**Worldpay Controller Integration**

Version 19.2.0

****

****

[1. SUMMARY 4](#_Toc18418281)

[2. Implementation Guide 6](#_Toc18418282)

[3.1 INSTALLATION 6](#_Toc18418284)

[3.1.1 Adding the Cartridges in Salesforce Commerce Cloud Studio 6](#_Toc18418285)

[3.1.2 IP Address 6](#_Toc18418286)

[3.1.3 Activating the Cartridges in Business Manager 6](#_Toc18418287)

[3.2 CONFIGURATION 7](#_Toc18418288)

[3.1.1 Importing Meta Data with a single Site Import 7](#_Toc18418289)

[3.2.1 Setting Worldpay Custom Site Preferences 11](#_Toc18418290)

[3.2.2 Manage Worldpay Payment Processor 14](#_Toc18418291)

[3.2.3 Managing Worldpay Payment Methods 15](#_Toc18418292)

[3.2.4 Managing Job 17](#_Toc18418293)

[3.2.6 Order Notifications 18](#_Toc18418294)

[3.2.7 Order Inquiry job 21](#_Toc18418295)

[3.2.8 Job Failure Mailer Service 22](#_Toc18418296)

[3.2.9 APM Lookup Service 23](#_Toc18418297)

[3.2.10 Multiple Merchant ID Support 23](#_Toc18418298)

[3.2.11 Brazil Integration 24](#_Toc18418299)

[3.2.12 Order CleanUp Batch Job 24](#_Toc18418300)

[3.2.13 Initiate Cancel Order Job 26](#_Toc18418301)

[3.2.14 Managing Content Asset 28](#_Toc18418302)

[3.2.15 Managing Service 28](#_Toc18418303)

[3.2.16 IDEAL APM Integration 29](#_Toc18418304)

[3.2.17 BMultiple Merchant ID Support Integration 30](#_Toc18418305)

[3.2.18 Look and Feel Customization for Worldpay Redirect Pages 30](#_Toc18418306)

[3.2.19 Hosted Payment Pages iframe or lightbox for Worldpay Redirect Payment Method 31](#_Toc18418307)

[3.2.20 Country Currency Mapping 33](#_Toc18418308)

[3.2.21 Country Currency and Locale Mapping for Klarna 33](#_Toc18418309)

[3.3 CUSTOM CODE 34](#_Toc18418310)

[3.3.1 Error Messaging 34](#_Toc18418311)

[3.3.2 Client Side Encryption 34](#_Toc18418312)

[3.3.3 Form File Changes 34](#_Toc18418313)

[3.4 CUSTOM CODE FOR CONTROLLERS 35](#_Toc18418314)

[3.4.1 Generic Section 35](#_Toc18418315)

[3.4.2 APM Lookup Support 59](#_Toc18418324)

[3.4.3 Brazil Integration 65](#_Toc18418325)

[3.4.4 Street Number Implementation 66](#_Toc18418326)

[3.4.5 Error Messaging 67](#_Toc18418327)

[3.4.6 Client Side Encryption 67](#_Toc18418328)

[3.4.7 Form File Changes 68](#_Toc18418329)

[3.4.8 Update shippingaddress.xml and billingaddress.xml 69](#_Toc18418330)

[3.4.9 Update the paymentinstrumentdetails.isml 69](#_Toc18418331)

[3.4.10 Update the locale.properties and countries.json 69](#_Toc18418332)

[4 PRODUCTION SETUP 69](#_Toc18418333)

[4.1.1 Production Service Setup 69](#_Toc18418338)

[5 OPERATIONS, MAINTAINENCE 70](#_Toc18418339)

[5.1.1 Order Payment Instrument Attributes 70](#_Toc18418341)

[5.1.2 Order Notification Custom Object 70](#_Toc18418342)

[5.1.3 Order Payment Instrument Attributes 71](#_Toc18418343)

[6 USER GUIDE 71](#_Toc18418344)

[6.1.1 Production Service Setup 71](#_Toc18418345)

[APPENDIX A: APM/Card mapping keys 71](#_Toc18418346)

[APPENDIX B: IDEAL Bank List 74](#_Toc18418347)

[APPENDIX C: Error Codes and Error Messages for Transaction Notification 74](#_Toc18418348)

[APPENDIX D: Order Notification and SFCC Order status mapping 75](#_Toc18418349)

[APPENDIX E: Error Codes and Error Messages 76](#_Toc18418350)

[APPENDIX F: Enabling and disabling worldpay integration 76](#_Toc18418351)

[APPENDIX G: CHECKLIST 77](#_Toc18418352)

# SUMMARY

This document provides technical overview and implementation details for each Worldpay service integrated within SFCC platform with SiteGenesis version. The Worldpay cartridges (int\_worldpay and int\_worldpay\_core) extends the functionality of reference eCommerce Storefront, enabling synchronous and asynchronous access to Worldpay payment transaction services listed below.

* Payment Methods:

1. Sofort
2. IDeal
3. Qiwi
4. Alipay
5. Mistercash
6. Yandex
7. Boleto
8. Paypal
9. China Union Pay
10. Enets
11. Giropay
12. Poli
13. Poli NZ
14. Konbini
15. Przelewy24
16. CashU
17. Sepa DD
18. Klarna
19. 3D/Non 3D Credit Cards (XML Direct with client side encryption and tokenzation features)
20. 3D/Non 3D Credit Cards (XML Redirect with tokenzation and hosted payment pages lightbox/iframe feature)

* Order Notification Job
* Order Enquiry Job
* Order Cancel Or Refund Job for mac Issue identified orders
* Order CleanUp Batch Job
* APM Look Up Service
* Multiple Merchant ID support
* Statement Narrative support
* Brazil Integration

# Implementation Guide



## INSTALLATION

### Adding the Cartridges in Salesforce Commerce Cloud Studio

Inside int\_worldpay folder open package.json and ensure the base path mentioned is correctly resolved, open the command prompt and run ‘npm install’ (assuming node.js (8.9.4 and above) is installed). Run the “npm run compile:js” followed by “npm run compile:scss”. Create the dw.json file at the location and Upload the code to the environment using the command ‘npm run uploadCartridge’. Please go to the **int\_worldpay:int\_worldpay\_core** cartridge and upload the code by configuring the various parameters in the sample dw.json.

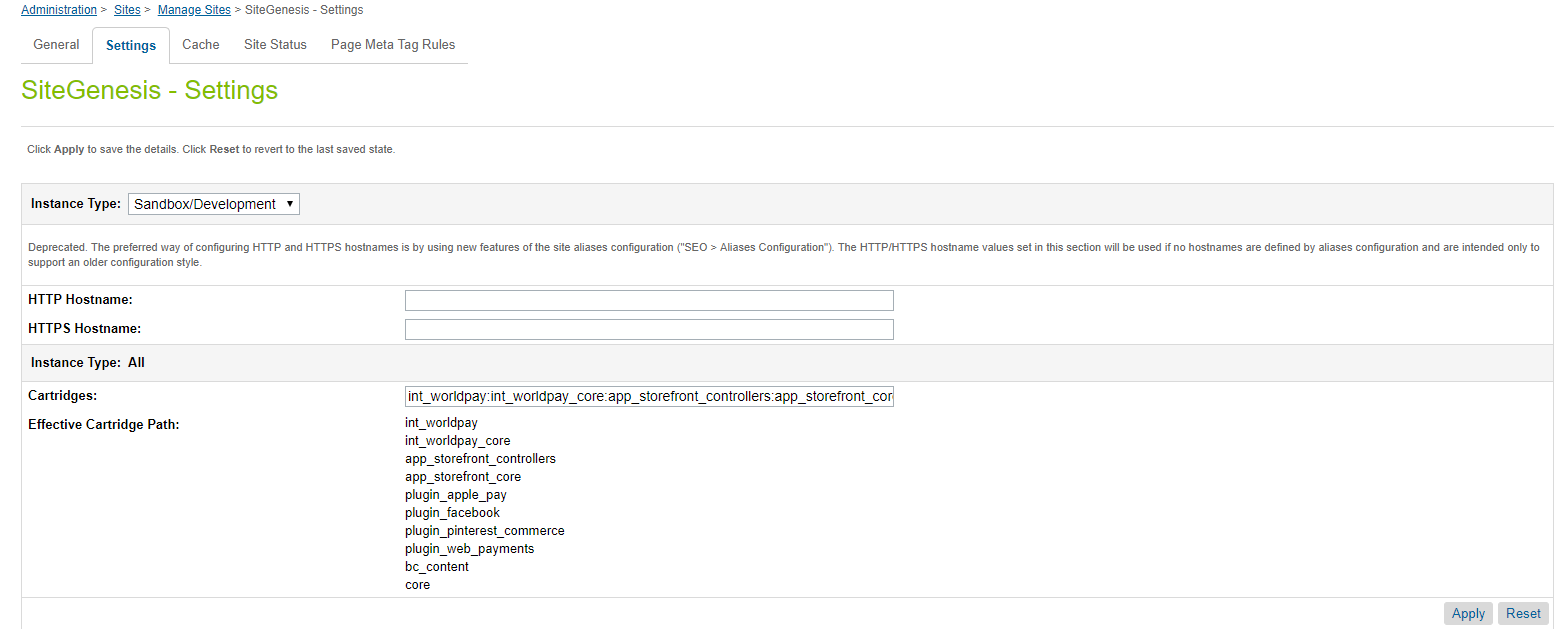
### IP Address

Please provide outgoing IP of your client in order to white list your IP otherwise some of the attempts to make payment will be marked as fraud.

