For all custom code changes an example is included in int\_hipay\_controllers and each is also described in the current section:

1. **app\_storefront\_core/cartridge/templates/default/checkout/billing/creditcardjson.isml** (see the int\_hipay\_controllers/cartridge/templates/default/checkout/billing/creditcardjson\_sample.isml as an example)

Add:

<isset name="hipayEnabled" value="${dw.system.Site.getCurrent().getCustomPreferenceValue('hipayEnabled')}" scope="page"/>

<isif condition="${hipayEnabled}" >

<iselse/>

</isif>

After:

<isinclude template="util/jsonmodule"/>

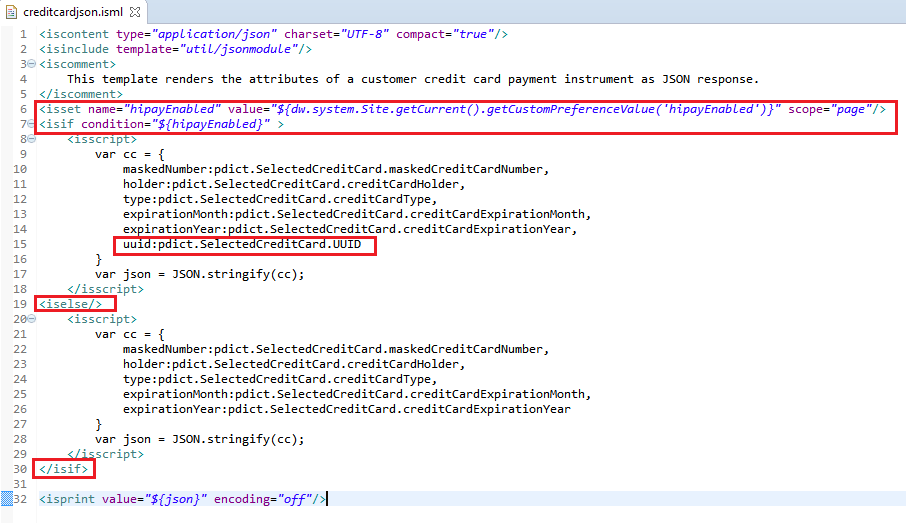
Put an existing <isscript>...</isscript> section between <iselse/></isif> lines and between <isif condition="${hipayEnabled}" ><iselse/> lines.

In the first if-else condition block add:

uuid:pdict.SelectedCreditCard.UUID

After:

expirationYear:pdict.SelectedCreditCard.creditCardExpirationYear



1. **app\_storefront\_core/cartridge/templates/default/checkout/billing/paymentmethods.isml** (see the int\_hipay\_controllers/cartridge/templates/default/checkout/billing/paymentmethods\_sample.isml as an example)

Add:

<isset name="hipayEnabled" value="${dw.system.Site.getCurrent().getCustomPreferenceValue('hipayEnabled')}" scope="page"/>

<isset name="hipayEnableOneClick" value="${dw.system.Site.getCurrent().getCustomPreferenceValue('hipayEnableOneClick')}" scope="page"/>

After:

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

Add:

<isif condition="${hipayEnabled}" >

<link rel="stylesheet" href="${URLUtils.staticURL('/css/hipay.css')}" />

<isinclude template="checkout/components/devicefingerprint"/>

<isscript>

var HiPayConfig = require('int\_hipay\_controllers/cartridge/scripts/lib/hipay/HiPayConfig').HiPayConfig;

var isHiPayHostedMode = false;

if(HiPayConfig.hipayOperationMode == HiPayConfig.OPERATION.HOSTED || HiPayConfig.hipayOperationMode == HiPayConfig.OPERATION.IFRAME) {

isHiPayHostedMode = true;

}

</isscript>

<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/></isif>

<!--- HIPAY INTEGRATION ------ skip all non-HiPay methods if HiPay is enabled--->

<isif condition="${(paymentMethodType.value.indexOf('HIPAY') == -1)}" ><iscontinue/></isif>

<isif condition="${(paymentMethodType.value.indexOf('HIPAY') > -1)}">

<iscomment>Skip HIPAY\_ payment methods if is in HOSTED operation mode.</iscomment>

<isif condition="${isHiPayHostedMode && paymentMethodType.value.indexOf('HIPAY\_HOSTED') == -1}"><iscontinue/></isif>

<iscomment>Skip hosted page payment method if is in API operation mode.</iscomment>

<isif condition="${!isHiPayHostedMode && paymentMethodType.value.indexOf('HIPAY\_HOSTED') > - 1}"> <iscontinue/></isif>

</isif>

<div class="form-row label-inline">

<isset name="radioID" value="${paymentMethodType.value}" scope="page"/>

<label for="is-${radioID}"> <isprint value="${Resource.msg(paymentMethodType.label,'forms',null)}"/>

<isscript>

var paymentMethod = dw.order.PaymentMgr.getPaymentMethod(paymentMethodType.value);

var hipayProductName = paymentMethod.custom.hipayProductName; if(!empty(hipayProductName)) { hipayProductName = hipayProductName.toLowerCase(); }

</isscript>

<isif condition="${!empty(hipayProductName)}">

<i class="payment-product-sprite-${hipayProductName}"></i>

</isif>

</label>

<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>

</div>

</isloop>

</div>

<div class="form-row form-row-button">

<button id="payment-select-go" name="${pdict.CurrentForms.billing.paymentSelect.htmlName}" type="submit" value="Go" class="simple-submit">Select</button>

</div>

<isif condition="${!empty(pdict.MissingPaymentProcessor)}">

<div class="billing-error">${Resource.msg('billing.missingprocessorerror','checkout',null)}</div>

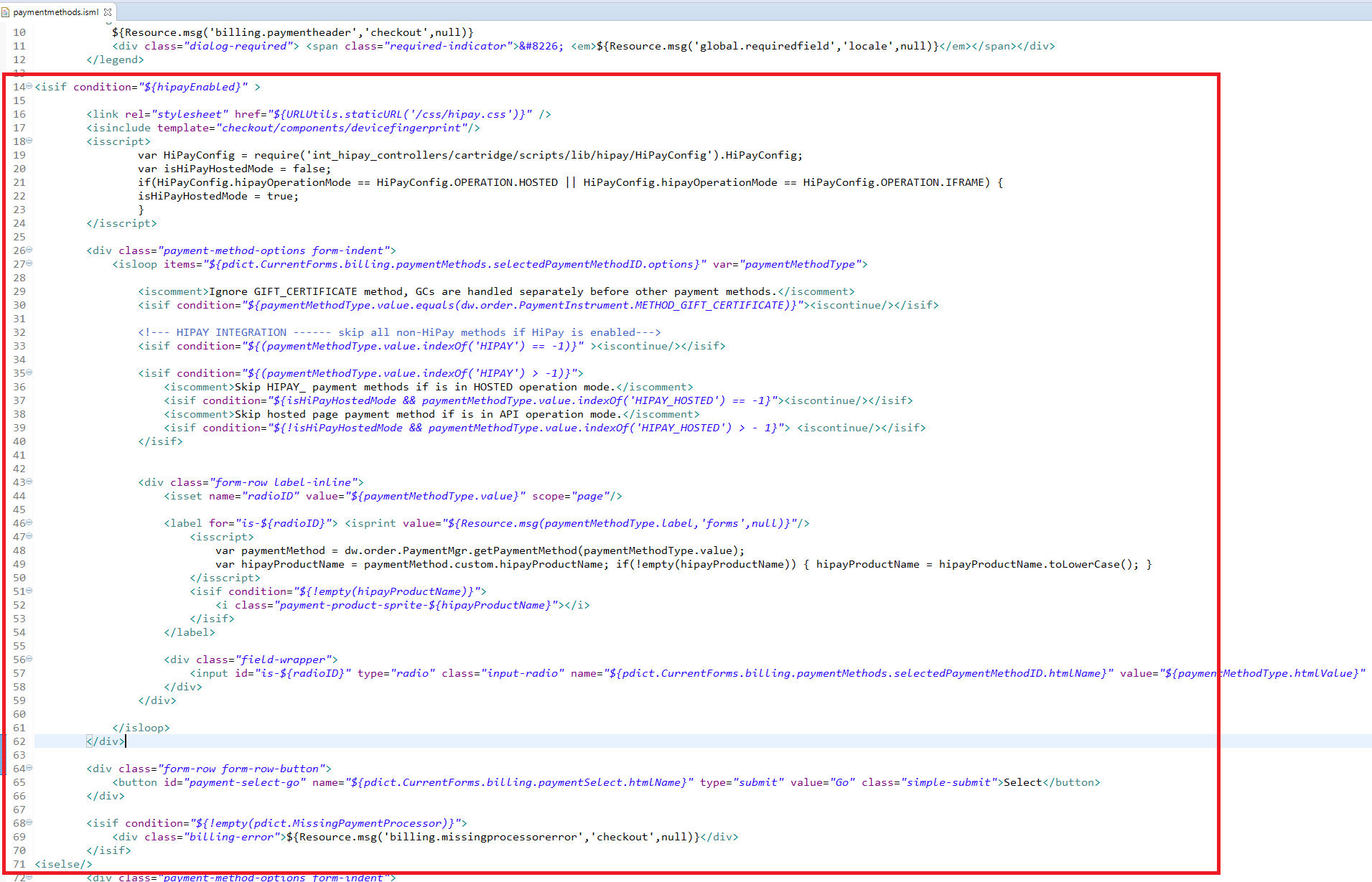
</isif>

<iselse/>

After:

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

</div>  
</legend>



Add </isif> after an existing code block for payment methods rendering.

Also add next lines:

<!--- HIPAY INTEGRATION ------ skip all HiPay methods if HiPay is disabled--->

<isif condition="${paymentMethodType.value.indexOf('HIPAY') > -1}" ><iscontinue/></isif>



Add <isif condition="${!hipayEnabled}" > before:

<iscomment>

Credit card block

</iscomment>



Add: </isif> after:

<iscomment>

Custom processor

--------------------------------------------------------------

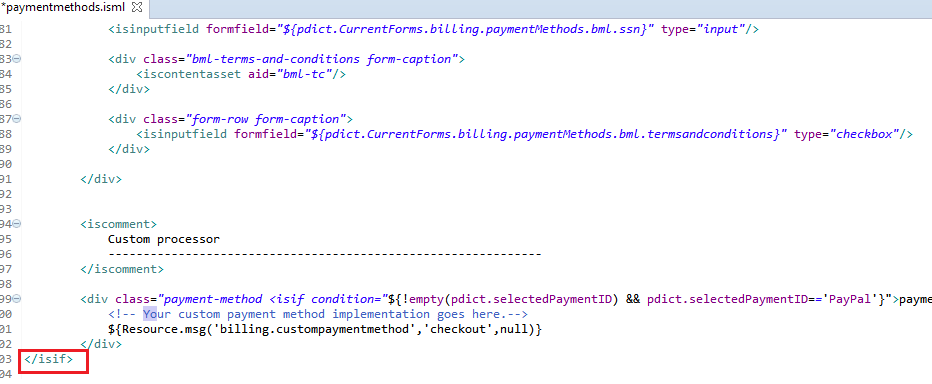
</iscomment>

<div class="payment-method <isif condition="${!empty(pdict.selectedPaymentID) && pdict.selectedPaymentID=='PayPal'}">payment-method-expanded</isif>" data-method="Custom">

<!-- Your custom payment method implementation goes here.-->

${Resource.msg('billing.custompaymentmethod','checkout',null)}

</div>



Add:

<isif condition="${hipayEnabled}" >

<!--- HIPAY INTEGRATION ------ BEGIN --->

<iscomment>

HIPAY Credit card

--------------------------------------------------------------

</iscomment>

<div class="payment-method <isif condition="${(empty(pdict.selectedPaymentID) || pdict.selectedPaymentID=='HIPAY\_CREDIT\_CARD') && isHiPayHostedMode == false }">payment-method-expanded</isif>" data-method="HIPAY\_CREDIT\_CARD">

<iscomment>display select box with stored credit cards if customer is authenticated</iscomment>

<isif condition="${pdict.CurrentCustomer.authenticated && !empty(pdict.ApplicableCreditCards) && hipayEnableOneClick}">

<div class="form-row">

<label class="label">${Resource.msg('billing.selectcreditcard','checkout',null)}</label>

<div class="field-wrapper">

<select name="${pdict.CurrentForms.billing.paymentMethods.creditCardList.htmlName}" id="creditCardList" class="input-select">

<option value="">${Resource.msg('billing.creditcardlistselect','checkout',null)}</option>

<isloop items="${pdict.ApplicableCreditCards}" var="creditCardInstr">

<option value="${creditCardInstr.UUID}">(<isprint value="${creditCardInstr.creditCardType}"/>) <isprint value="${creditCardInstr.maskedCreditCardNumber}"/> - ${Resource.msg('billing.creditcardlistexp','checkout',null)} <isprint value="${creditCardInstr.creditCardExpirationMonth}" formatter="00" />.<isprint value="${creditCardInstr.creditCardExpirationYear}" formatter="0000" /></option>

</isloop>

</select>

</div>

</div>

<div class="form-row form-row-button">

<button id="credit-card-select-go" name="${pdict.CurrentForms.billing.creditCardSelect.htmlName}" type="submit" value="Go" class="simple-submit">Select</button>

</div>

<iscomment>

<isloop items="${pdict.ApplicableCreditCards}" var="creditCardInstr">

<a href="${URLUtils.https('COBilling-UpdateCreditCardSelection', 'creditCardUUID', creditCardInstr.UUID)}">

(<isprint value="${creditCardInstr.creditCardType}"/>)

<isprint value="${creditCardInstr.maskedCreditCardNumber}"/>

- ${Resource.msg('billing.creditcardlistexp','checkout',null)}

<isprint value="${creditCardInstr.creditCardExpirationMonth}" formatter="00" />

.<isprint value="${creditCardInstr.creditCardExpirationYear}" formatter="0000" />

</a>

</isloop>

</iscomment>

</isif>

<div class="ccfields-wrap">

<isset name="selectedCreditCardType" value="${ pdict.CurrentForms.billing.paymentMethods.creditCard.type.selectedOptionObject ? pdict.CurrentForms.billing.paymentMethods.creditCard.type.selectedOptionObject.cardType : null}" scope="page" />

<div class="owner-wrap">

<div class="owner-split ${selectedCreditCardType != 'Amex' ? 'hide' : ''}">

<isinputfield formfield="${pdict.CurrentForms.billing.paymentMethods.creditCard.ownerfirst}" type="input" attribute1="autocomplete" value1="off" xhtmlclass="owner"/>

<isinputfield formfield="${pdict.CurrentForms.billing.paymentMethods.creditCard.ownerlast}" type="input" attribute1="autocomplete" value1="off" xhtmlclass="owner"/>

</div>

<div class="owner-single ${selectedCreditCardType == 'Amex' ? 'hide' : ''}">

<isinputfield formfield="${pdict.CurrentForms.billing.paymentMethods.creditCard.owner}" type="input" attribute1="autocomplete" value1="off" xhtmlclass="owner"/>

</div>

</div>

<isinputfield formfield="${pdict.CurrentForms.billing.paymentMethods.creditCard.type}" type="select"/>

<isinputfield formfield="${pdict.CurrentForms.billing.paymentMethods.creditCard.number}" type="input" attribute1="autocomplete" value1="off"/>

<isinputfield formfield="${pdict.CurrentForms.billing.paymentMethods.creditCard.uuid}" type="hidden" attribute1="autocomplete" value1="off"/>

<div class="form-label-text"><span class="required-indicator">${Resource.msg('billing.requiredindicator','checkout',null)}</span>${Resource.msg('billing.creditcardlistexpdate', 'checkout', null)}</div>

<isif condition="${pdict.CurrentForms.billing.paymentMethods.creditCard.expiration.month.selectedOption === null || pdict.CurrentForms.billing.paymentMethods.creditCard.expiration.year.selectedOption === null}">

<isscript>

var currentYear = (new Date()).getYear(),

currentMonth = (new Date()).getMonth(),

year = pdict.CurrentForms.billing.paymentMethods.creditCard.expiration.year,

month = pdict.CurrentForms.billing.paymentMethods.creditCard.expiration.month;

currentMonth += 1;

year.setHtmlValue(currentYear);

month.setHtmlValue(currentMonth);

</isscript>

</isif>

<isinputfield formfield="${pdict.CurrentForms.billing.paymentMethods.creditCard.expiration.month}" type="select" rowclass="month"/>

<isinputfield formfield="${pdict.CurrentForms.billing.paymentMethods.creditCard.expiration.year}" type="select" rowclass="year"/>

<div class="cvn-wrap ${selectedCreditCardType === 'BancontactMisterCash' ? 'hide' : ''}">

<isinputfield formfield="${pdict.CurrentForms.billing.paymentMethods.creditCard.cvn}" type="input" rowclass="cvn" attribute1="autocomplete" value1="off" helpcid="checkout-security-code" helplabel="${Resource.msg('billing.linkcvn', 'checkout', null)}"/>

</div>

<isif condition="${pdict.CurrentCustomer.authenticated && hipayEnableOneClick}">

<isinputfield formfield="${pdict.CurrentForms.billing.paymentMethods.creditCard.saveCard}" type="checkbox" rowclass="label-inline form-indent save-card"/>

</isif>

</div>

</div>

<iscomment>Belfius Direct Net</iscomment>

<div class="payment-method <isif condition="${!empty(pdict.selectedPaymentID) && pdict.selectedPaymentID=='HIPAY\_DEXIA\_DIRECTNET'}">payment-method-expanded</isif>" data-method="HIPAY\_DEXIA\_DIRECTNET">

<!--- No form is applied --->

</div>

<iscomment>Giropay</iscomment>

<div class="payment-method <isif condition="${!empty(pdict.selectedPaymentID) && pdict.selectedPaymentID=='HIPAY\_GIROPAY'}">payment-method-expanded</isif>" data-method="HIPAY\_GIROPAY">

<isinputfield formfield="${pdict.CurrentForms.billing.paymentMethods.hipaymethods.username}" type="input" />

<isinputfield formfield="${pdict.CurrentForms.billing.paymentMethods.hipaymethods.password}" type="input" />

</div>

<iscomment>Hosted Pages</iscomment>

<div class="payment-method <isif condition="${!empty(pdict.selectedPaymentID) && pdict.selectedPaymentID=='HIPAY\_HOSTED'}">payment-method-expanded</isif>" data-method="HIPAY\_HOSTED">

<!--- No form is applied --->

</div>

<iscomment>iDEAL</iscomment>

<div class="payment-method <isif condition="${!empty(pdict.selectedPaymentID) && pdict.selectedPaymentID=='HIPAY\_IDEAL'}">payment-method-expanded</isif>" data-method="HIPAY\_IDEAL">

<isinputfield formfield="${pdict.CurrentForms.billing.paymentMethods.hipaymethods.issuer\_bank\_id}" type="select" />

</div>

<iscomment>ING HomePay</iscomment>

<div class="payment-method <isif condition="${!empty(pdict.selectedPaymentID) && pdict.selectedPaymentID=='HIPAY\_ING\_HOMEPAY'}">payment-method-expanded</isif>" data-method="HIPAY\_ING\_HOMEPAY">

<!--- No form is applied --->

</div>

<iscomment>Klarna</iscomment>

<div class="payment-method <isif condition="${!empty(pdict.selectedPaymentID) && pdict.selectedPaymentID=='HIPAY\_KLARNA'}">payment-method-expanded</isif>" data-method="HIPAY\_KLARNA">

<isinputfield formfield="${pdict.CurrentForms.billing.paymentMethods.hipaymethods.username}" type="input" />

<isinputfield formfield="${pdict.CurrentForms.billing.paymentMethods.hipaymethods.password}" type="input" />

</div>

<iscomment>Przelewy24</iscomment>

<div class="payment-method <isif condition="${!empty(pdict.selectedPaymentID) && pdict.selectedPaymentID=='HIPAY\_PRZELEWY24'}">payment-method-expanded</isif>" data-method="HIPAY\_PRZELEWY24">

<!--- No form is applied --->

</div>

<iscomment>QIWI Wallet</iscomment>

<div class="payment-method <isif condition="${!empty(pdict.selectedPaymentID) && pdict.selectedPaymentID=='HIPAY\_QIWI\_WALLET'}">payment-method-expanded</isif>" data-method="HIPAY\_QIWI\_WALLET">

<!--- No form is applied --->

</div>

<iscomment>Sisal</iscomment>

<div class="payment-method <isif condition="${!empty(pdict.selectedPaymentID) && pdict.selectedPaymentID=='HIPAY\_SISAL'}">payment-method-expanded</isif>" data-method="HIPAY\_SISAL">

<!--- No form is applied --->

</div>

<iscomment>Sofort Uberweisung</iscomment>

<div class="payment-method <isif condition="${!empty(pdict.selectedPaymentID) && pdict.selectedPaymentID=='HIPAY\_SOFORT\_UBERWEISUNG'}">payment-method-expanded</isif>" data-method="HIPAY\_SOFORT\_UBERWEISUNG">

<!--- No form is applied --->

</div>

<iscomment>WebMoney Transfer</iscomment>

<div class="payment-method <isif condition="${!empty(pdict.selectedPaymentID) && pdict.selectedPaymentID=='HIPAY\_WEBMONEY\_TRANSFER'}">payment-method-expanded</isif>" data-method="HIPAY\_WEBMONEY\_TRANSFER">

<!--- No form is applied --->

</div>

<iscomment>Yandex Money</iscomment>

<div class="payment-method <isif condition="${!empty(pdict.selectedPaymentID) && pdict.selectedPaymentID=='HIPAY\_YANDEX'}">payment-method-expanded</isif>" data-method="HIPAY\_YANDEX">

<isinputfield formfield="${pdict.CurrentForms.billing.paymentMethods.hipaymethods.username}" type="input" />

<isinputfield formfield="${pdict.CurrentForms.billing.paymentMethods.hipaymethods.password}" type="input" />

</div>

<iscomment>PayPal</iscomment>

<div class="payment-method <isif condition="${!empty(pdict.selectedPaymentID) && pdict.selectedPaymentID=='PayPal'}">payment-method-expanded</isif>" data-method="PayPal">

<!--- No form is applied --->

</div>

<!--- HIPAY INTEGRATION ------ END --->

</isif>

After:

<div class="payment-method <isif condition="${!empty(pdict.selectedPaymentID) && pdict.selectedPaymentID=='PayPal'}">payment-method-expanded</isif>" data-method="Custom">  
 <!--- Your custom payment method implementation goes here. --->

${Resource.msg('billing.custompaymentmethod','checkout',null)}  
</div>

</isif>



1. **Create a file devicefingerprint.isml in app\_storefront\_core/cartridge/templates/default/checkout/components/devicefingerprint.isml** (see the int\_hipay\_controllers/cartridge/templates/default/checkout/components/devicefingerprint\_sample.isml as an example)

<!-- hidden field to store blackbox -->  
<input type="hidden" name="${pdict.CurrentForms.billing.paymentMethods.deviceFingerprint.htmlName}" id="ioBB">  
<!-- Include HiPay JavaScript fingerprint library -->  
<script language="javascript" src="https://secure-gateway.hipay-tpp.com/gateway/toolbox/fingerprint">  
</script>

1. **app\_storefront\_core/cartridge/templates/default/checkout/summary/summary.isml** (see the int\_hipay\_controllers/cartridge/templates/default/checkout/summary/summary\_sample.isml as an example)

Add:

<isset name="hipayEnabled" value="${dw.system.Site.getCurrent().getCustomPreferenceValue('hipayEnabled')}" scope="page"/>

<isif condition="${hipayEnabled}" >

<iselse/>

</isif>

Before:

<isif condition="${pdict.PlaceOrderError != null}">

<div class="error-form">

${Resource.msg(pdict.PlaceOrderError.code,'checkout',null)}

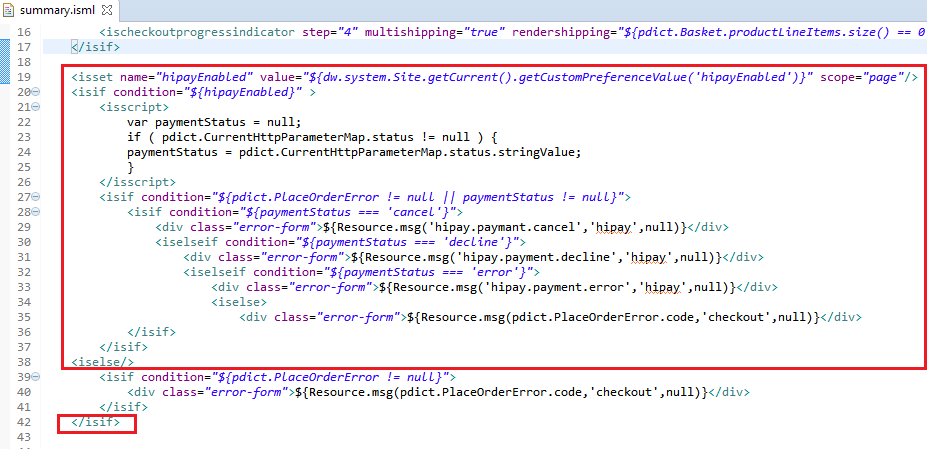
</div>

</isif>

Put an existing <isif condition="${pdict.PlaceOrderError != null}">...</isif> section between just added <iselse/></isif> lines.

In the first if-else condition block add:

<isscript>  
var paymentStatus = null;  
if ( pdict.CurrentHttpParameterMap.status != null ) {  
 paymentStatus = pdict.CurrentHttpParameterMap.status.stringValue;  
}  
</isscript>  
<isif condition="${pdict.PlaceOrderError != null || paymentStatus != null}">  
 <isif condition="${paymentStatus === 'cancel'}">  
 <div class="error-form">${Resource.msg('hipay.paymant.cancel','hipay',null)}</div>  
 <iselseif condition="${paymentStatus === 'decline'}">   
 <div class="error-form">${Resource.msg('hipay.payment.decline','hipay',null)}</div>  
 <iselseif condition="${paymentStatus === 'error'}">   
 <div class="error-form">${Resource.msg('hipay.payment.error','hipay',null)}</div>  
 <iselse>   
 <div class="error-form">${Resource.msg(pdict.PlaceOrderError.code,'checkout',null)}</div>  
 </isif>  
</isif>



1. **app\_storefront\_core/cartridge/js/pages/checkout/billing.js**

Replace:

**var** $creditCard = $('[data-method="CREDIT\_CARD"]');

With:

if (SitePreferences.HIPAY\_ENABLED) {

var $creditCard = $('[data-method="HIPAY\_CREDIT\_CARD"]');

$creditCard.find('input[name$="uuid"]').val(data.uuid).trigger('change');

} else {

var $creditCard = $('[data-method="CREDIT\_CARD"]');

}



Create clearCCFields function after setCCFields function:

/\*\*

\* @function

\* @description Clears the Credit Card form and show fields

\*/

function clearCCFields() {

var $creditCard = $('[data-method="HIPAY\_CREDIT\_CARD"]');

$creditCard.find('input[name$="creditCard\_owner"]').val('').trigger('change');

$creditCard.find('input[name$="creditCard\_uuid"]').val('').trigger('change');

$creditCard.find('select[name$="\_type"]').val('').trigger('change');

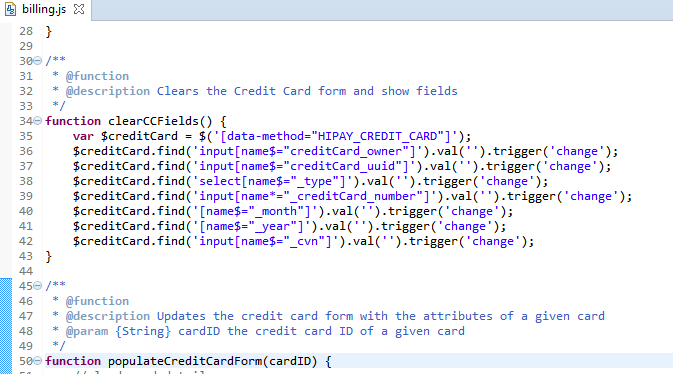
$creditCard.find('input[name\*="\_creditCard\_number"]').val('').trigger('change');

$creditCard.find('[name$="\_month"]').val('').trigger('change');

$creditCard.find('[name$="\_year"]').val('').trigger('change');

$creditCard.find('input[name$="\_cvn"]').val('').trigger('change');

}



Add:

if (SitePreferences.HIPAY\_ENABLED && SitePreferences.HIPAY\_ENABLE\_ONECLICK) {

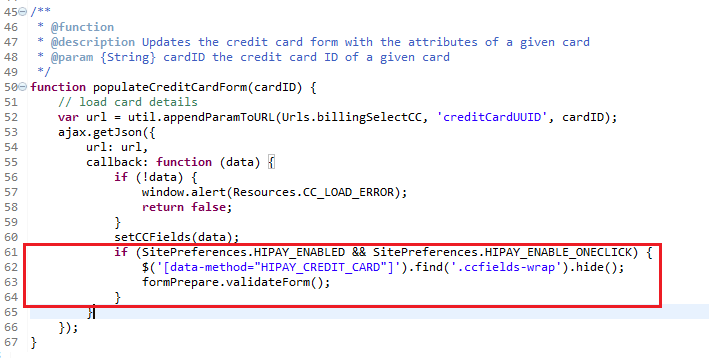
$('[data-method="HIPAY\_CREDIT\_CARD"]').find('.ccfields-wrap').hide();

formPrepare.validateForm();

}

After:

setCCFields(data);



Replace:

// default payment method to 'CREDIT\_CARD'

updatePaymentMethod((selectedPaymentMethod) ? selectedPaymentMethod : 'CREDIT\_CARD');

$selectPaymentMethod.on('click', 'input[type="radio"]', function () {

updatePaymentMethod($(this).val());

});

With:

if (SitePreferences.HIPAY\_ENABLED) {

// select payment method if available

if(selectedPaymentMethod){

updatePaymentMethod(selectedPaymentMethod);

}

$selectPaymentMethod.on('click', 'label', function () {

var radio = $(this).parent().find('input[type="radio"]');

radio.prop('checked', true);

updatePaymentMethod($(radio).val());

});

} else {

// default payment method to 'CREDIT\_CARD'

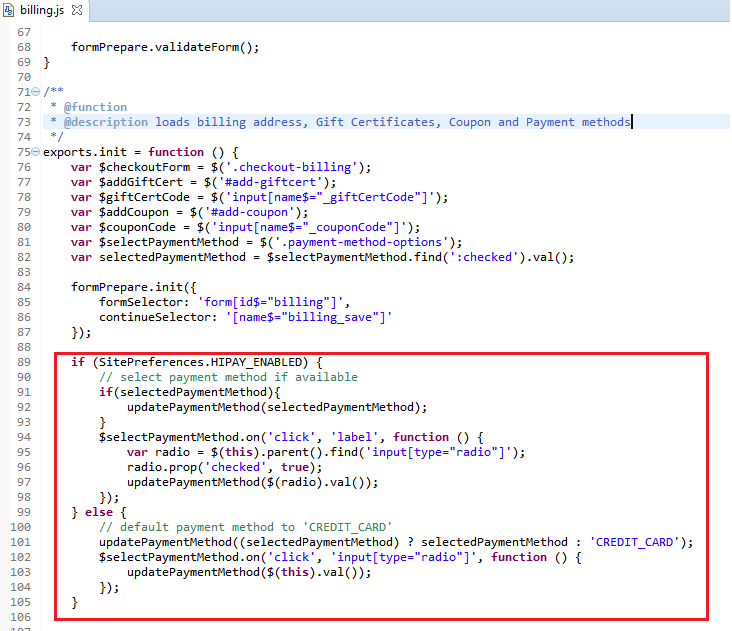
updatePaymentMethod((selectedPaymentMethod) ? selectedPaymentMethod : 'CREDIT\_CARD');