### Activating the Cartridges in Business Manager

Before the Worldpay functionality can become available to Reference Architecture, the cartridges needs 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\_worldpay:int\_worldpay\_core:app\_storefront\_controllers:app\_storefront\_core**”
5. Click *Apply*.
6. To activate the cartridge for the Sandbox instances repeat steps **4** and **5** after selecting the appropriate instance from the *Instance Type* dropdown menu.
7. Repeat steps **3** to **6** for each site that is to use Worldpay.
8. To run the Job in worldpay cartridge,Navigate to *Administration* 🡪 *Sites* 🡪 *Manage Sites*.
9. Go to “Manage the **Business Manager** site”
10. In the textbox *Cartridges* add : “**int\_worldpay:int\_worldpay\_core:app\_storefront\_controllers:app\_storefront\_core**”



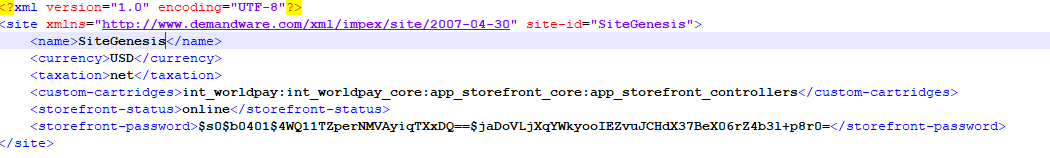
## CONFIGURATION

This chapter will guide the user to configure the cartridge in Business Manager.

### Importing Meta Data with a single Site Import

All import files are in the import folder (metadata) within cartridge installation pack. To import all necessary Worldpay settings,

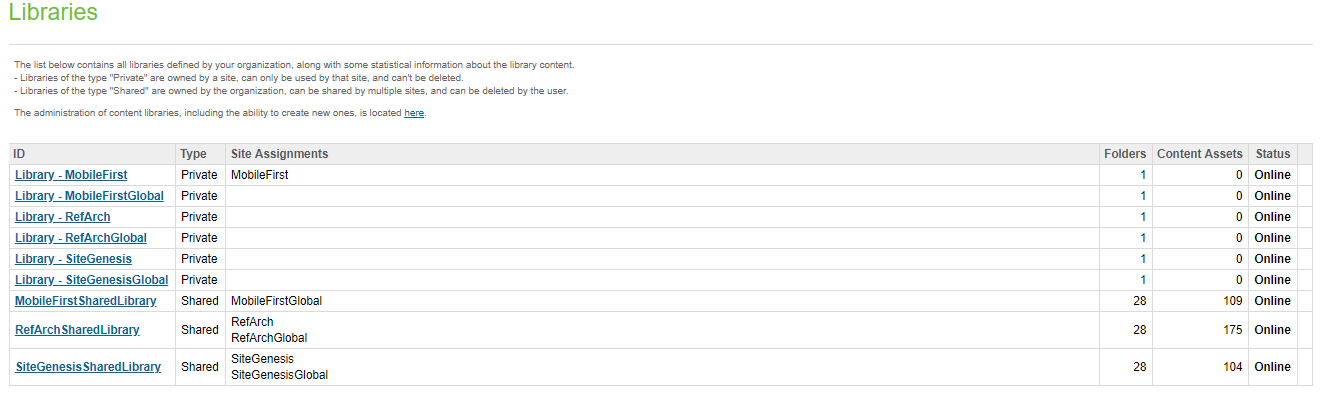
* + Open the metadata folder from the repository structure 🡪 Go to sites folder 🡪 change the name of the site according to the site id in your sandbox on which you want to perform the site import (eg. If your site id is worldpayDemo, change the name from RefArch to worldpayDemo. As My sanbox ID is SiteGenesis, I am changing it to SiteGenesis). 🡪 Open site.xml and change the references of RefArch to your site id (in this case worldpayDemo). As my controller site id is SiteGenesis, I am changing the reference of RefArch to SiteGenesis.



* + Open the metadata folder from the repository structure 🡪 open jobs.xml 🡪 change the references of RefArch to SiteGenesis(in my sandbox) or to your site id (in this case worldpayDemo)
  + Ensure content assets in the import has appropriate library Id of the site where XML to be imported

For this, In the business manger

* + - Go to **Merchant Tools > Content > Libraries**
    - Checkthe library ID assignment to your site (here for SiteGenesis the library assignment is SiteGenesisSharedLibrary)



* + - Open the metadata folder from the repository structure **>** open libraries **>** change the folder name according to the library id to which your site is assigned (Here in the BM we saw that for SiteGenesis, assigned Library is SiteGenesisSharedLibrary) > open library.xml and change the reference of SiteGenesisSharedLibrary, if needed (according to your site assignment)
  + ZIP the metadata folder from the repository structure,
  + Log in to the Business Manager and navigate to *Administration* 🡪 *Site Development* 🡪 *Site Import & Export*, and upload the ***zipped*** folder (Zipped Metadata Folder) using the upload button and, finally go back and use the import button to import the file.
  + After a successful import, the entire configuration requirements needed for worldpay is available to you is according to your Worldpay account data.
  + Also, verify the **Order level attributes** in BM (Site -> Ordering -> Order) open any order and navigate to the Attributes tab. It should be as below: -