$selectPaymentMethod.on('click', 'input[type="radio"]', function () {

updatePaymentMethod($(this).val());

});

}



Add:

if (SitePreferences.HIPAY\_ENABLED && SitePreferences.HIPAY\_ENABLE\_ONECLICK) {

clearCCFields();

var $creditCard = $('[data-method="HIPAY\_CREDIT\_CARD"]');

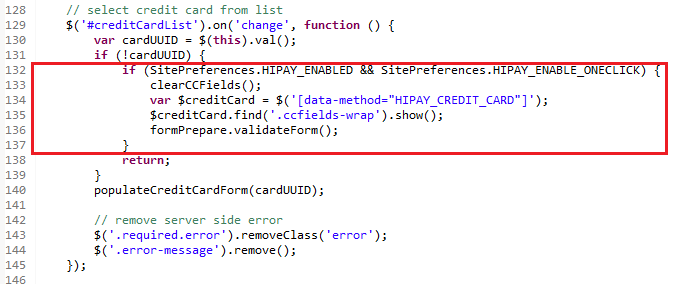
$creditCard.find('.ccfields-wrap').show();

formPrepare.validateForm();

}

After:

if (!cardUUID) {



Add:

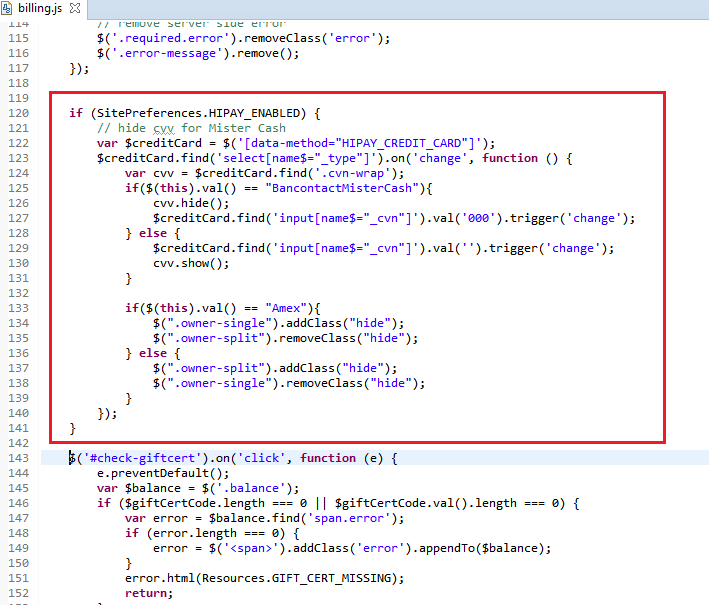
if (SitePreferences.HIPAY\_ENABLED) {

// hide cvv for Mister Cash  
var $creditCard = $('[data-method="HIPAY\_CREDIT\_CARD"]');  
$creditCard.find('select[name$="\_type"]').on('change', function () {  
 var cvv = $creditCard.find('.cvn-wrap');  
 if($(this).val() == "BancontactMisterCash"){  
 cvv.hide();  
 $creditCard.find('input[name$="\_cvn"]').val('000').trigger('change');  
 } else {  
 $creditCard.find('input[name$="\_cvn"]').val('').trigger('change');  
 cvv.show();  
 }  
   
 if($(this).val() == "Amex"){  
 $(".owner-single").addClass("hide");  
 $(".owner-split").removeClass("hide");  
 } else {  
 $(".owner-split").addClass("hide");  
 $(".owner-single").removeClass("hide");  
 }  
});

}

Before:

$('#check-giftcert').on('click', function (e) {



1. **app\_storefront\_core/cartridge/forms/default/billing.xml** (see the int\_hipay\_controllers/cartridge/forms/default/billing\_sample.xml as an example)

Change:

<field formid="selectedPaymentMethodID" type="string" default-value="CREDIT\_CARD">

To:

<field formid="selectedPaymentMethodID" type="string" default-value="HIPAY\_CREDIT\_CARD">

Add:

<!-- HyPay fingerprint -->

<field formid="deviceFingerprint" type="string" />

Before:

<!-- list of available credit cards to select from -->

<list formid="creditCardList">

Add:

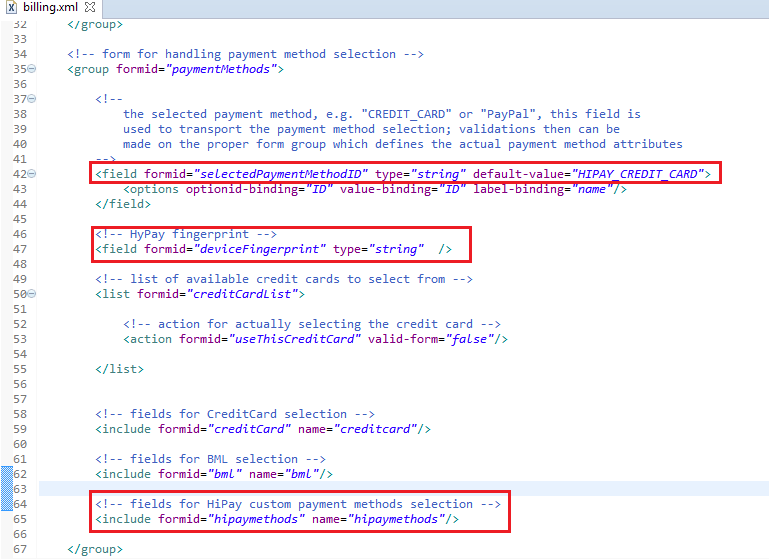
<!-- fields for HiPay custom payment methods selection -->

<include formid="hipaymethods" name="hipaymethods"/>

After:

<!-- fields for BML selection -->

<include formid="bml" name="bml"/>



1. **app\_storefront\_core/cartridge/forms/default/creditcard.xml** (see the int\_hipay\_controllers/cartridge/forms/default/creditcard\_sample.xml as an example)

Change:

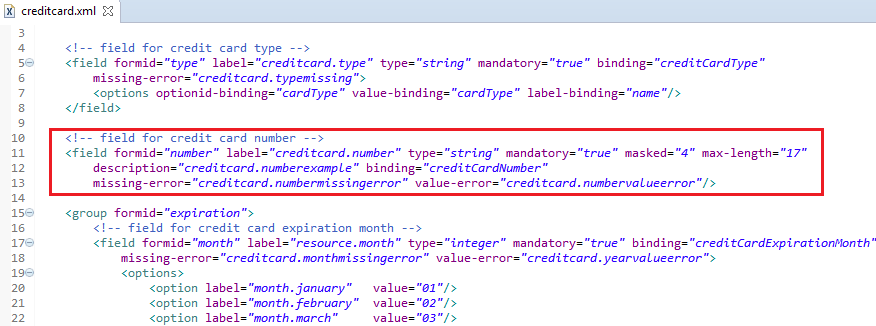
<field formid="number" label="creditcard.number" type="string" mandatory="true" masked="4" max-length="16"

description="creditcard.numberexample" binding="creditCardNumber"

missing-error="creditcard.numbermissingerror" value-error="creditcard.numbervalueerror"/>

To:

<field formid="number" label="creditcard.number" type="string" mandatory="true" masked="4" max-length="17" description="creditcard.numberexample" binding="creditCardNumber" missing-error="creditcard.numbermissingerror" value-error="creditcard.numbervalueerror"/>



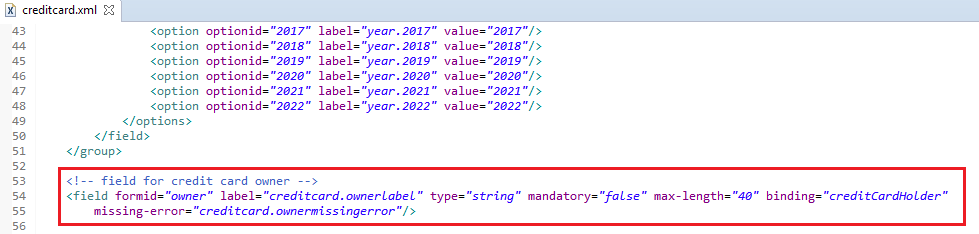
Change:

<field formid="owner" label="creditcard.ownerlabel" type="string" mandatory="true" max-length="40" binding="creditCardHolder"

missing-error="creditcard.ownermissingerror"/>

To:

<field formid="owner" label="creditcard.ownerlabel" type="string" mandatory="false" max-length="40" binding="creditCardHolder" missing-error="creditcard.ownermissingerror"/>



Add:

<!-- field for credit card owner first name -->

<field formid="ownerfirst" label="creditcard.ownerlabel.first" type="string" mandatory="false" max-length="40" missing-error="creditcard.ownermissingerror"/>

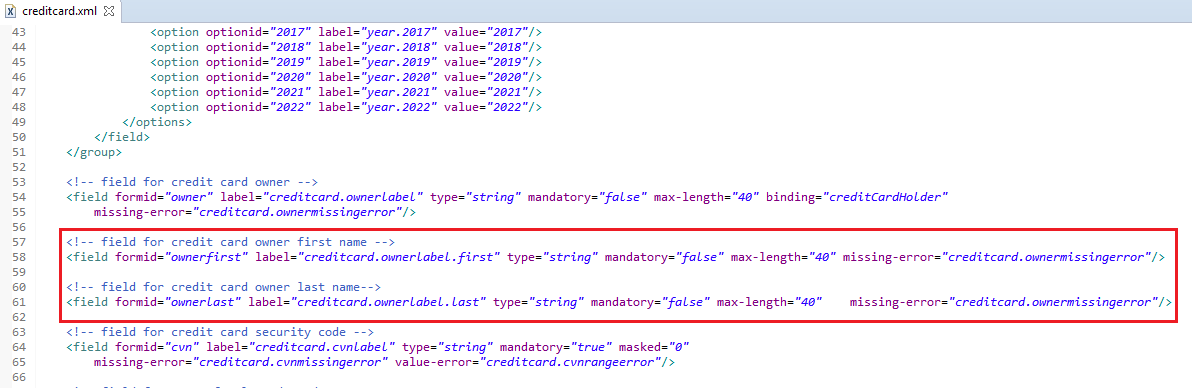
<!-- field for credit card owner last name-->

<field formid="ownerlast" label="creditcard.ownerlabel.last" type="string" mandatory="false" max-length="40" missing-error="creditcard.ownermissingerror"/>

Before:

<!-- field for credit card security code -->

<field formid="cvn" label="creditcard.cvnlabel" type="string" mandatory="true" masked="0" missing-error="creditcard.cvnmissingerror" value-error="creditcard.cvnrangeerror"/>



Add:

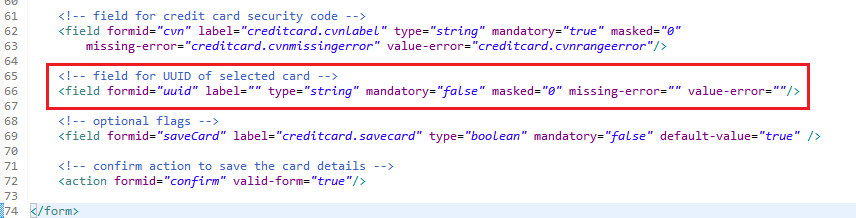
<!-- field for UUID of selected card -->

<field formid="uuid" label="" type="string" mandatory="false" masked="0" missing-error="" value-error=""/>

Before:

<!-- optional flags -->

<field formid="saveCard" label="creditcard.savecard" type="boolean" mandatory="false" default-value="true" />



1. **Create a file app\_storefront\_core/cartridge/forms/default/hipaymethods.xml** (see the int\_hipay\_controllers/cartridge/forms/default/hipaymethods\_sample.xml as an example)

Add next code into the hipaymethods.xml file:

<?xml version="1.0" encoding="UTF-8"?>

<form xmlns="http://www.demandware.com/xml/form/2008-04-19">

<field formid="username"

label="hipay.label.username"

description="resource.5\_100characters"

type="string"

mandatory="true"

binding="login"

regexp="^[-\w\.@\_ ]{5,100}$"

parse-error="profile.usernameerror"

value-error="profile.usernametaken" />

<field formid="password" label="hipay.label.password" description="resource.8\_255characters" type="string" mandatory="true" min-length="8" max-length="255"

range-error="resource.8\_255characters" />

<field formid="issuer\_bank\_id" label="hipay.label.issuer\_bank\_id" default="ABNANL2A" type="string" mandatory="true" >

<options>

<option optionid="" label="resource.select" value=""/>

<option optionid="ABNANL2A" value="ABNANL2A" label="ABN AMRO"/>

<option optionid="INGBNL2A" value="INGBNL2A" label="ING"/>

<option optionid="RABONL2U" value="RABONL2U" label="Rabobank"/>

<option optionid="SNSBNL2A" value="SNSBNL2A" label="SNS Bank"/>

<option optionid="ASNBNL21" value="ASNBNL21" label="ASN Bank"/>

<option optionid="FRBKNL2L" value="FRBKNL2L" label="Friesland Bank"/>

<option optionid="KNABNL2H" value="KNABNL2H" label="Knab"/>

<option optionid="RBRBNL21" value="RBRBNL21" label="SNS Regio Bank"/>

<option optionid="TRIONL2U" value="TRIONL2U" label="Triodos bank"/>

<option optionid="FVLBNL22" value="FVLBNL22" label="Van Lanschot"/>

</options>

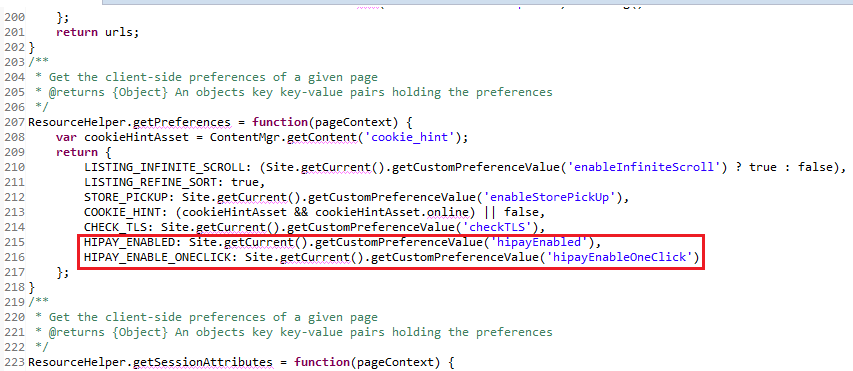
</field>

</form>

1. **app\_storefront\_core/cartridge/scripts/util/Resource.ds**

Add next line into the preferences object:

HIPAY\_ENABLED: Site.getCurrent().getCustomPreferenceValue('hipayEnabled'), HIPAY\_ENABLE\_ONECLICK: Site.getCurrent().getCustomPreferenceValue('hipayEnableOneClick')



The following steps should be applied only to pipeline storefront version. For controllers storefront version please jump to step 13.