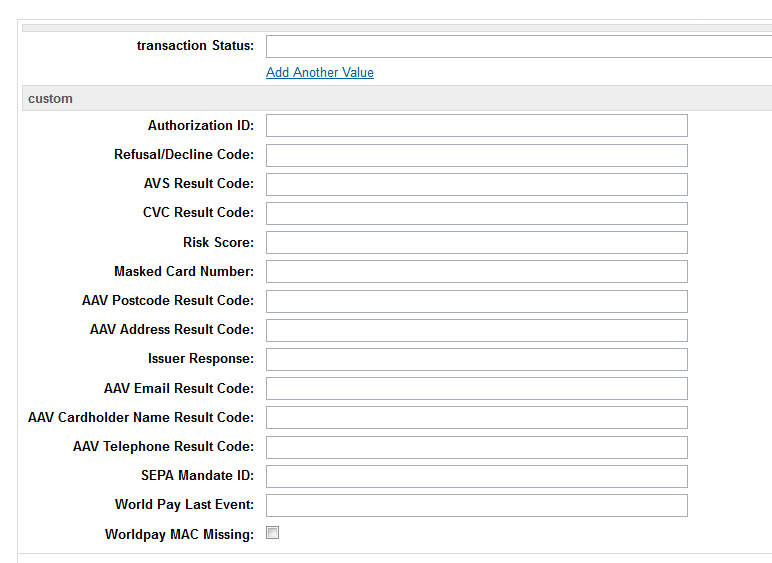
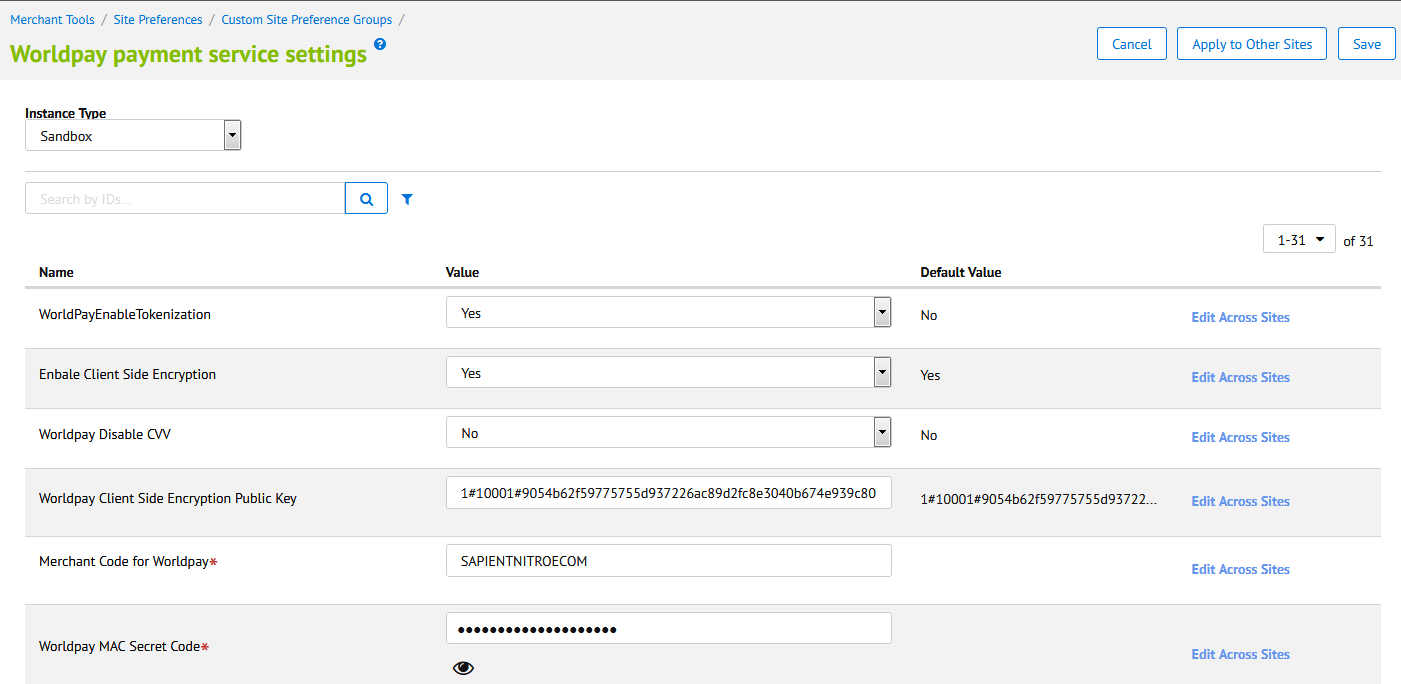
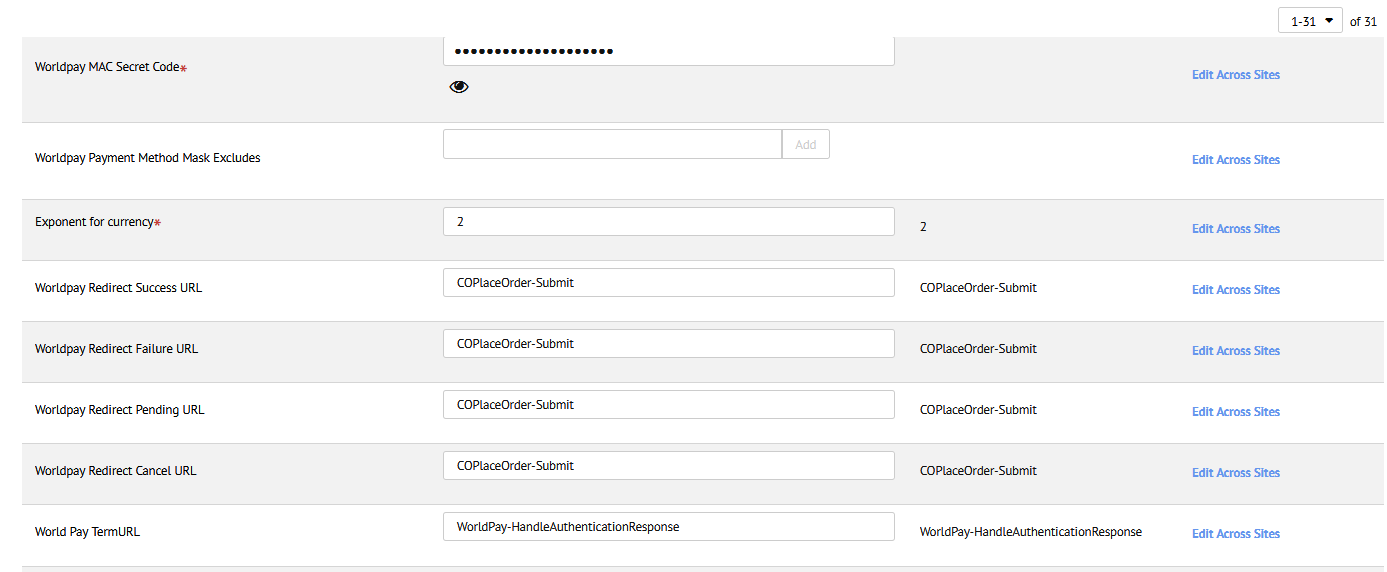
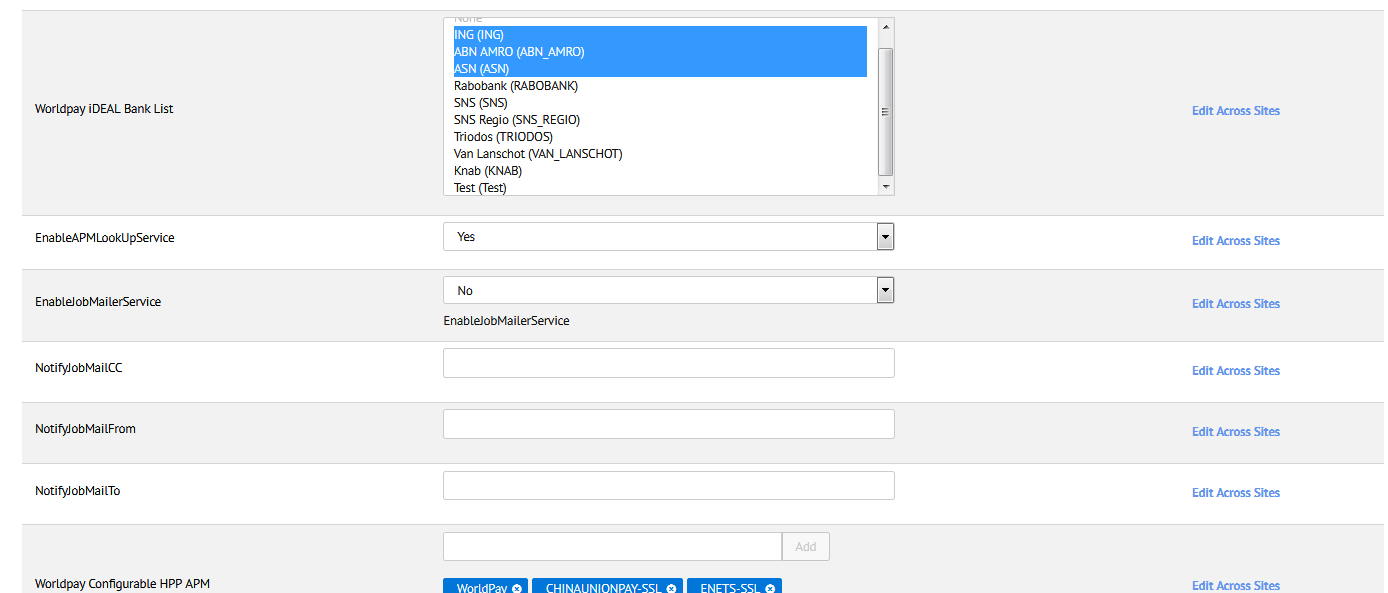


Fig: 2.1

Custom Preferences Screenshot







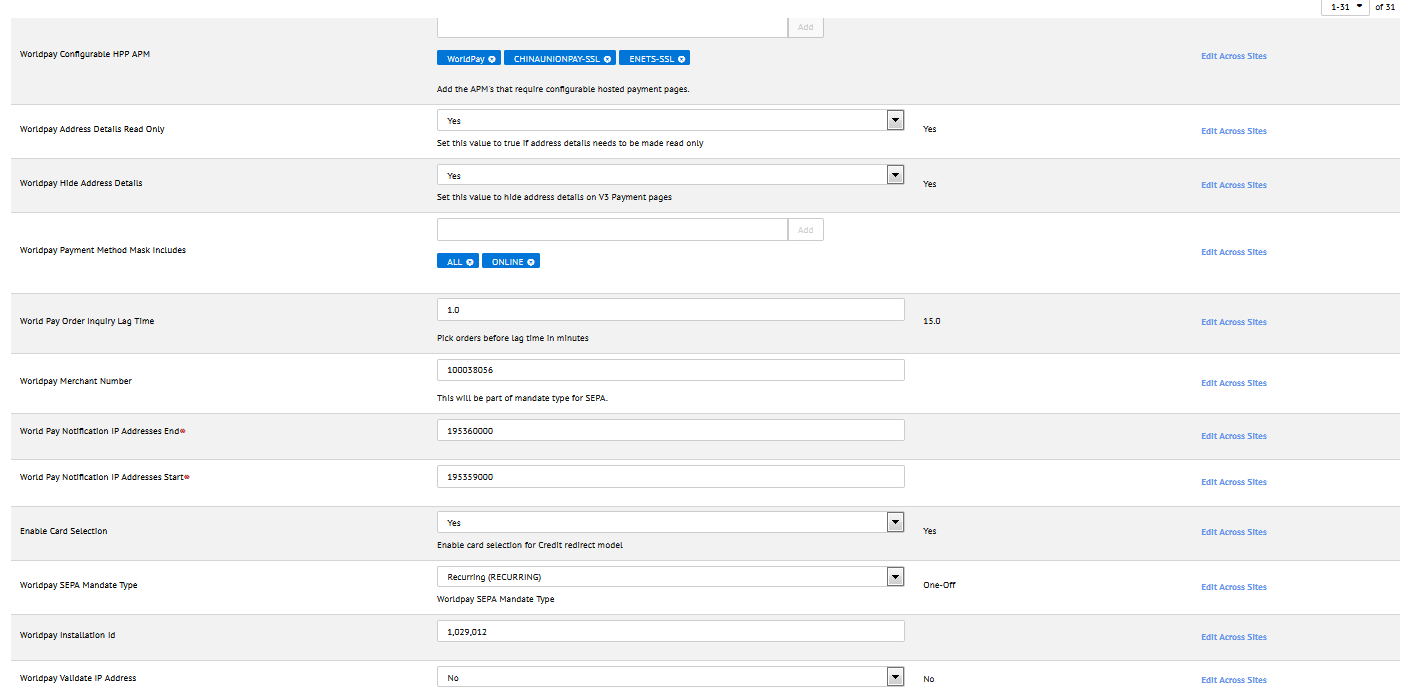


Fig: 2.2

### Setting Worldpay Custom Site Preferences

In Business Manager, navigate to the *Site🡪* Merchant Tools🡪 *Site Preferences* 🡪 *Custom Preferences*. A custom site preference group with the ID Worldpay is available. Please select it and edit the attributes according to your Worldpay account data and the data shown in figure 2.2. In addition to the ones supplied by Worldpay,fill out the following properties with the values listed against them:

* *Worldpay Redirect SuccessURL: COPlaceOrder-Submit*
* *Worldpay Redirect PendingURL: COPlaceOrder-Submit*
* *Worldpay Redirect FailureURL: COPlaceOrder-Submit*
* *Worldpay Redirect CancelURL: COPlaceOrder-Submit*
* *Worldpay TermURL: Worldpay-HandleAuthenticationResponse*

\*\* The fields highlighted in blue in the below table are mandatory fields

|  |  |  |
| --- | --- | --- |
| **Site Preference** | **Description** | **Default Values** |
| **Merchant Code for Worldpay** | This field represents the specific Merchant code value. | Merchant Specific - Provided by Worldpay |
| **Worldpay MAC Secret Code** | The field represents the MAC secret code that configurable in Worldpay Merchant Interface. | Merchant Specific |
| **Worldpay Payment Method Mask Includes** | This field represents all the Cards and APM’s supported and will be shown on Hosted Payment Page. | Merchant Specific - Add the Cards /APMs supported on Hosted Payment Pages.  (Refer to Appendix A for values) |
| **Worldpay Payment Method Mask Excludes** | This field represents all the Cards and APM’s that are needs to be excluded and will not be shown on Hosted Payment Page. | Merchant Specific |
| **Exponent for currency** | This field represents the decimal places supported in the Order Amount. | Merchant Specific  (Default Value : 2) |
| **Worldpay Redirect Success URL** | This field represents the Success URL, that the user will be redirected to incase of a successful transaction in APM’s or Credit Card Redirect. | Default Value : COPlaceOrder-Submit |
| **Worldpay Redirect Failure URL** | This field represents the Failure URL, that the user will be redirected to incase of a Failure transaction in APM’s or Credit Card Redirect. | Default Value : COPlaceOrder-Submit |
| **Worldpay Redirect Pending URL** | This field represents the Pending URL, that the user will be redirected to incase of a Pending or Pending-Open transaction in APM’s or Credit Card Redirect. | Default Value : COPlaceOrder-Submit |
| **Worldpay Redirect Cancel URL** | This field represents the Cancel URL, that the user will be redirected to incase of a cancelled transaction in APM’s or Credit Card Redirect. | Default Value : COPlaceOrder-Submit |
| **Worldpay TermURL** | This field represents the Term URL on which the user will be redirected in case of 3D secure. | Worldpay-HandleAuthenticationResponse |
| **Worldpay iDEAL Bank List** | This field represents the Bank’s supported for iDEAL APM. The merchant can select the Banks that needs to be displayed on the Merchant Site. | Merchant Specific - Select the ones to be supported. |
| **Worldpay Installation Id** | This field represents the installation ID provided by Worldpay, it will determine the look and feel of the Hosted Payment Pages. | Merchant Specific - Provided by Worldpay |
| **Worldpay Address Details Read Only** | This field is set to true if address details needs to be made read only | Merchant Specific  (Default Value : true) |
| **Worldpay Hide Address Details** | This field is set to true to hide address details on V3 Payment pages | Merchant Specific  (Default Value : true) |
| **EnableAPMLookUpService** | This field can be used to Turn On/Off the Look Up service, based upon the Merchant requirement. | Merchant Specific |
| **EnableJobMailerService** | This field can be used to Turn On/Off the Email Service. The Emails are triggered in-case the cartridge logic fails to handle the Order Notification/Enquiry and update its status in Order transaction field. | Merchant Specific |
| **NotifyJobMailCC** | This field represents the Email address of the people to kept in CC. | Merchant Specific: Multiple Emails can be added separated by semicolon. |
| **NotifyJobMailFrom** | This field represents the Email address of the people to be in From. | Merchant Specific: Multiple Emails can be added separated by semicolon. |
| **NotifyJobMailTo** | This field represents the Email address of the people to be in To. | Merchant Specific: Multiple Emails can be added separated by semicolon. |
| **WorldpayEnable**  **Tokenization** | This field is set to true if merchant wants to enable tokenization | Merchant Specific(Default value : false) |
| **Worldpay Disable CVV** | This field is set to true if merchant want to disable CVV check for Saved Card with token and new card.  We are suppressing the cvv field in the request of the order placed with saved card having token and new card. | Merchant Specific(Default value : false) |
| **Enable Client Side Encryption** | This field is set to true if merchant wants to enable Client Side Encryption | Merchant Specific(Default value : false) |
| **Worldpay Client Side Encryption Public Key** | This field represents the public key provided by Worldpay, used to calculate mac value of an order | Merchant Specific |
| **Worldpay Order Inquiry Lag Time** | This field represents the maximum time(in minutes) for which orders to be look-up in Worldpay for Order Enquiry or order cancel/refund Job | Merchant Specific(Default value : 15) |
| **Worldpay Configurable HPP APM** | This field represents the APM’s for which the site supports configurable Hosted Payment Pages (Look & Feel configuration). | Merchant Specific - Add the APM's that support configurable Hosted Payment Pages.  Ensure that the **Worldpay** **InstallationID** set if some value is added to this Preference. |
| **Enable Card Selection** | This field represents the Merchants choice to enable /disable Card selection on the Site. | Merchant Specific: Enable /Disable Card selection on the Site. |
| **Worldpay Merchant Number** | This field represents a Merchant specific number provided by Worldpay; this is sent as a part of request for SEPA. | Merchant Specific - Provided by Worldpay |
| **Worldpay SEPA Mandate Type** | This field represents the Mandate type to be sent in the SEPA request. (Default value ONE-OFF) | Merchant Specific – This represents the mandate type to be set for SEPA request. The merchant can choose to set it either as ONE-OFF or RECURRING. |
| **Worldpay Notification IP Addresses End** | This field represents the Worldpay IP Address range which send notification, last IP Address in range | Merchant Specific - Provided by Worldpay |
| **Worldpay Notification IP Addresses Start** | This field represents the Worldpay IP Address range which send notification, first IP Address in range | Merchant Specific - Provided by Worldpay |
| **Worldpay validate IP Address** | This field is set to true if merchant want to validate IP address | Merchant Specific(Default value : false) |
| **CaptureServiceTrackingID** | This field is set to true if merchant want to Validates the order id and token upon match it will proceed for capture service initiation with tracking ID | Merchant Specific(Default value : false) |

### Manage Worldpay Payment Processor

In Business Manager navigate to ***SiteGenesis Site🡪Merchant Tools* *🡪 Ordering 🡪 Payment Processors*.**

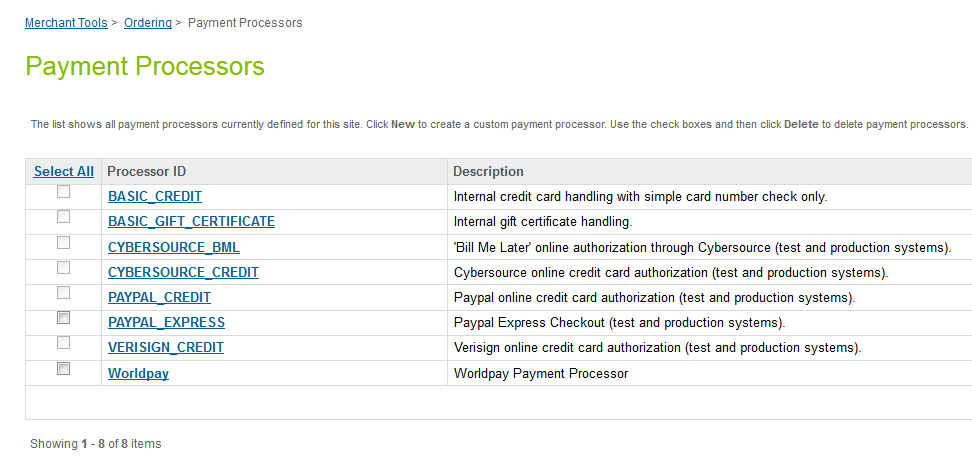


Fig: 2.3

### Managing Worldpay Payment Methods

In the integration package, a payment method definition is provided in the file that you have alredy imported

1. Navigate to🡪 Merchant Tools 🡪 Ordering🡪 Payment Methods.
2. Enable the Payment Method under **Ordering -> Payment Methods section** that needs to be activated. Repeat these steps for all the Payment Methods that are required to be supported on the site. (Refer to Fig: 2.4)

Set below custom attributes of Payment method as provided by merchant: (Refer to Fig: 2.5)

|  |  |
| --- | --- |
| **Custom Attribute** | **Description** |
| Merchant Code | This field represents the hex code provided by merchant |
| User Name | This field represents the username provided by merchant |
| Type | Select the Payment Method Type to DIRECT/REDIRECT as per the one that needs to be supported on the merchant site. (Refer to Fig: 2.5) |
| Password | This field represents the password provided by merchant |
| Days after which Order expires | Configure the number of days after which the Order should be cleaned up |
| SEPA mandate number | This will be part of mandate type for SEPA. |
| HPP CustomOptions JSON | This would only be populated in case of payment method as “Credit Card Redirect” where merchant defines JSON which determines the type as iframe/lightbox in which Hosted Payment Page appears on Merchant site.  Default Value is blank represent Worldpay redirection page appears |