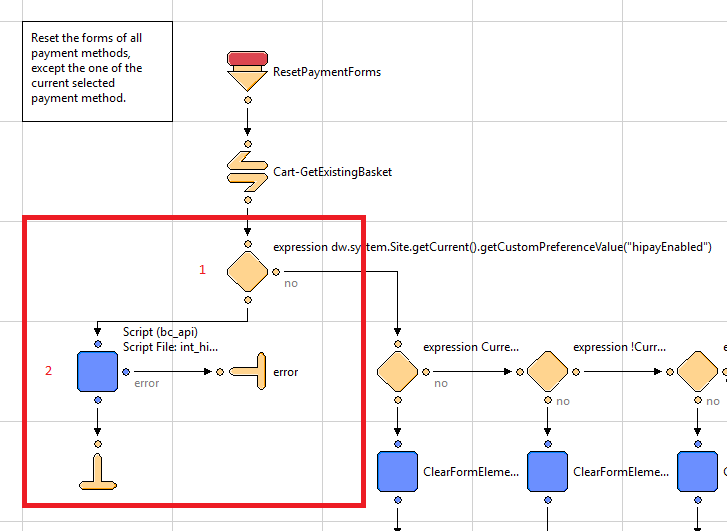
1. **app\_storefront\_pipelines/cartridge/pipelines/COBilling.xml**

There is a change made in COBilling-ResetPaymentForms due to the fact that multiple HiPay payment methods have been added. This change removes the existing payment instruments, if any, and applies only the selected one for case if HiPay enabled.

After the Cart-GetExistingBasket add the Decision node. The ‘no’ transition goes to the default implementation for case if HiPay is disabled.

For case if HiPay is enabled add the script node set to HiPayResetPayments.js and two end nodes as described below.

1. Decision Key : dw.system.Site.getCurrent().getCustomPreferenceValue('hipayEnabled')
2. Script Node : int\_hipay\_controllers:pipelets/HiPayResetPaymentForms.js



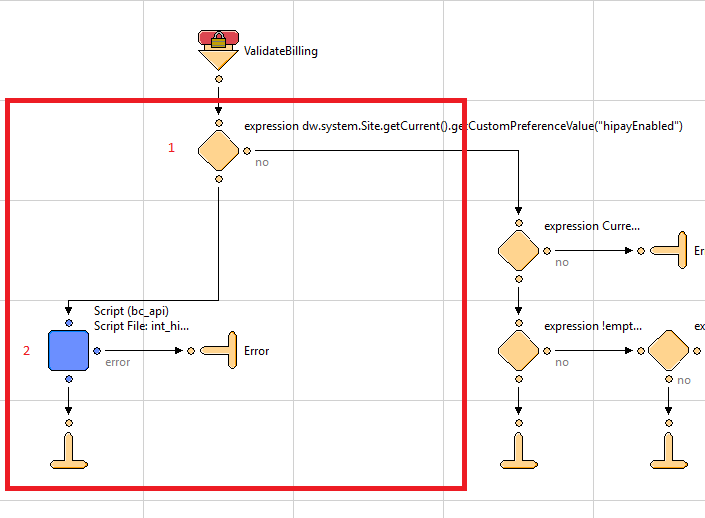
To split the full Card Holder name for American Express cards into First Name and Last Name the validation logic was changed in COBilling-ValidateBilling.

After the ValidateBilling start node add the Decision node. The ‘no’ transition goes to the default implementation for case if HiPay is disabled.

For case if HiPay is enabled add the nodes described below.

For case if HiPay is enabled add the script node set to HiPayValidateBilling.js and two end nodes as described below.

1. Decision Key : dw.system.Site.getCurrent().getCustomPreferenceValue('hipayEnabled')
2. Script Node : int\_hipay\_controllers:pipelets/HiPayValidateBilling.js

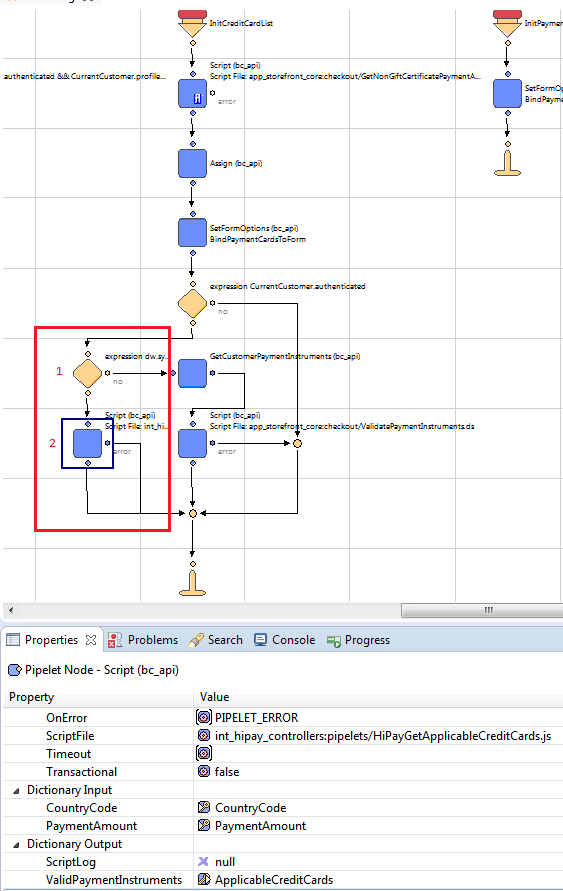


There is a change made in COBilling-InitCreditCardList to initialize the credit cards list using saved customer payment instruments of type hipay credit card.

After the first Decision Node (expression CurrentCustomer.authenticated) add another Decision node as described in the picture below. The ‘no’ transition goes to the default implementation for case if HiPay or HiPay One-Click is disabled.

For case if HiPay and HiPay One-Click is enabled add the script node set to HiPayGetApplicableCreditCards.js as described below.

1. Decision Key : dw.system.Site.getCurrent().getCustomPreferenceValue("hipayEnabled") && dw.system.Site.getCurrent().getCustomPreferenceValue("hipayEnableOneClick")
2. Script Node :
   1. Script File: int\_hipay\_controllers:pipelets/HiPayGetApplicableCreditCards.js
   2. CountryCode: CountryCode
   3. PaymentAmount: PaymentAmount
   4. ValidPaymentInstruments: ApplicableCreditCards

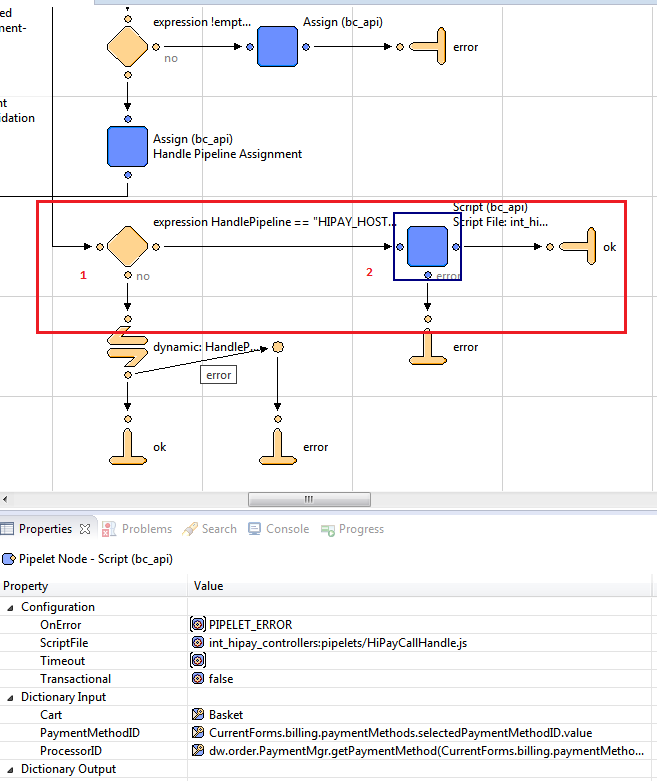


There is a change made in COBilling-HandlePaymentSelection to call HAYPAY or HAYPAY\_HOSTED controller responsible for specific validation and payment instrument creation.

After the “Handle Pipeline Assignment” Assign Node add a Decision node as described in the picture below. The ‘no’ transition goes to the default implementation for case if HandlePipeline is not “HIPAY\_HOSTED” and “HIPAY”.

For case if HandlePipeline is “HIPAY\_HOSTED” or “HIPAY” add the script node set to HiPayCallHandle.js and two end nodes as described below

1. Decision Key : HandlePipeline == "HIPAY\_HOSTED" || HandlePipeline == "HIPAY"
2. Script Node :
   1. Script File: int\_hipay\_controllers:pipelets/HiPayCallHandle.js
   2. Cart: Basket
   3. PaymentMethodID: CurrentForms.billing.paymentMethods.selectedPaymentMethodID.value
   4. ProcessorID:dw.order.PaymentMgr.getPaymentMethod(CurrentForms.billing.paymentMethods.selectedPaymentMethodID.value).paymentProcessor.ID



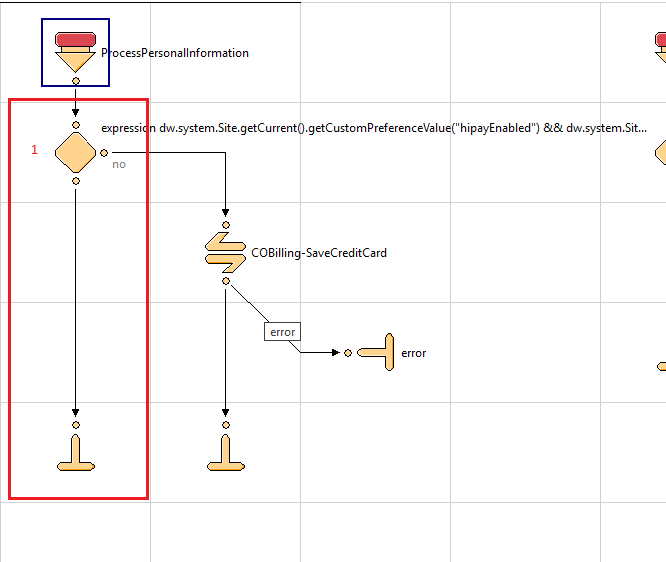
1. **app\_storefront\_pipelines/cartridge/pipelines/COPlaceOrder.xml**

There is a change made in COPlaceOrder-ProcessPersonalInformation to skip the default call to COBilling-SaveCreditCard.

After the ProcessPersonalInformation start node add a decision node. The ‘no’ transition goes to the default implementation for case if HiPay and HiPay One-Click is disabled.

For case if HiPay and HiPay One-Click is enabled add the end node described below

1. Decision Key: dw.system.Site.getCurrent().getCustomPreferenceValue("hipayEnabled") && dw.system.Site.getCurrent().getCustomPreferenceValue("hipayEnableOneClick")

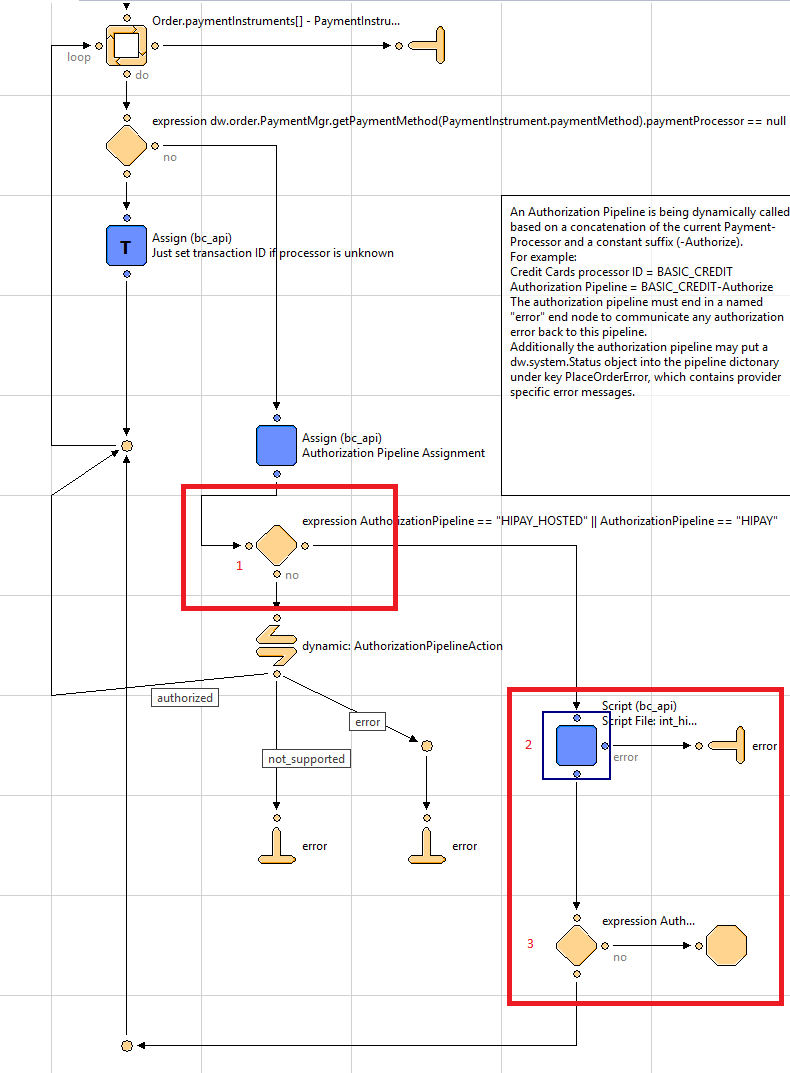


There is a change made in COPlaceOrder-HandlePayments to call HAYPAY or HAYPAY\_HOSTED controller responsible for authorize HiPay payment.

After the “Authorization Pipeline Assignment” Assign Node add a Decision node as described in the picture below. The ‘no’ transition goes to the default implementation for case if AuthorizationPipeline is not “HIPAY\_HOSTED” and “HIPAY”.

For case if AuthorizationPipeline is “HIPAY\_HOSTED” or “HIPAY” add the script node set to HiPayCallHandle.js, one end node, one decision node and a stop node as described below

1. Decision Key : AuthorizationPipeline == "HIPAY\_HOSTED" || AuthorizationPipeline == "HIPAY"
2. Script Node :
   1. Script File: int\_hipay\_controllers:pipelets/HiPayCallAuthorize.js
   2. Order: Order
   3. PaymentInstrument: PaymentInstrument
   4. PaymentMethodID: PaymentInstrument.paymentMethod
   5. ProcessorID: dw.order.PaymentMgr.getPaymentMethod(PaymentInstrument.paymentMethod).paymentProcessor.ID
   6. AuthorizationResult: AuthorizationResult
3. Decision Key : AuthorizationResult.authorized



1. **app\_storefront\_pipelines/cartridge/pipelines/PaymentInstruments.xml**

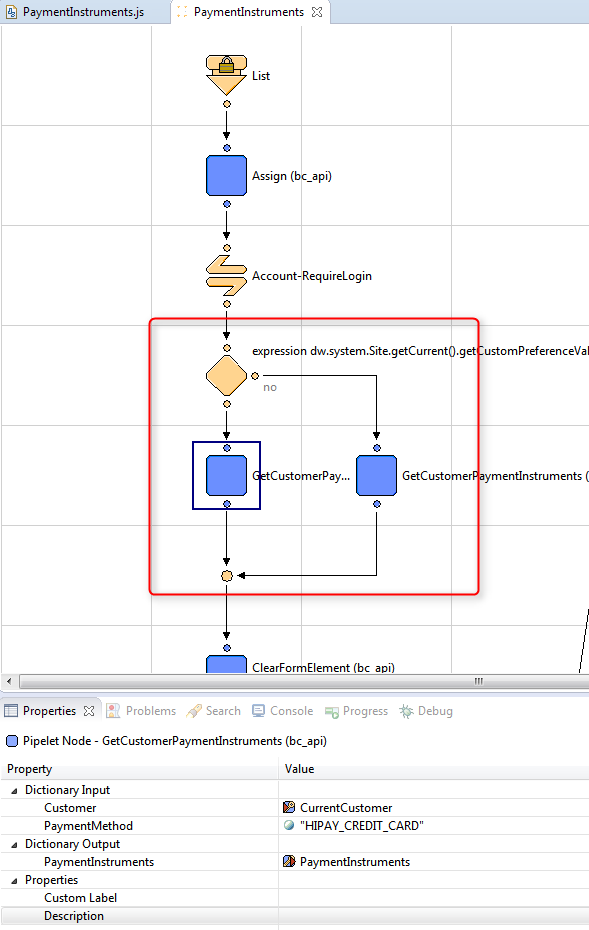
There is a change made in PaymentInstruments-Llist to get the saved credit cards info.