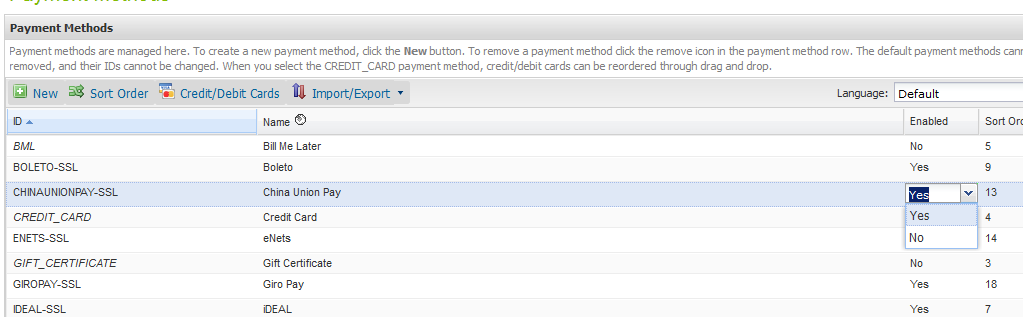
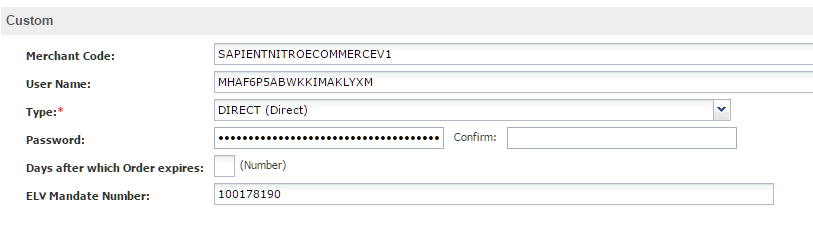


Fig: 2.4

Fig: 2.5

* Once the payment methods are added and enabled successfully, update credit or debit card details as provided by merchant under **Payment method-> Credit/Debit Cards** section.Enable the credit/debit card under **Credit/Debit Cards Section** that needs to be activated. Ensure custom attribute named “Worldpay Card Type” is populated for all active cards with corresponding Worldpay type like for VISA it is “VISA-SSL”. Repeat these steps for all the cards that are required to be supported on the site. (Refer to Fig: 2.6)

Set below card details of Payment method as provided by merchant. All these fields are optional (Refer to Fig: 2.6)

|  |  |
| --- | --- |
| **Card Attribute** | **Description** |
| Security Code Length | This field represents maximum length of security code for card type |
| Card Number Verification | This field represents maximum length of CVV for card type |
| Card Number Length | This field represents maximum length of card number(range between 13-16) for card type |
| Checksum Verification | This field is set to True if merchant wants to verify the card details |
| Worldpay Card Type | This field represents the card type in Worldpay like for VISA it is “VISA-SSL” |

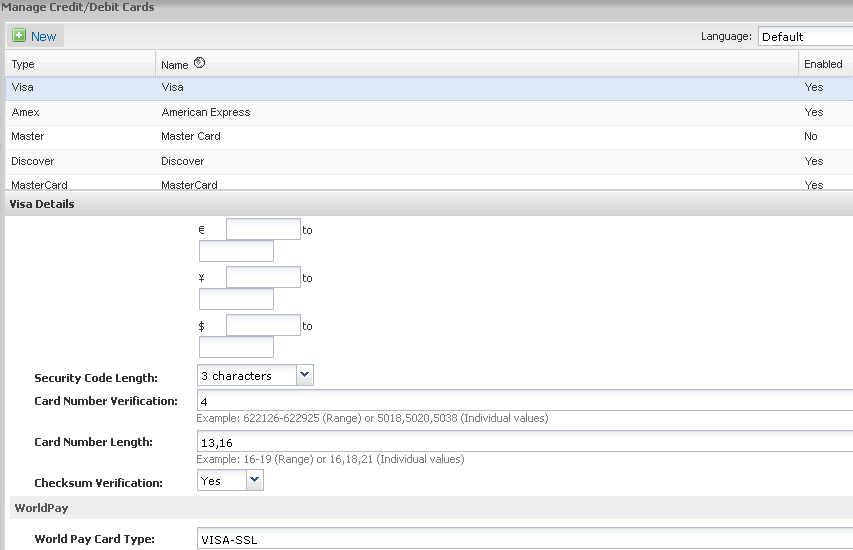
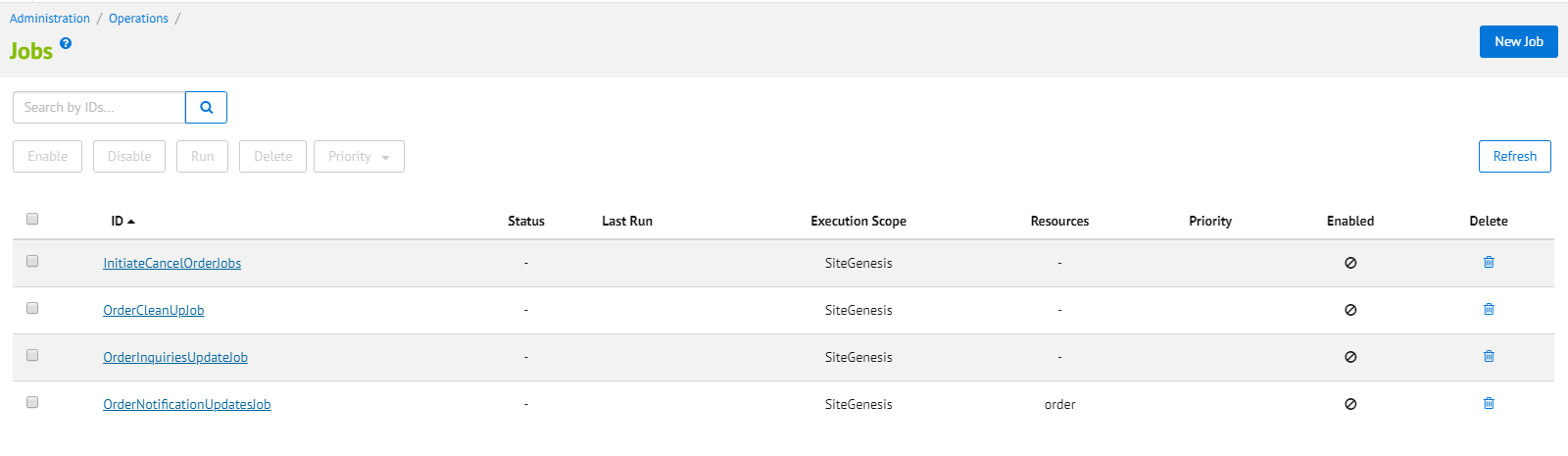


Fig: 2.6

### Managing Job

In the integration package, a job definition is provided in the ***import zip*** file (jobs.xml).

1. Make sure to associate the jobs to respective merchant site.



Enable the jobs to be active, apply the configurations setting of scheduling into General and Sites sections one by one.

### 3.2.6 Order Notifications

The int\_worldpay\_core cartridge provides a service to update all the SFCC Order status according to status changes of individual payments in Worldpay. (PUSH mechanism)

**Worldpay-Notify** is the secure route function exposed to Worldpay to receive Order Notifications i.e. a HTTPS message whenever a payment status changed at Worldpay. Following URL (**https://mechantsandbox/Worldpay-Notify**) has been exposed to Worldpay to send the notifications. SFCC will send an HTTP 200 [OK] response to Worldpay to acknowledge the receipt of notification. Corresponding to each Order Notification received a custom object is created with the Order Number and Order Notification XML captured. A SFCC Job is then run (manually or scheduled), to process those custom objects and update the Orders corresponding to each Order Notification.

If the Worldpay system does not receive an [OK] in reply to an HTTPS order notification it waits for approximately an hour to deliver the notification again. Then it will resend the order notification approximately every five minutes for a week until it is acknowledged. Worldpay places order notifications in a queue and the first notification in the queue will be resent until it is acknowledged. If acknowledged, or if after a week still no [OK] is returned, the retry mechanism stops sending the message and proceeds with the next order notification from the queue.

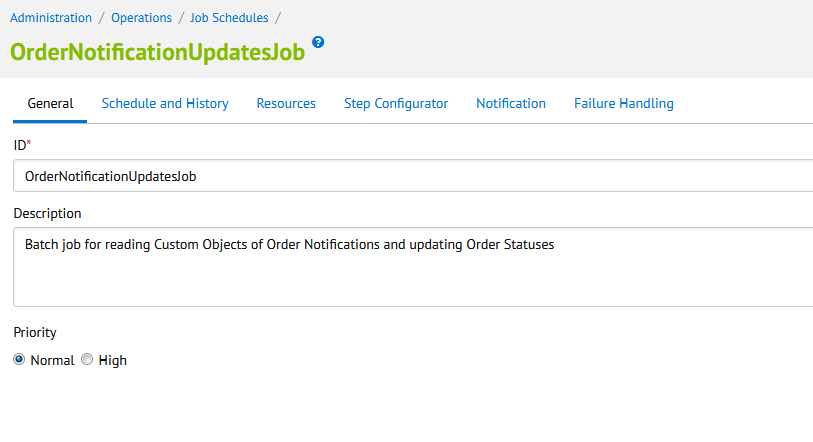
This URL (**https://mechantsandbox/Worldpay-Notify**) needs to be updated in Worldpay Admin console by the merchant to receive the Order Notifications. The order notifications are captured from Worldpay Payment Service when a payment reaches one of the following statuses:

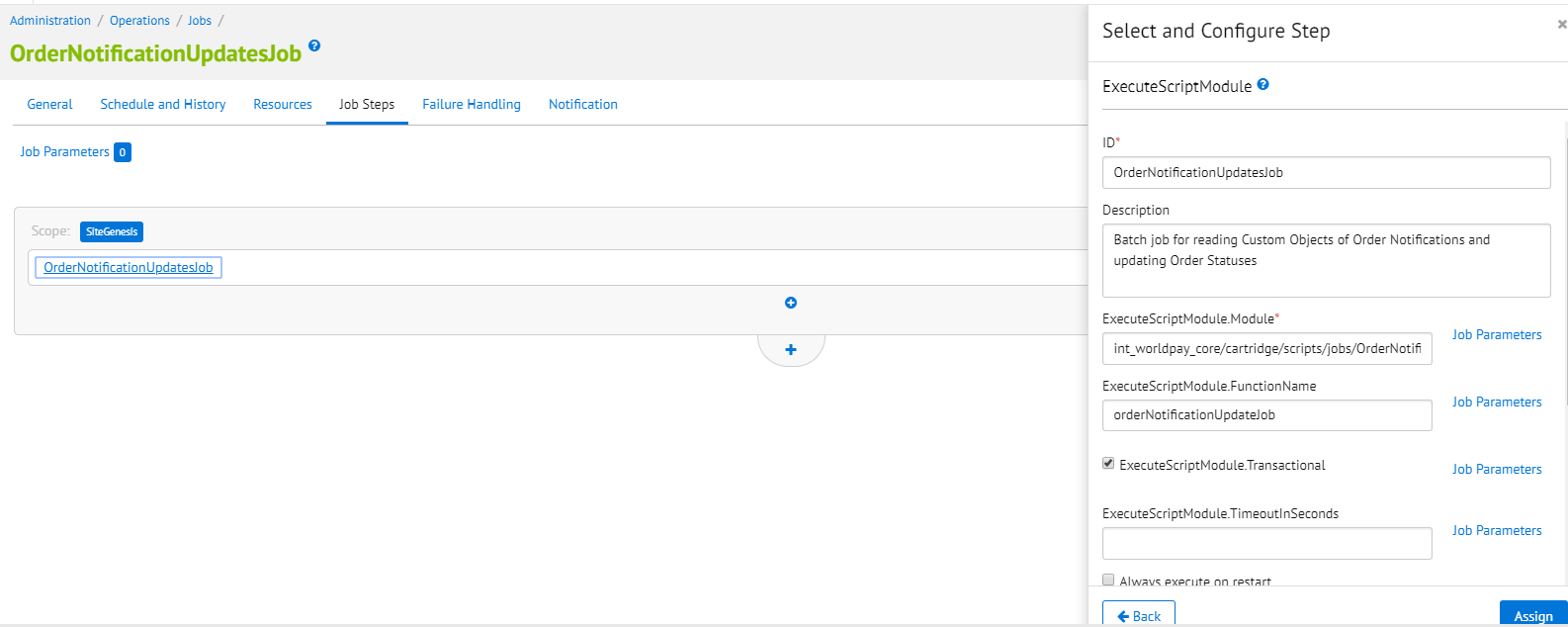
* AUTHORISED
* CANCELLED
* CAPTURED
* EXPIRED
* SENT\_FOR\_REFUND
* REFUSED
* SETTLED
* INFORMATION\_REQUESTED
* CHARGED\_BACK
* POST AUTH CANCELLED

#### Order Notification job

The OrderNotificationUpdatesJob is a batch job which can be run manually or scheduled at a specific interval of time -i.e., 2 mins, 5 mins. Each time the job runs, it will pick all the custom objects created because of the prior received Order Notifications, sorted by the creation time. The job read the custom objects one by one, will update the statuses in Business Manager (Order Status/Payment Status/Confirmation Status) depending on the notification status in xml string and update the token details in order/customer card and finally remove the custom object post processing from SFCC.

Refer below Step configuration section in job to execute script and their functions:





|  |  |
| --- | --- |
| Fields | Fields data |
| ID | OrderNotificationUpdatesJob |
| Description | Batch job for reading Custom Objects of Order Notifications and updating Order Statuses |
| ExecuteScriptModule.Module | int\_worldpay\_core/cartridge/scripts/jobs/OrderNotificationUpdatesJob.js |
| ExecuteScriptModule.FunctionName | orderNotificationUpdateJob |
| ExecuteScriptModule.Transactional | Enabled |

#### Error Handling for Order Notification Job

Please refer to[Appendix C](#_APPENDIX_C:_Error) for error codes and error messages for Order Notifications

#### Transaction payment status handling for Order Notification Job

Please refer to [Appendix D](#_APPENDIX_D:_Order) for mapping of different status on receipt of different order notification from WP into BM.

#### Notification update Service

[The Cartridge provides feature to expose the history of the Payment Transaction change (Order Notifications) for a particular Order in SFCC business manager.](https:// /WorldPay-GetNotificationUpdates  is%20the%20pipeline%20exposed%20to%20get%20the%20History%20of%20Notifications%20received%20by%20particular%20Order. )A JSON object is returned by the notifications updates service. This service is provided in 2 flavors, depending upon the need to fetch the entire status history or just the latest status. If order no or status history is not found an error JSON is returned.

**[Worldpay-GetNotificationUpdates](https://merchantSandbox/WorldPay-GetNotificationUpdates  is the pipeline exposed to get the History of Notifications received by particular Order.)** [is the route exposed to get the History of Notifications received by particular Order.](https://merchantSandbox/WorldPay-GetNotificationUpdates  is the pipeline exposed to get the History of Notifications received by particular Order.)

This service takes 2 parameters as Http Parameters,

1. orderNo
2. allupdates

If ‘allupdates’ is true, all notifications received for the order will be returned as JSON response.

If ‘allupdates’ is false, only the latest notifications received for the order will be returned as JSON response.

All values for ‘allupdates’ i.e. true, TRUE, True will be considered as true, any value other than these will be considered as false.

Example for the JSON response when ‘allupdates’ = true is as below:

**RESPONSE:**

{"statusList":

[{"status":"AUTHORISED:Mon Nov 10 2014 08:11:24 GMT-0000 (GMT)"},

{"status":"CANCELLED:Mon Nov 11 2014 08:11:24 GMT-0000 (GMT)"}

]

}

Example for the JSON response when ‘allupdates’ = false is as below:

**RESPONSE:**

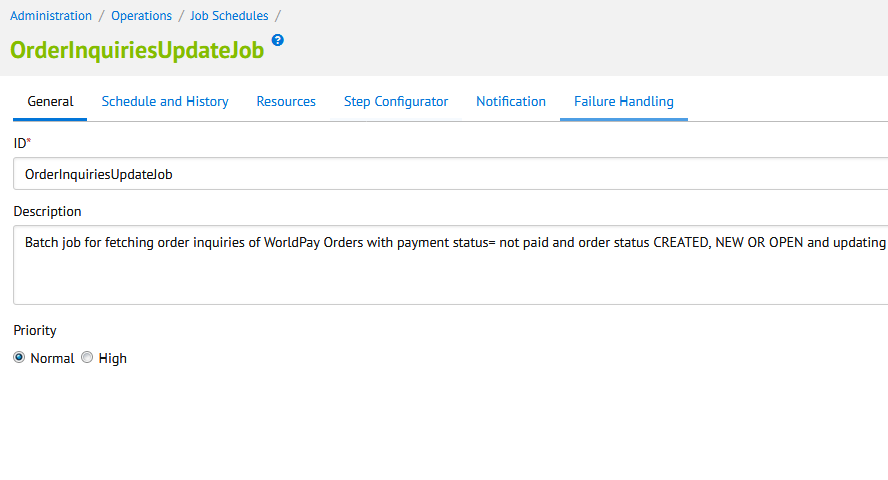
{"latestStatus":

[{"Status":"AUTHORISED:Mon Nov 10 2014 08:11:24 GMT-0000 (GMT)"}]

}

### Order Inquiry job

The OrderInquiryUpdatesJob is a batch job which can be run manually or scheduled at a specific interval of time -i.e., 2 mins, 5 mins. Each time the job runs, it will pick all the orders with status as CREATED/ NEW/OPEN and payment status as NOT PAID and order placed at least 15 mins back (configurable custom preference in business manager) by the creation time. The job iterates the orders and make order inquiry service call, will update the statuses in Business Manager (Order Status/Payment Status/Confirmation Status) depending on the response status in xml response of order enquiry and update the token details in order/customer card



Refer below Step configuration section in job to execute script and their functions