After the Account-RequireLogin call node add a decision node with next parameters:

1. Decision Key : dw.system.Site.getCurrent().getCustomPreferenceValue("hipayEnabled") && dw.system.Site.getCurrent().getCustomPreferenceValue("hipayEnableOneClick")

The ‘no’ transition goes to the default implementation. For case if HiPay and HiPay One-Click enabled add a GetCustomerPaymentInstruments pipelet with next parameters:

1. Customer : CurrentCustomer
2. PaymentMethod : “HIPAY\_CREDIT\_CARD”
3. PaymentInstruments : PaymentInstruments



The next steps should be applied to controller storefront version.

1. **app\_storefront\_controllers/cartridge/controllers/COBilling.js**

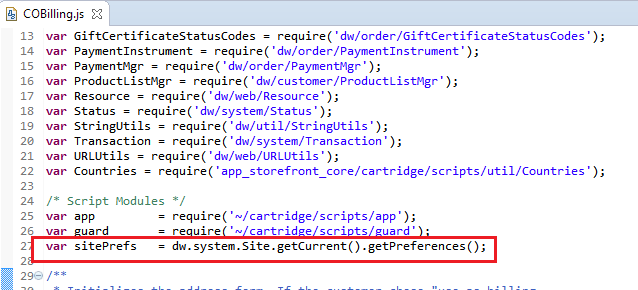
Add:

var sitePrefs = dw.system.Site.getCurrent().getPreferences();

After:

var app = require('~/cartridge/scripts/app');

var guard = require('~/cartridge/scripts/guard');

****

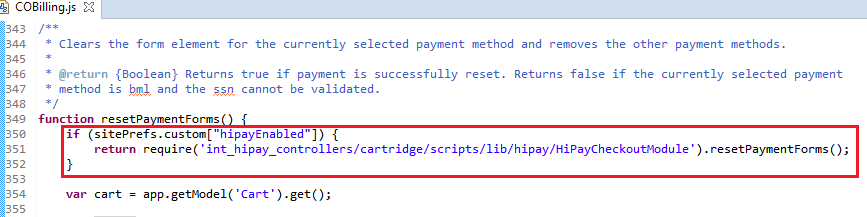
There is a change made in the resetPaymentForms() function due to the fact that multiple HiPay payment methods have been added. This change removes the existing payment instruments, if any, and applies only the selected one for case if HiPay enabled.

In the resetPaymentForms()function insert the following lines as described in the example bellow.

if (sitePrefs.custom("hipayEnabled")) {

return require('int\_hipay\_controllers/cartridge/scripts/lib/hipay/HiPayCheckoutModule').resetPaymentForms();

}

****

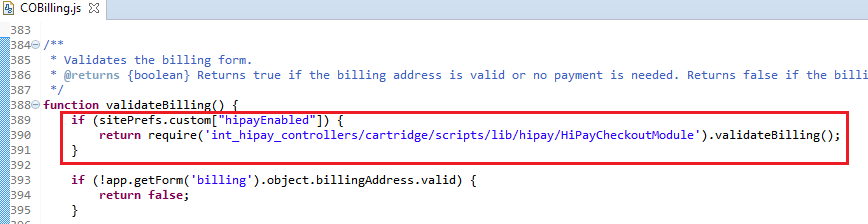
To split the full Card Holder name for American Express cards into First Name and Last Name the validation logic was changed in the validateBilling() function.

In the validateBilling() function insert the following lines as described in the example bellow.

if (sitePrefs.custom("hipayEnabled")) {

return require('int\_hipay\_controllers/cartridge/scripts/lib/hipay/HiPayCheckoutModule').validateBilling();

}

****

There is a change made in the initCreditCardList(cart) function to initialize the credit cards list using saved customer payment instruments of hipay credit card type.

Replace next code:

if (customer.authenticated) {

var profile = app.getModel('Profile').get();

if (profile) {

applicableCreditCards = profile.validateWalletPaymentInstruments(countryCode, paymentAmount.getValue()).ValidPaymentInstruments;

}

}

With next code:

if (customer.authenticated) {

if (sitePrefs.custom["hipayEnabled"] && sitePrefs.custom["hipayEnableOneClick"]) { /\* if hipay and one click payment enabled \*/

applicableCreditCards = require('int\_hipay\_controllers/cartridge/scripts/lib/hipay/HiPayCheckoutModule').getApplicableCreditCards(countryCode, paymentAmount.getValue()).ValidPaymentInstruments;

} else {

var profile = app.getModel('Profile').get();

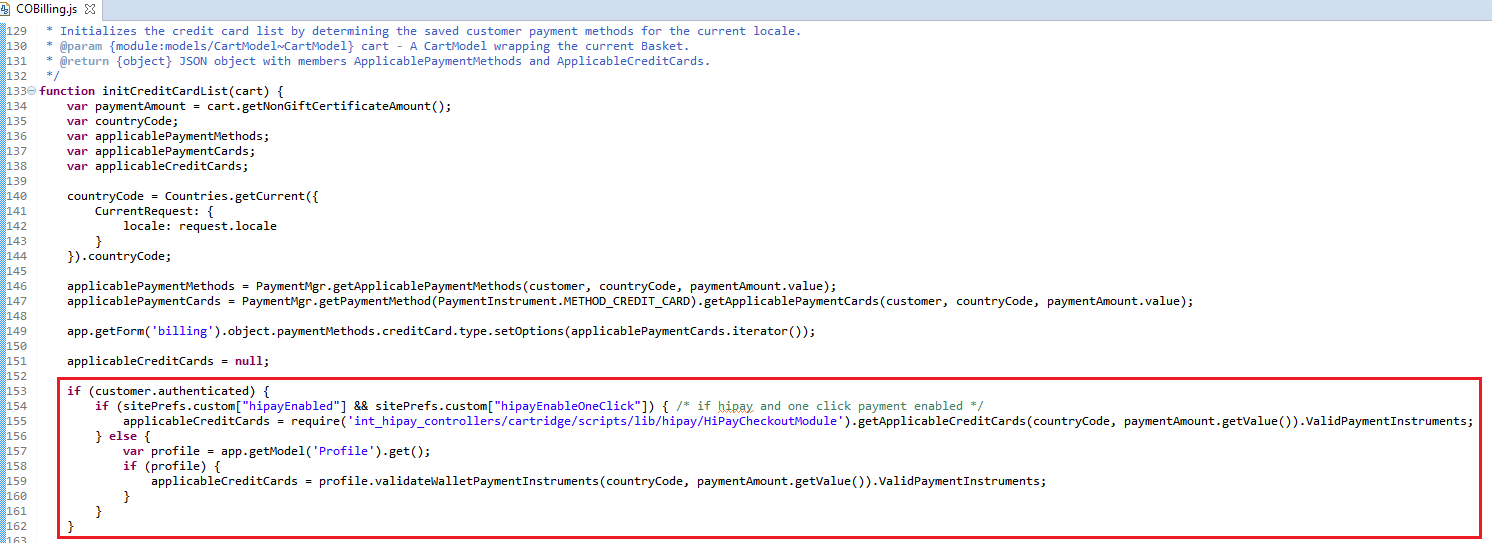
if (profile) {

applicableCreditCards = profile.validateWalletPaymentInstruments(countryCode, paymentAmount.getValue()).ValidPaymentInstruments;

}

}

}

****

There is a change made in the saveCreditCard() function to skip the default implementation.

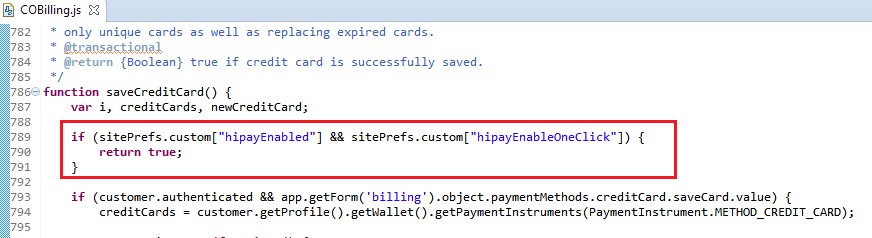
In the saveCreditCard() function insert the following lines as described in the example bellow.

if (sitePrefs.custom["hipayEnabled"] && sitePrefs.custom["hipayEnableOneClick"])

{

return true;

}

****

1. **app\_storefront\_controllers/cartridge/controllers/COPlaceOrder.js**

Add:

var sitePrefs = dw.system.Site.getCurrent().getPreferences();

After:

var PaymentProcessor = app.getModel('PaymentProcessor');

****

In the handlePayments(order) function to define if the payment was authorized or redirected.

Add:

if (sitePrefs.custom["hipayEnabled"] && authorizationResult.authorized) {

return {

authorized : true

}

}

After:

if (authorizationResult.not\_supported || authorizationResult.error) {

return {

error: true

};

}

****

In the start() function add next changes.

Replace:

var orderPlacementStatus = Order.submit(order);

if (!orderPlacementStatus.error) {

clearForms();

}

return orderPlacementStatus;

With:

if (sitePrefs.custom["hipayEnabled"]) {

if (handlePaymentsResult.authorized) {

var orderPlacementStatus = Order.submit(order);

if (!orderPlacementStatus.error) {

clearForms();

}

return orderPlacementStatus;

} else {

return {

Order: order,

order\_created: false

};

}

} else {

var orderPlacementStatus = Order.submit(order);

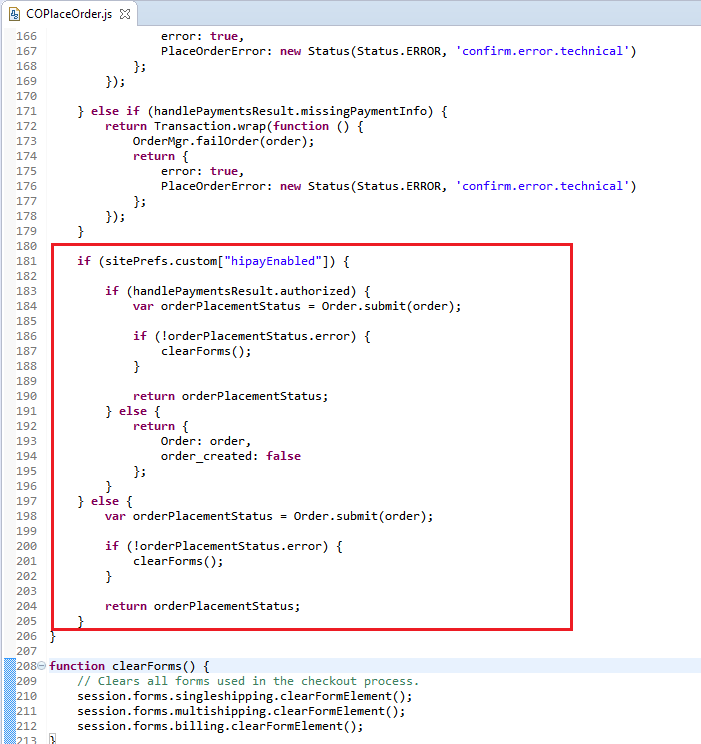
if (!orderPlacementStatus.error) {

clearForms();

}

return orderPlacementStatus;

}

****

1. **app\_storefront\_controllers/cartridge/controllers/PaymentInstruments.js**

In the list() function add next changes.

Add:

var sitePrefs = dw.system.Site.getCurrent().getPreferences();

Replace:

var paymentInstruments = wallet.getPaymentInstruments(dw.order.PaymentInstrument.METHOD\_CREDIT\_CARD);

With:

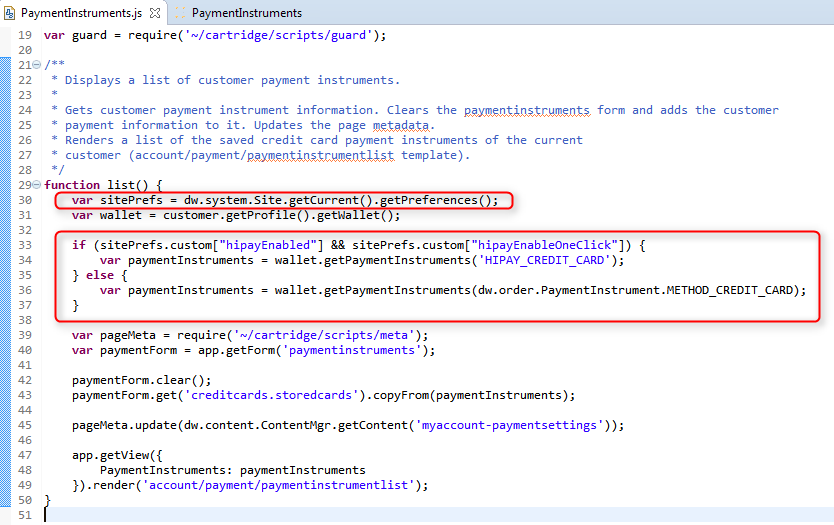
if (sitePrefs.custom["hipayEnabled"] && sitePrefs.custom["hipayEnableOneClick"]) {

var paymentInstruments = wallet.getPaymentInstruments('HIPAY\_CREDIT\_CARD');

} else {

var paymentInstruments = wallet.getPaymentInstruments(dw.order.PaymentInstrument.METHOD\_CREDIT\_CARD);

}



### HiPay Multi-account

If you don’t need different HiPay accounts for each site, please skip this step.

In order to provide the ability to handle different HiPay accounts for each site, additional 4 service configurations should be added per site in hiPayServiceInit.js :

* **hipay.rest.createtoken.{siteID}**
* **hipay.rest.hpayment.{siteID}**
* **hipay.rest.maintenance.{siteID}**
* **hipay.rest.order.{siteID}**

Where siteID is the ID of the site for which you wish to add new HiPay account. A code example for a new HiPay account is added in the hiPaySerivceInit.js that is located in **int\_hipay\_controllers\cartridge\scripts\init** folder. You need to copy and paste in the same file the default services code and change the line in each service configuration where name of service is defined:

Default:

ServiceRegistry.configure("hipay.rest.hpayment", {

Example for new one:

ServiceRegistry.configure("hipay.rest.hpayment.SiteGenesis", {

A full example appears and is commented in hiPaySerivceInit.js. After extending the code with new service registry configurations, you will need to provide new service configurations in BM after this step, described in [5.2.3 Services](#_Services).

## External Interfaces

None.