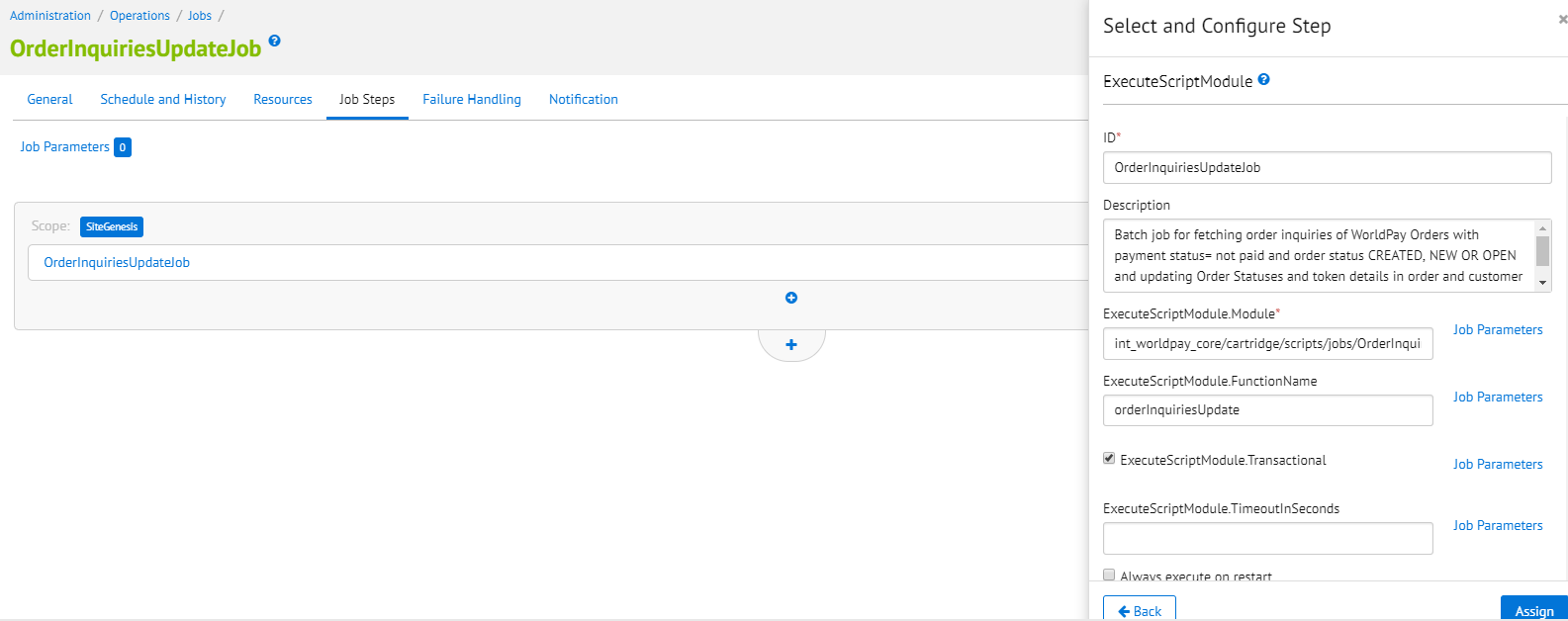


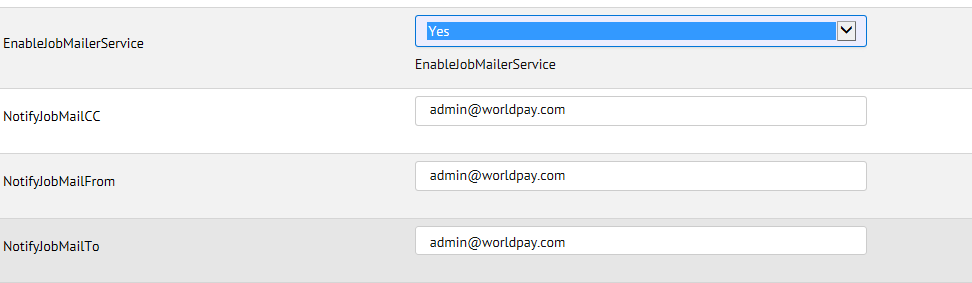
Table for Reference:

|  |  |
| --- | --- |
| Fields | Fields data |
| ID | OrderInquiriesUpdateJob |
| Description | Batch job for fetching order inquiries of Worldpay Orders with payment status= not paid and order status CREATED, NEW OR OPEN and updating Order Statuses and token details in order and customer saved payment instruments |
| ExecuteScriptModule.Module | int\_worldpay\_core/cartridge/scripts/jobs/OrderInquiriesUpdateJob.js |
| ExecuteScriptModule.FunctionName | orderInquiriesUpdate |
| ExecuteScriptModule.Transactional | Enabled |

### Job Failure Mailer Service

The cartridge also provides a feature to send Emails to the registered Emails in case of failure of any Order Notification update. On receipt of any Order Notification a custom object is created in SFCC with the Order Number and the Order Notification XML, if for any reason the XML captured is incorrect of some error occurs while the execution of the job an email is triggered.

To enable job mailer service to notify BM admin for any failure happens while processing the Order Notification Batch Job, the site preference EnableJobMailerService needs to be select **Yes** from the dropdown. Also, the below site preferences needs to be set



### APM Lookup Service

To enable APM look up service, the site preference **EnableAPMLookUpService** needs to be selected Yes from dropdown.

|  |  |
| --- | --- |
|  |  |
| **EnableAPMLookUpService** when enabled | 1. This service works for merchant code configured in site preference, which return APMs which are enabled for respective merchant code. 2. All active Payments methods which have different merchant code configured in payment method section will be displayed 3. All active payment methods other than Worldpay processor will be displayed |
| **EnableAPMLookUpService** when disabled | 1. Loookup service will not be executed 2. All active payment methods will be displayed |

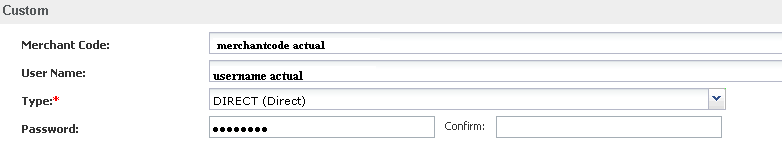
[Note: If merchant supports multiple merchant Ids through payment methods in that case APM Look service may not work so, it should not be enabled in site preference.]

### Multiple Merchant ID Support

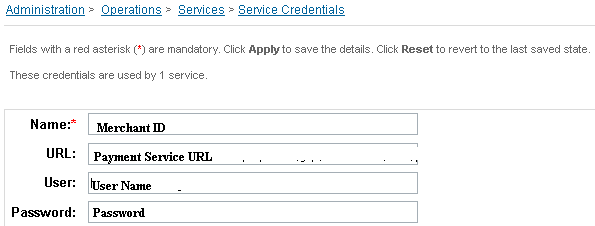
Worldpay cartridge also supports a mechanism to provide separate Merchant ID’s at APM level. This is primarily required for Merchants that need to configure different Merchant ID for financial purpose or if a specific APM is of a different version (to support both V1 and V3 APM’s on a single Merchant Site).

This is achieved, by setting a different MID and its details (Username & password) in the Payment Methods section in Business Manager. If merchant ID, username and password present at payment method level then would get preference compare to merchant ID in site preference, username and password from service credentials. If they are not configured at payment method level then merchant ID (in site preference), username and password provided in service credential will be used in request.

Payment Method Merchant ID Support Settings



Service Credential Settings when service merchant settings to be picked up

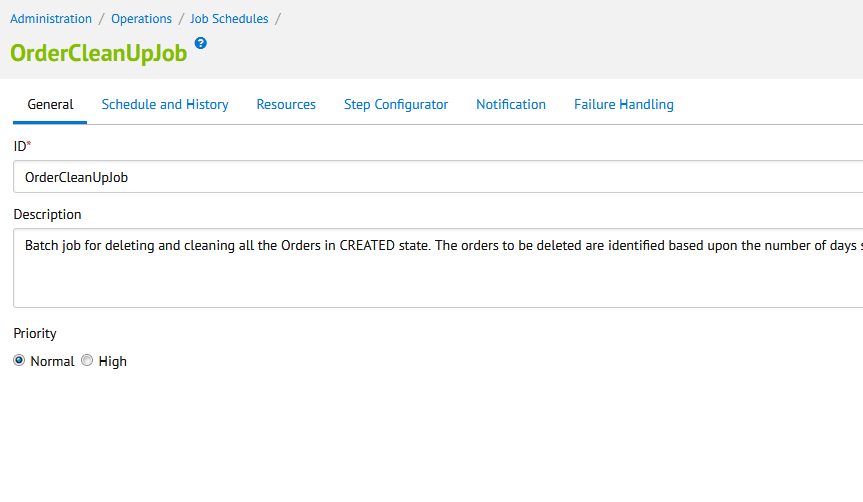


### Brazil Integration

The Worldpay cartridge provides support to Credit Cards and supported APM’s (BOLETO) in Brazil. The implementation ensures that all the additional information (Example: CPF/CNJP and Installment Number) that are required by the payment service provider in Brazil are also captured.

### Order CleanUp Batch Job

The Worldpay cartridge provides support to a batch job that will mark all the orders with status as **CREATED** and aged by a defined time interval. The time interval field is Business Manager configurable and also configurable at Payment Method level. A Batch Job configured in Business Manager shall run (as per the set schedule) and check the orders with order status **CREATED** which are old to a defined interval and will mark those orders in **FAILED** status in demandware system. This ensures that all the inventory transactions and coupon redemptions associated with the Order are rolled back. Job can be Enabled/Disabled or from Business Manager itself. Email will be sent to BM configured recipients once the job run with Job run status.