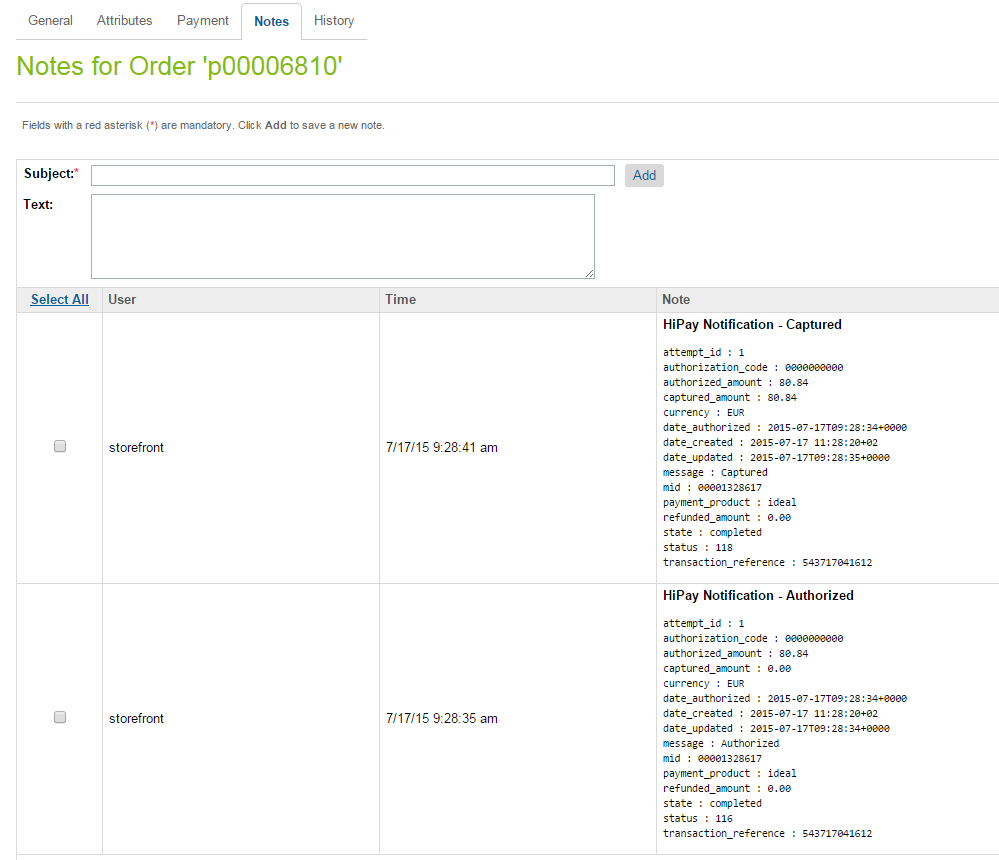
## Testing

Please refer to the use cases in section 2.2. If you have any issues, our technical team will work with you to ensure that everything is thoroughly tested and meets all requirements.

# Operations, Maintenance

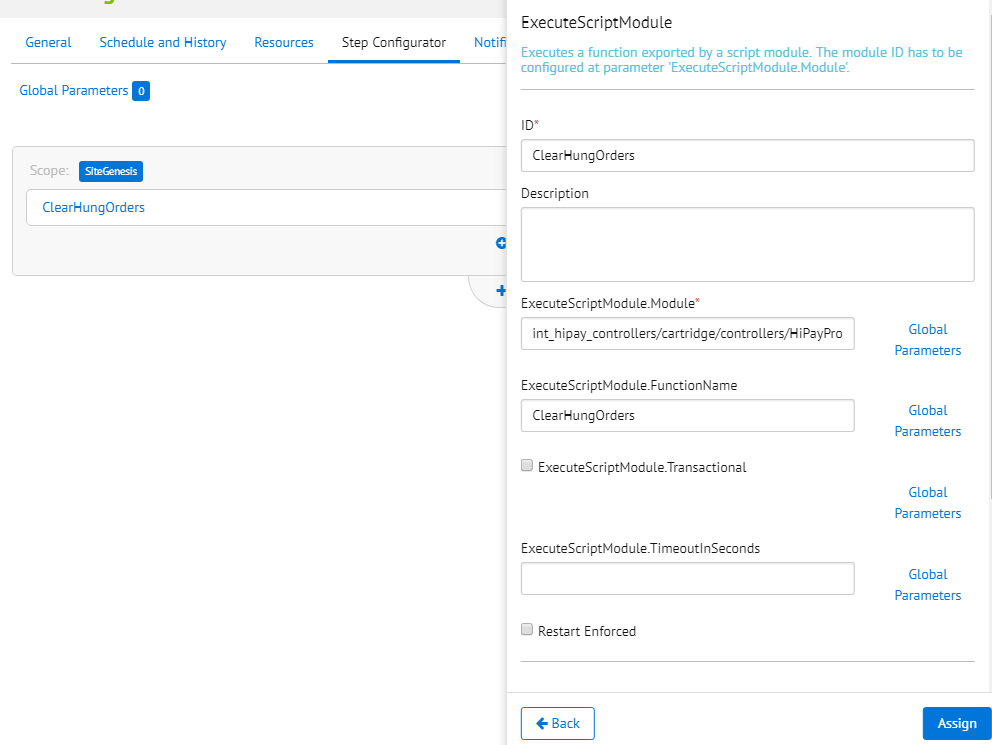
## Data Storage

### Orders

HiPay Fullservice sends notifications for each event that occurs. In order to handle all notifications, notes are added to the order. There should be one event logged for each notification.

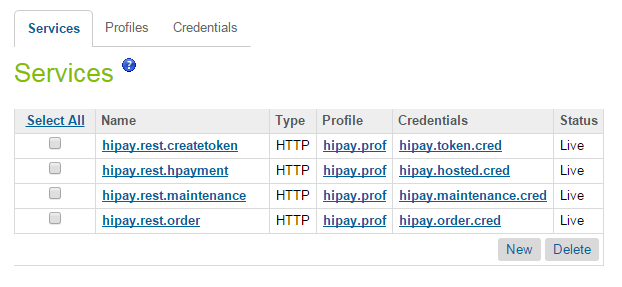
### ClearHungOrders job

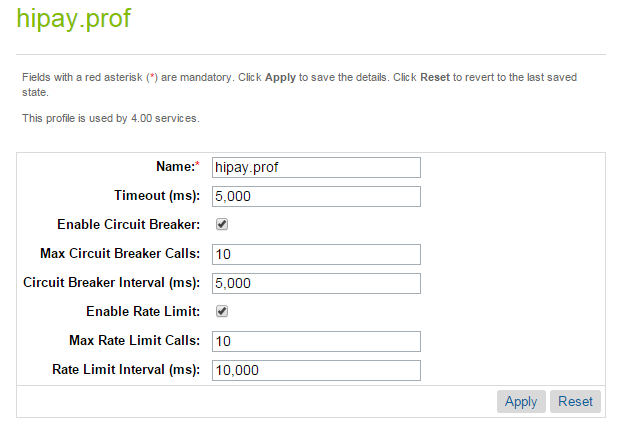
When using the HiPay hosted page or an iFrame, the actual payment can be interrupted due to, for example, the user closing out of the browser. In this situation the orders are left in created state and no further progress is made. In order to handle this these orders, a ClearHungOrders job has been created. This can be configured to change the order status to “Failed” and it runs on the configured timeframe (for example, at 30-minutes intervals).



## Availability

HiPay integration cartridge uses the below services.



All services are using the below Profile configuration.

The circuit breaker suspends platform calls to a web service if a certain number of calls fail within a specified time interval. It can be easily changed for different merchants. A different/ a separate Profile can be added for each service.

## Support

In the event of problems with the integration, missing features, etc. please contact the HiPay’s Business IT Services at <http://help.hipay.com> or your HiPay’s account manager.

# User Guide

## Roles, Responsibilities

The store administrator should check the correct configuration of the HiPay Merchant account in the HiPay back office and should check the receipt of HiPay notification messages on a regular basis.

Admins should regularly, check all the payments received to ensure they contain the expected data (currencyCode, amount, shopperEmail, etc).

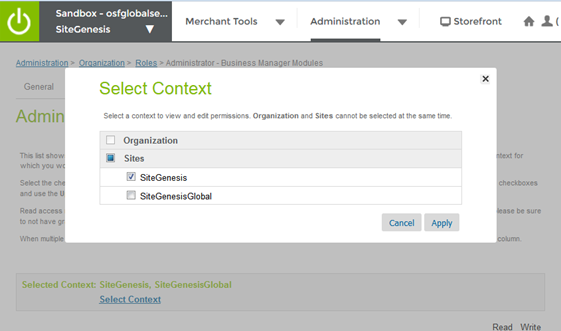
## Business Manager

### HiPay business module

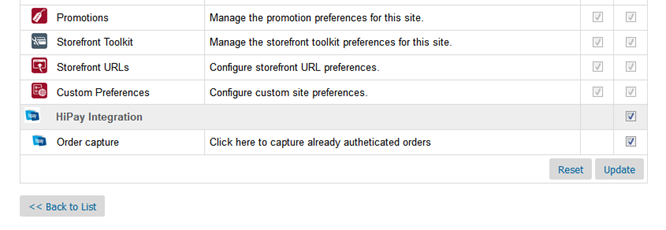
HiPay Integration business modules have been created. It is used to capture amounts for already authorized orders.

**Configuration**

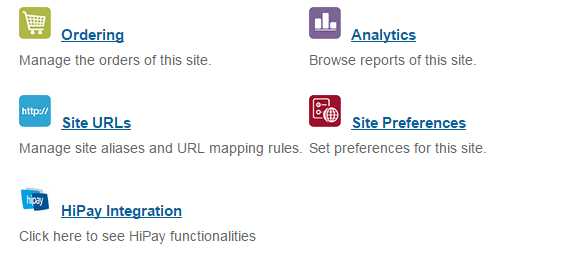
1. Go to: Administration > Organization > Roles > Administrator - Business Manager Modules
2. Select your site context and press **Apply.**

****

1. Check HiPay Integration module and press Update.

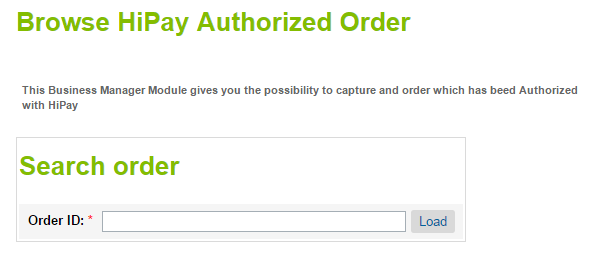
****

1. Go to the Business Manager> Merchant Tools> HiPay module should be displayed.

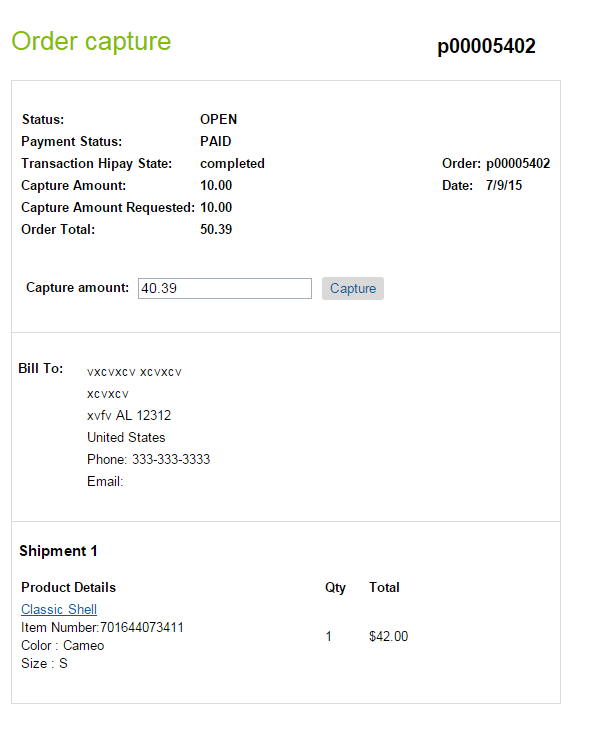


Usage

1. Go to: **HiPay Integration > Order capture**

****

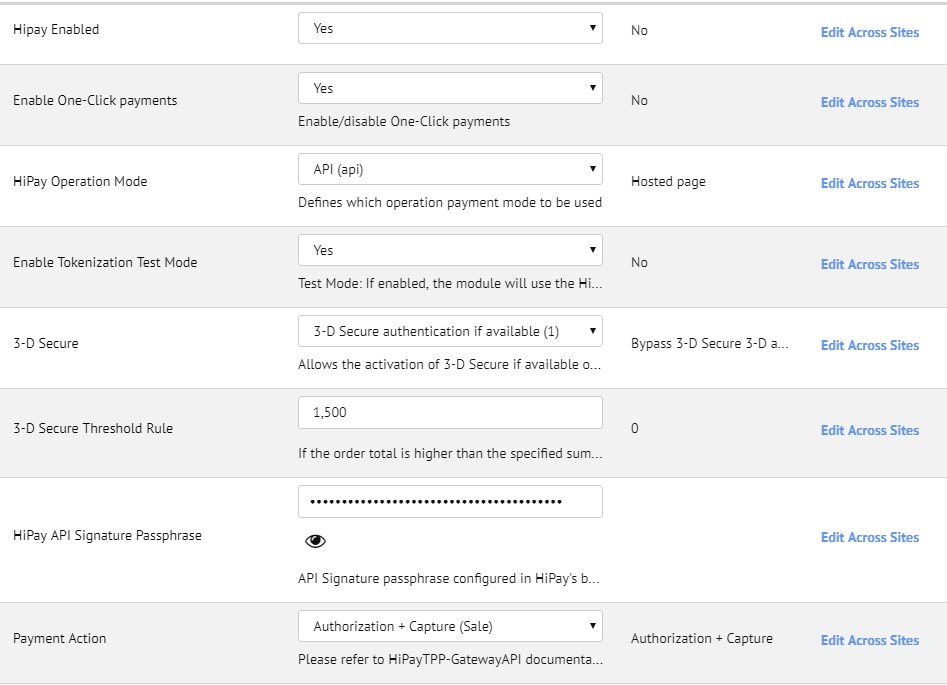
1. Enter an order ID and press **Load**

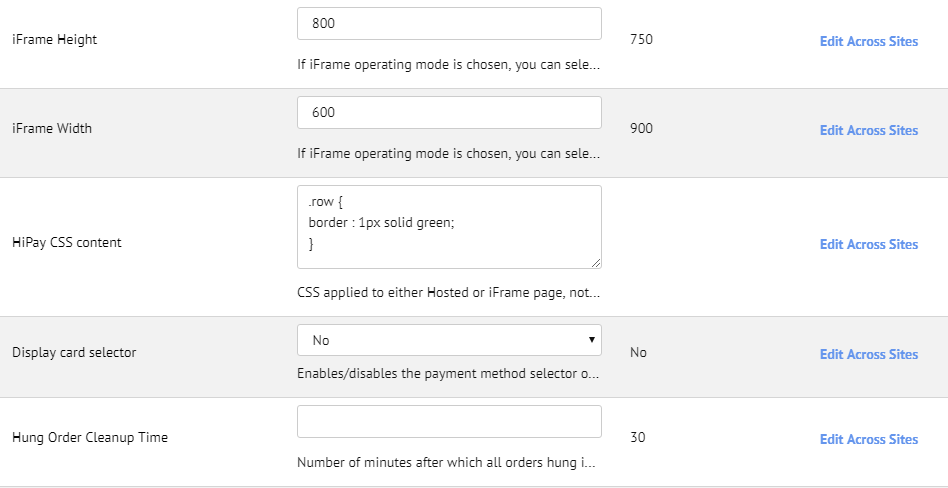


1. Partial or full capture can be requested. If the requested amount is captured, this will send a HiPay notification to Salesforce Commerce Cloud and the **Capture amount** will be updated.

### HiPay Site Preferences

Go to: Merchant Tools > Site Preferences > Custom Site Preferences > HiPay Settings



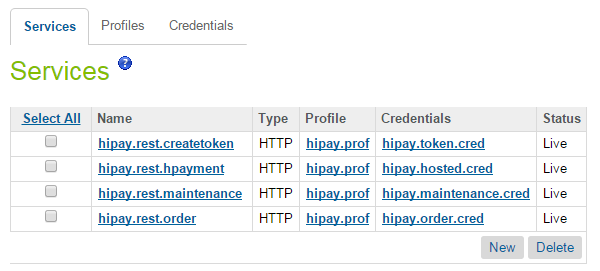


Here you will find the HiPay configuration options.

|  |  |  |  |
| --- | --- | --- | --- |
| Property name | Possible values | Default Value | Description |
| Hipay Enabled | true / false | false | Enable/disable HiPay functionality |
| Enable One-Click payments | true / false | false | Enable/disable One-Click payments |
| HiPay Operation Mode | hosted (Hosted page)  iframe (IFrame)  api (API) | hosted (Hosted page) | Defines which operation payment mode to be used |
| Enable Tokenization Test Mode | true / false | false | Test Mode: If enabled, the module will use the Hipay TPP test platform |
| 3D Secure | 0 *(Bypass 3-D Secure 3-D authentication)*  1 (3-D Secure authentication if available)  2 (3-D Secure authentication mandatory) | 0 *(Bypass 3-D Secure 3-D authentication)* | Allows the activation of 3D secure if available on used card |
| 3D Secure Threshold Rule | Integer | 0 | If the Order total is higher than the specified sum then 3D Secure will be forced. If 0 is specified the rule is disabled. |
| HiPay API Signature Passphrase | String |  | API Signature passphrase configured in HiPay backoffice. Use to verify the requests made to DW. |
| Payment Action | Sale *(Authorization + Capture)*  Authorization (Authorization Only) |  | Please refer to HiPayTPP-GatewayAPI documentation, chapter 3.1 Request a New Order – operation |
| iFrame Height | Integer | 750 | If iFrame operating mode is chosen, you can select your iFrame height to fit with your CSS |
| iFrame Width | Integer | 950 | If iFrame operating mode is chosen, you can select your iFrame width to fit with your CSS |
| HiPay CSS content | Text |  | CSS applied to either Hosted or iFrame page |
| Display card selector | true / false | false | Enable/disable the payment methods selector on iFrame and Hosted page |
| Hung Order Cleanup Time | Integer | 30 | The time in minutes after which all Orders hung in status CREATED that are left by the payment processing are cleaned. |

### Services

Go to: Administration > Operations > Services



There are four services implemented. Each one corresponds to a different HiPay service, as   
follows:

* **hipay.rest.createtoken** – handles the credit card tokenization
* **hipay.rest.hpayment –** handles the hosted payment calls
* **hipay.rest.maintenance** – handles the order updates after being placed
* **hipay.rest.order** – handles placing order

#### HiPay Multi-account

If you don’t need different HiPay accounts for each site, please skip this step.

In order to configure sandbox for multiple HiPay accounts you need duplicate previous services with new names:

* **hipay.rest.createtoken.{siteID}**
* **hipay.rest.hpayment.{siteID}**
* **hipay.rest.maintenance.{siteID}**
* **hipay.rest.order.{siteID}**

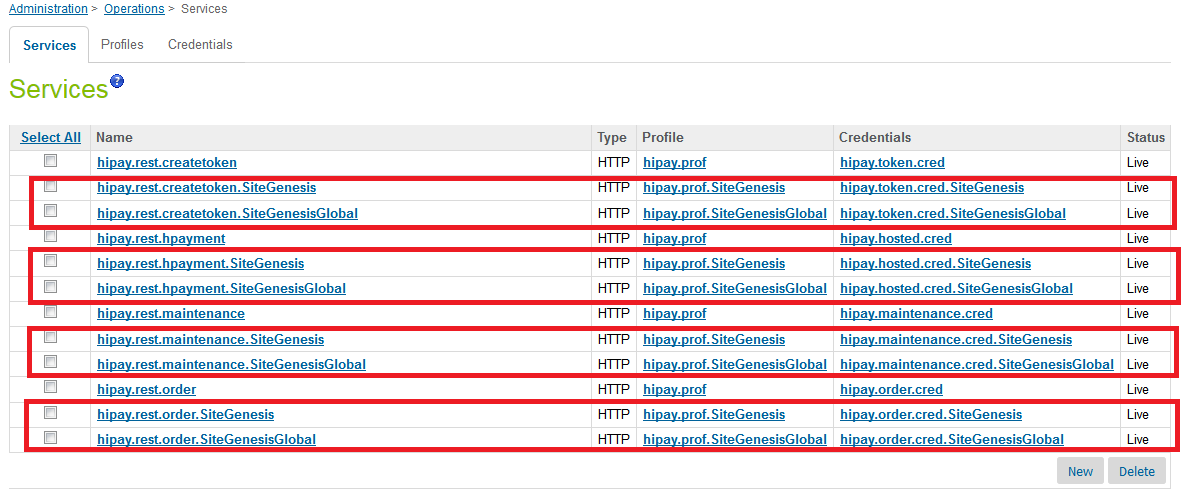
Where siteID is the ID of the site for which you wish to add a new HiPay account. Services without a siteID in the service name should not be deleted as, they will become default services, in case a specific service cannot be found in BM. You can use the **hipay.prof** profile for those services or you can create a new one in order to make customizations in future for specific site. Also you will need duplicate default credentials:

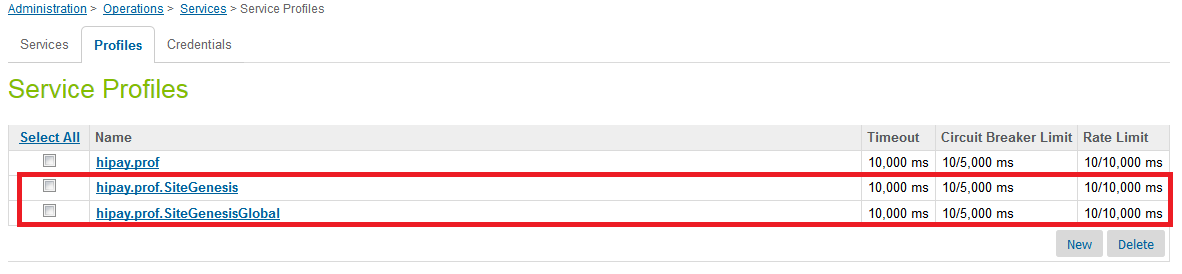
* **hipay.hosted.cred**
* **hipay.maintence.cred**
* **hipay.order.cred**
* **hipay.token.cred**

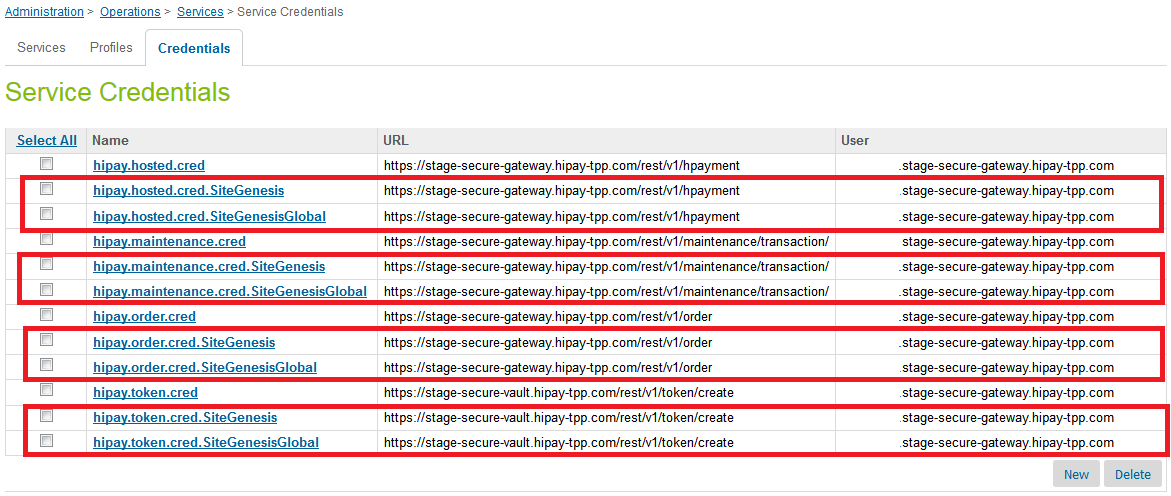
Update them with a new name, HiPay account username and credentials.This is an example mapping for each service which credential should be assigned:

* **hipay.rest.hpayment.SiteGenesis** *service* **- hipay.hosted.cred.SiteGenesis** *credential*
* **hipay.rest.createtoken.SiteGenesis** *service* **- hipay.token.cred.SiteGenesis** *credential*
* **hipay.rest.order.SiteGenesis** *service* **- hipay.order.cred.SiteGenesis** *credential*
* **hipay.rest.maintenance.SiteGenesis** *service* **- hipay.maintenance.cred.SiteGenesis** *credential*

Below you may find an example with Services configured for two different sites and two HiPay accounts (please refer to the marked lines).



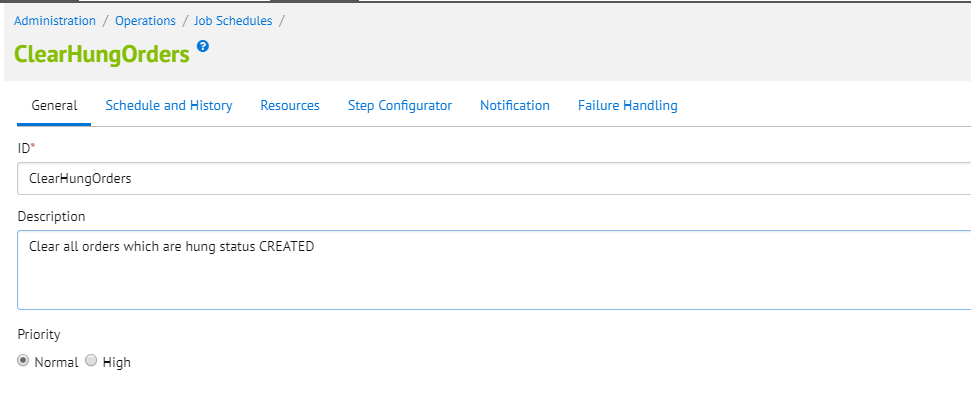
****

****

### Schedules

A **ClearHungOrder** process is implemented which is used to clear all orders that have been started with a HiPay hosted solution but have not been completed within the configured time period.

Go to: **Administration > Operations > Job Schedules > ClearHungOrders**

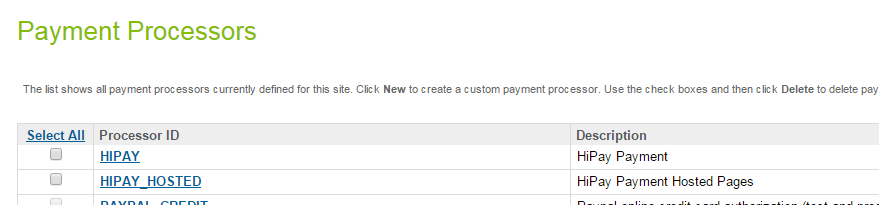
****

Under the “Step Configurator” tab assign the job to your site.

### Payment Processors

There are two payment processors added for HiPay – **HIPAY\_HOSTED** and **HIPAY.** The hosted is used only for the hosted and iframe solution. HIPAY processor is used for the HiPay API integration. It is used for all different types of payments like credit cards, IDEAL, ING HomePay, Giropay, KLARNA, etc.

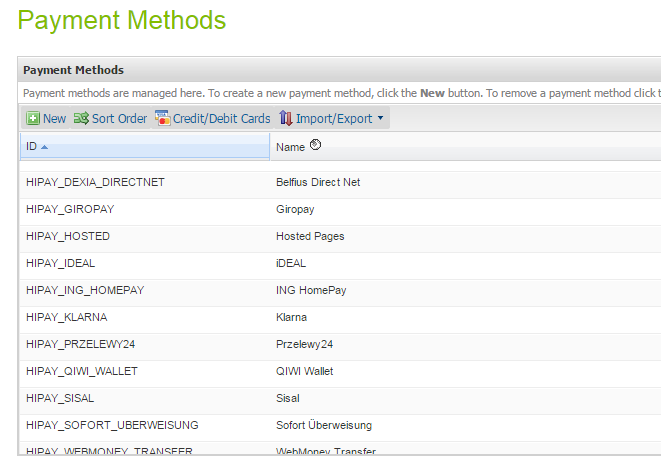
Go to: **Merchant Tools > Ordering > Payment Processors**

****

### Payment Methods

All possible HiPay payment methods have been added to the BM. In order to function in Salesforce Commerce Cloud they must also be activated in the HiPay Fullservice application.

Go to: **Merchant Tools > Ordering > Payment Methods**

****

### Logs

The following logs can be found related to the HiPay implementation:

* service-HIPAY-REST-\*service\*-blade1-3-appserver-\*date\*.log – logs HiPay services specific information
* error-blade1-3-appserver-\*date\*.log – logs Salesforce Commerce Cloud errors
* customerror-blade1-3-appserver-\*date\*.log – logs Salesforce Commerce Cloud custom errors

All of the logs can be found in: Administration > Site Development > Development Setup

Select logs and enter you BM credentials if needed.

### Notification url

Go to HiPay Fullservice -> Integration -> Notifications

In the Notification URL, the following url should be added:

[http://\*domain.name\*/on/demandware.store/Sites-SiteGenesis-Site/default/HiPayNotification-Notify](http://*domain.name*/on/demandware.store/Sites-SiteGenesis-Site/default/HiPayNotification-Notify)

This url will be used to handle all HiPay notifications.

### Custom CSS configuration

In order to use custom CSS for HiPay a custom site preference must be configured with the desired CSS.

Go to: **Merchant Tools > Site Preferences > Custom Site Preferences,** select the HiPay custom preferences group. In the **HiPay CSS content** field enter the CSS that is to be utilized on the payment page and click **Apply.** After completing this configuration, the applied CSS should be visible. The content of this preference is printed via a controller called HiPayResource-Style and its URL (e.g. [http://\*domain.name\*/on/demandware.store/Sites-\*site.name\*-Site/default/HiPayResource-Style](http://*domain.name*/on/demandware.store/Sites-*site.name*-Site/default/HiPayResource-Style) ) is sent when calling the HiPay service, which in turn adds this CSS in the header section of the iFrame or hosted page so it can be applied.

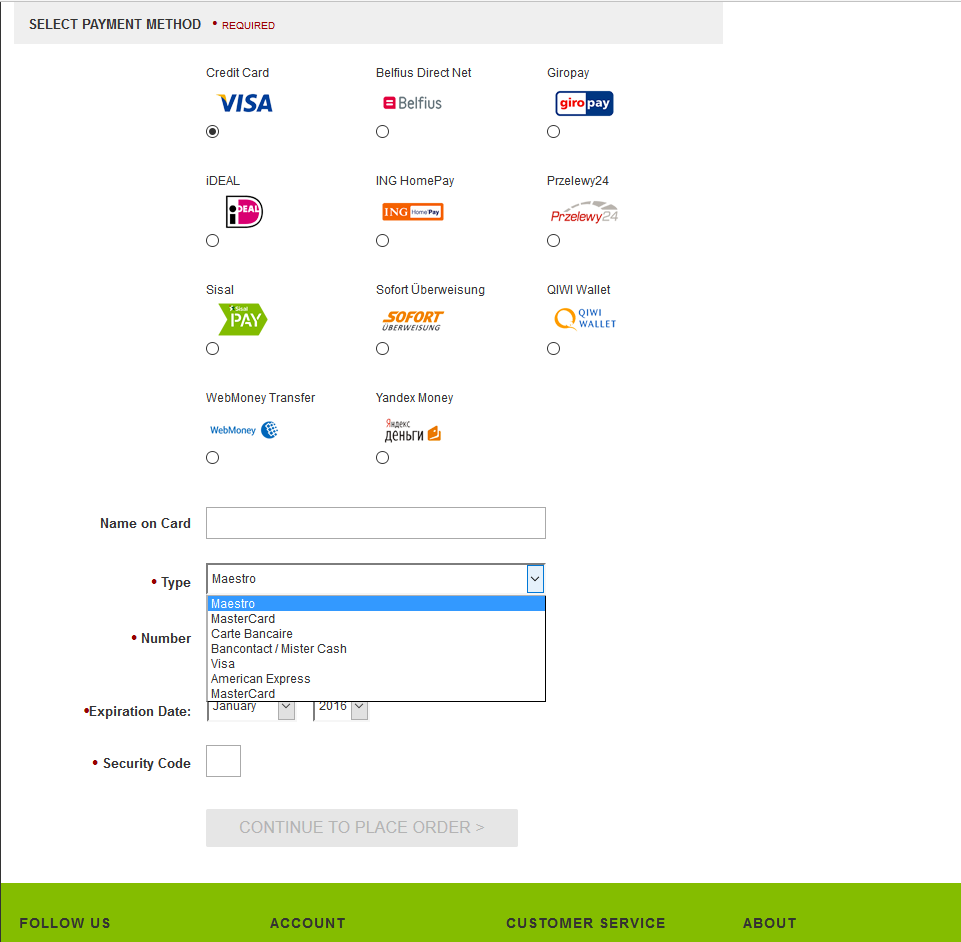
The CSS used must be plain (@include syntax is not supported) and any <style> tags must be omitted.

## Storefront Functionality

Storefront functionality is fully described in section Use Cases.

### Merchant Payment API - Credit Cards

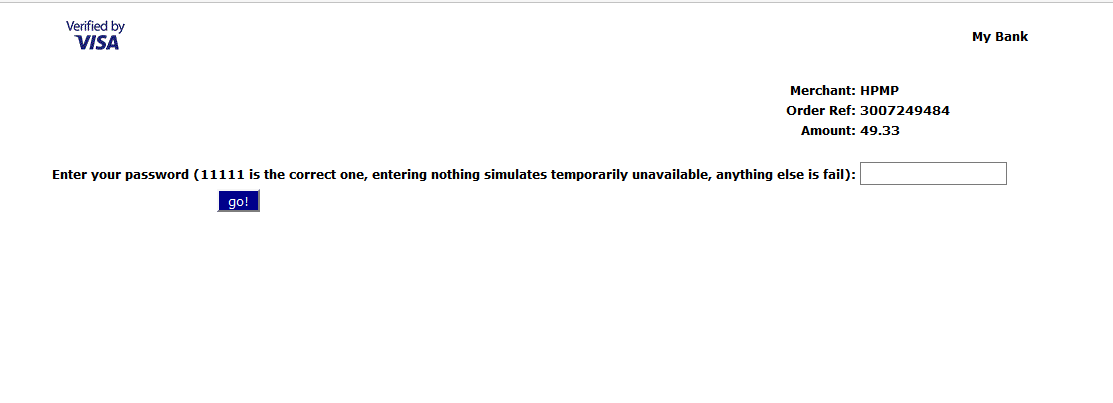
When using this method, the storefront functionality will be the same as the standard one for credit cards in Site Genesis. The card is authorized inline during checkout directly on the merchant's website.



### Payment API - Merchant Link Credit Cards (with 3DSecure)

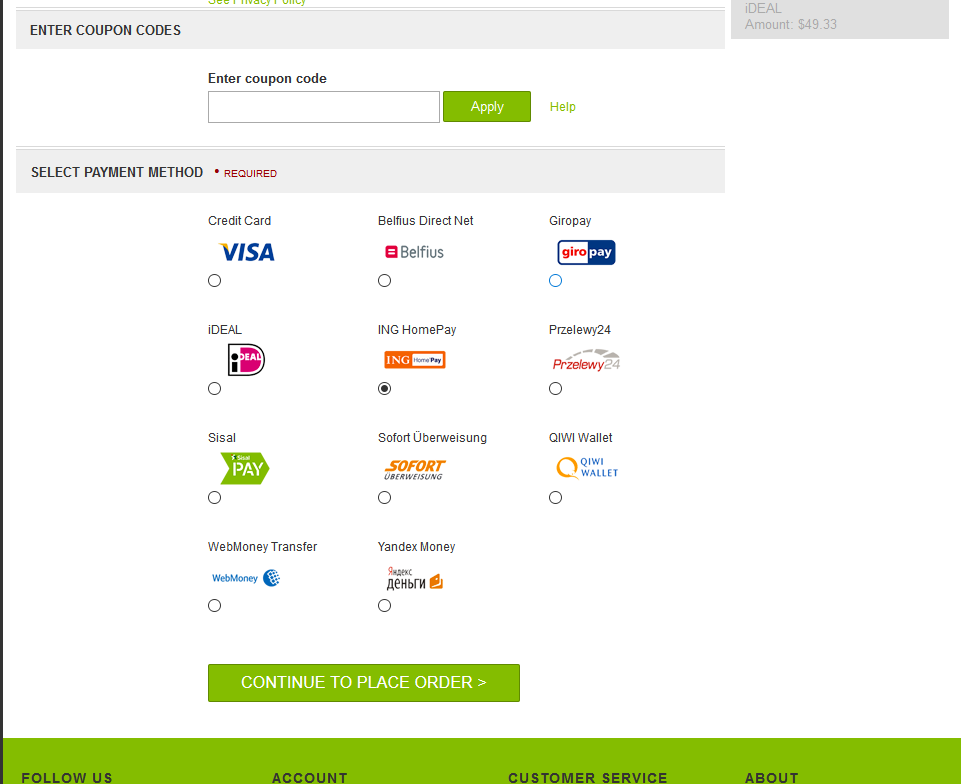
When using this method, the customer still selects what type of card they wish to use on the Billing page.

After Submitting the order from the summary page, if card is applicable by the customer’s bank to 3D Secure, they will be directed to the 3DSecure page.

  
When payment is complete, the customer is redirected to the confirmation page as per the normal flow.

### Hosted Merchant Link

When using these methods, if the currency is applicable with one of these methods, the bank payment option will be shown (each one of them work only in specific currencies).

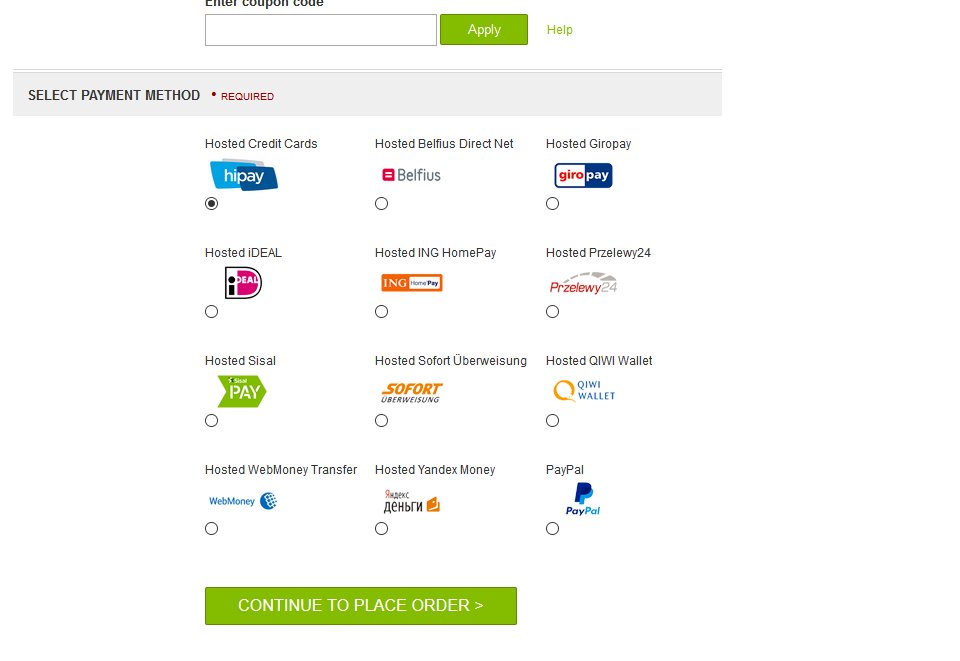


After Submitting the order from the summary page, the customer is redirected to the 3rd party website.

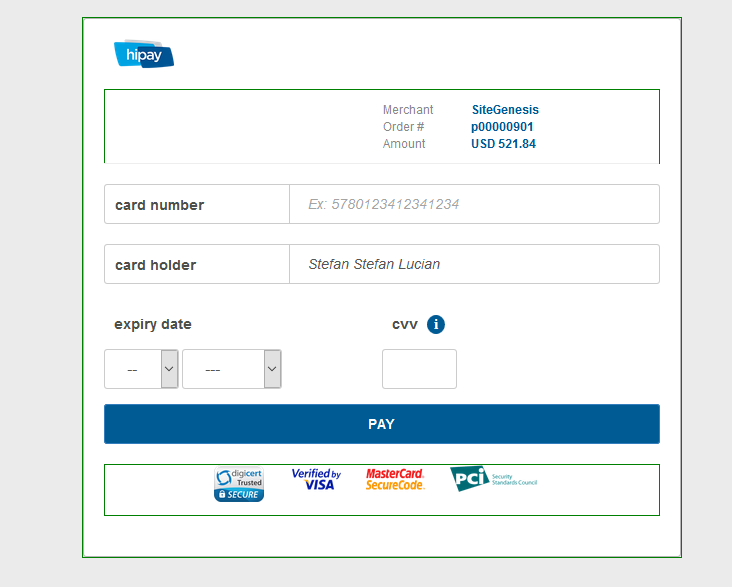
After completing payment with the 3rd party, the customer is redirected back to the storefront. If the payment is successful or unknown, the customer is taken to the confirmation page. Is the payment was unsuccessful, the customer is given an error message on the summary page and asked to select a different payment method.

### Hosted Merchant Link Credit Cards

On the billing page, the customer selects Hosted Credit Cards.



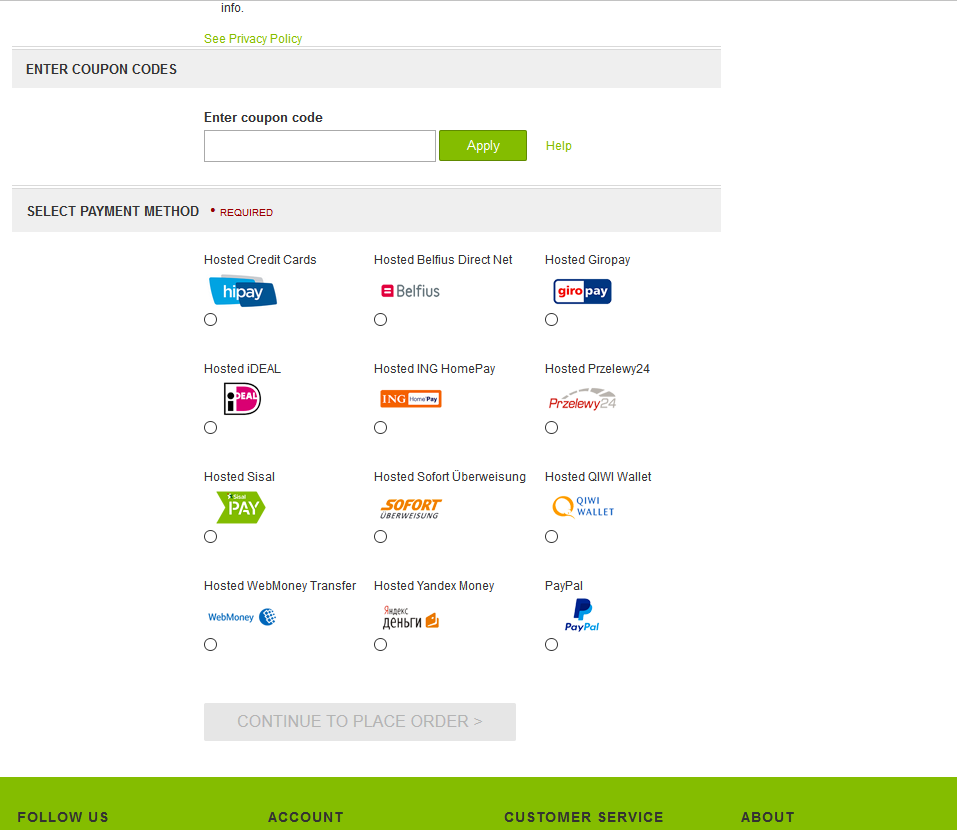
After the customer submits the order is redirected to a Payment page hosted and provided by HiPay, where the credit card form appears where the end-customer can enter the payment details:



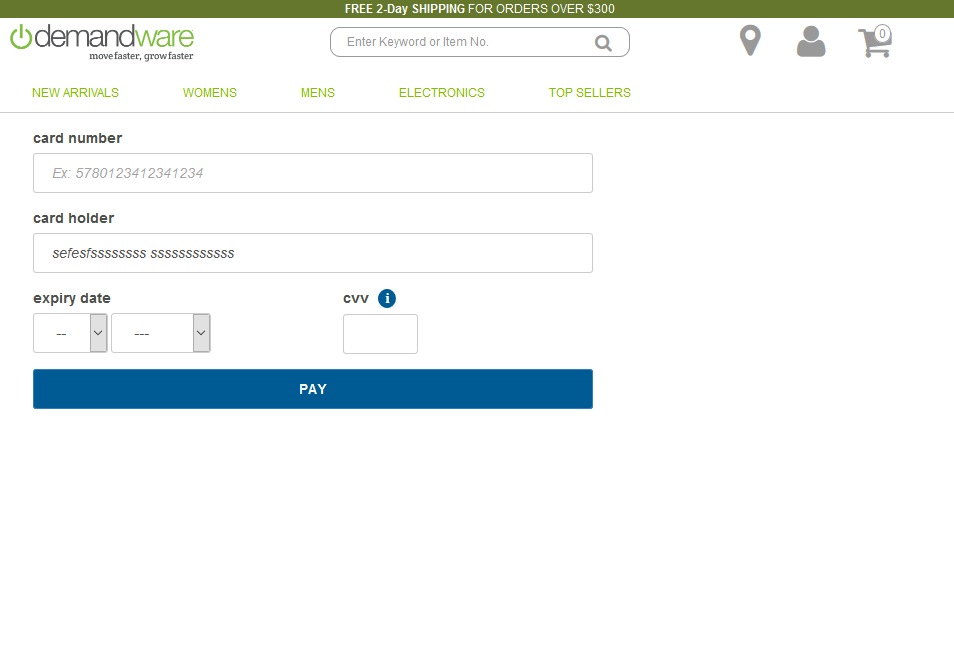
After completing payment with the HiPay, the customer is redirected back to the storefront. If the payment is successful or unknown, the customer is taken to the confirmation page.

### Hosted Merchant iFrame

On the billing page, the customer selects Hosted Credit Cards or other payment method.



After Submitting the order from the summary page, the customer is then taken to a Payment page, where the HiPay credit card form appears inside an iFame:



When payment is complete, the customer is directed to the confirmation page as usual. If the payment is successful or unknown, the customer is taken to the confirmation page. Is the payment was unsuccessful, the customer is given an error message on the summary page and asked to select a different payment method.

# Known Issues

The cartridge and its documentation will be in English only.

The HiPay integration cartridge is compatible with the latest SFCC API version, currently 17.8 and SG version 103.1.11. It might require updates and reviews for future versions and releases of Site Genesis.

# Release History

|  |  |  |
| --- | --- | --- |
| **Version** | **Date** | **Changes** |
| 1.0.0 | 02/11/2015 | Initial release |
| 1.0.1 | 7/29/2016 | HiPay Multi-account integration |
| 16.1.0 | 8/8/2016 | Updated compatibility with SG 16.7 |
| 18.1.0 | 29/1/2018 | Add HiPay One-Click payment setting (Api integration). Controller Version. |