Refer below Step configuration section in order to execute script and their functions

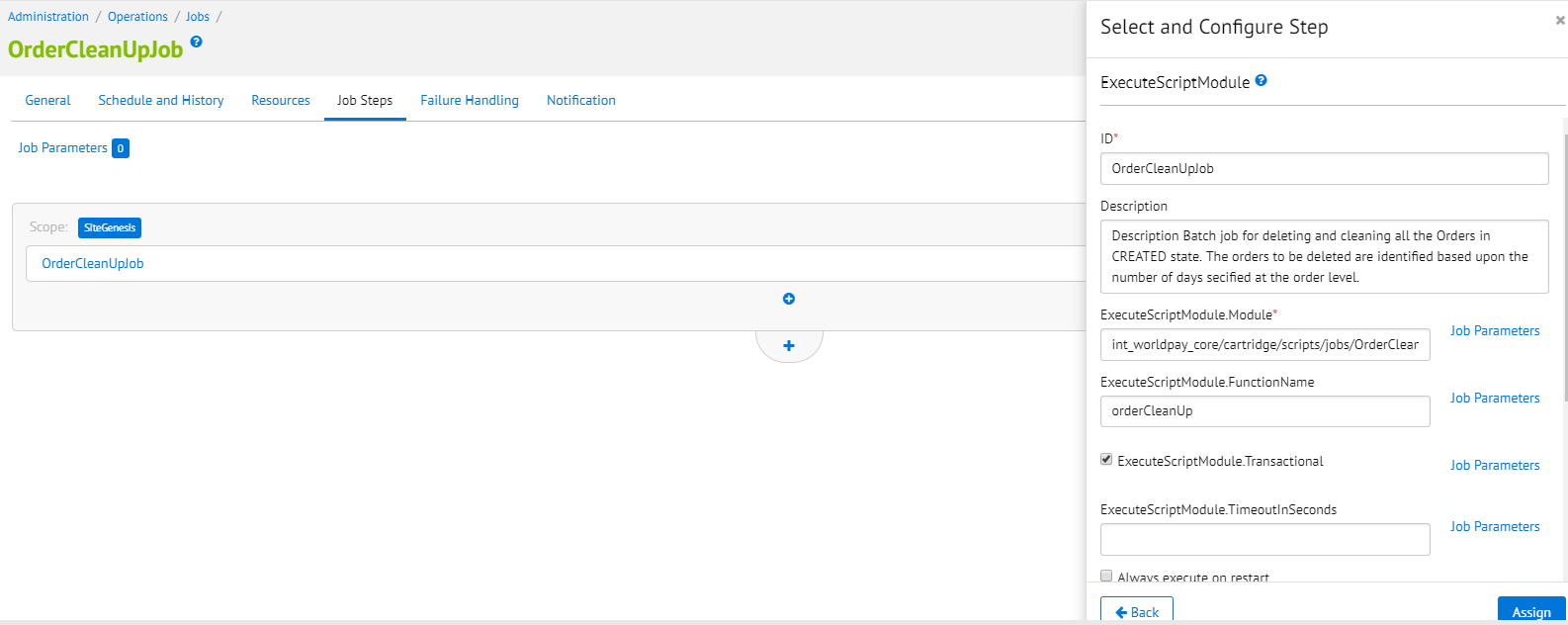
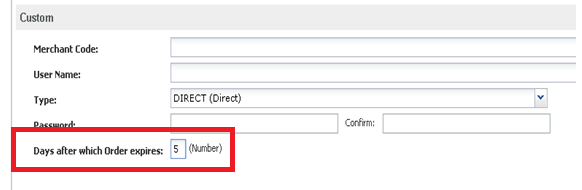


Table for Reference:

|  |  |
| --- | --- |
| Attributes Fields | Attributes |
| ID | OrderCleanUpJob |
| Description | Description Batch job for deleting and cleaning all the Orders in CREATED state. The orders to be deleted are identified based upon the number of days secified at the order level. |
| ExecuteScriptModule.Module | int\_worldpay\_core/cartridge/scripts/jobs/OrderCleanUpJob.js |
| ExecuteScriptModule.FunctionName | orderCleanUp |
| ExecuteScriptModule.Transactional | Enabled |

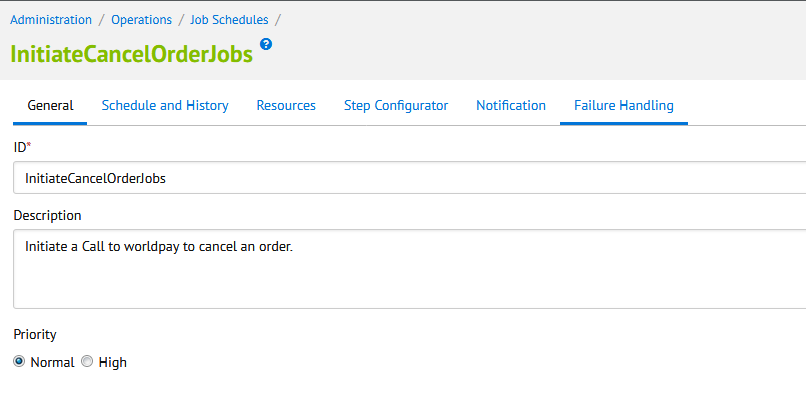
Note: Please ensure before running batch job every payment method in site has attribute “Days after which order expires” value configured.

Business Manager Configuration field:



### Initiate Cancel Order Job

The Worldpay cartridge provides support to a batch job that cancels or refunds FAILED orders with the attribute Worldpay MAC Missing. The time interval for selecting orders defined in the custom preference Worldpay Order Inquiry Lag Time. A Batch Job configured in Business Manager shall run (as per the set schedule) and check orders with order status FAILED, with Worldpay MAC Missing which are old to a defined interval of time and will send a request to Worldpay System to either cancel or refund.  Job can be Enable/Disable from Business Manager itself. Email will be sent to BM configured recipients after job run with Job run status.



Refer below Step configuration section in order to execute script and their functions

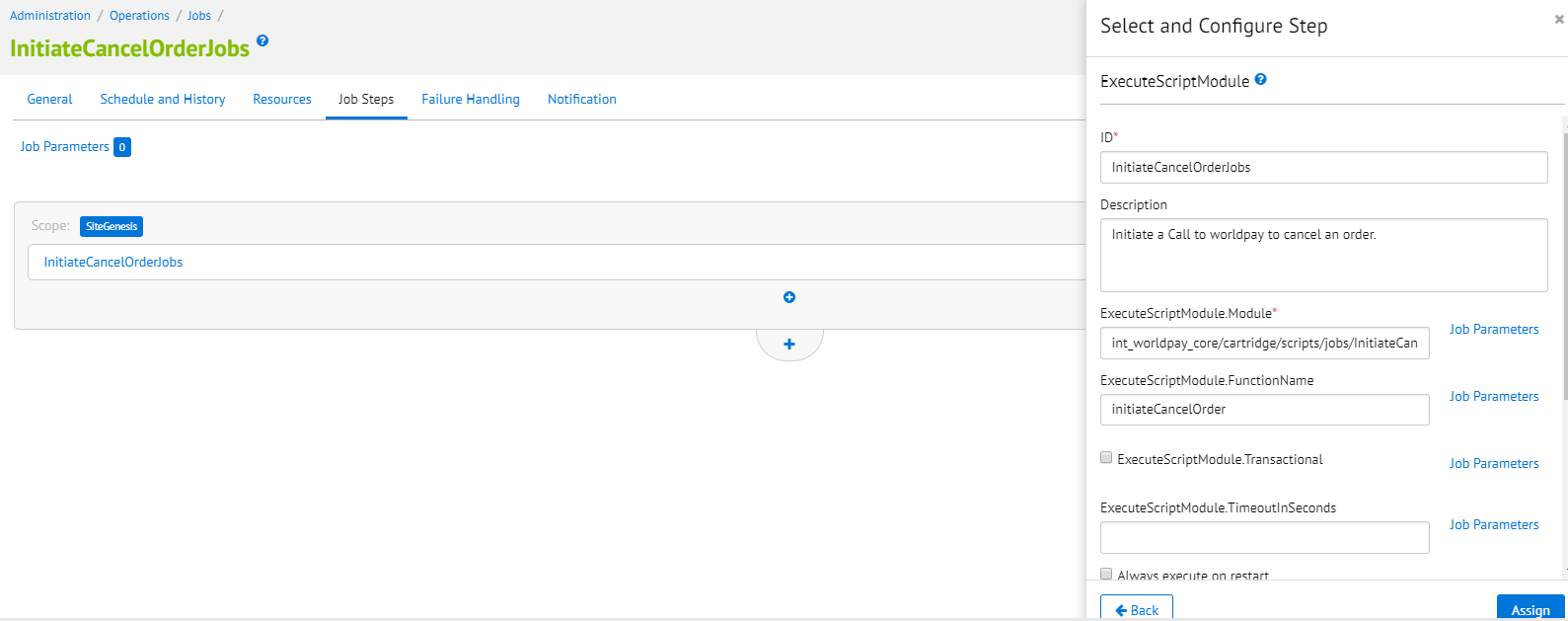


Table for Reference:

|  |  |
| --- | --- |
| Fields | Fields data |
| ID | InitiateCancelOrderJobs |
| Description | Initiate a Call to worldpay to cancel an order. |
| ExecuteScriptModule.Module | int\_worldpay\_core/cartridge/scripts/jobs/InitiateCancelOrderJob.js |
| ExecuteScriptModule.FunctionName | initiateCancelOrder |
| ExecuteScriptModule.Transactional | Enabled |

### Managing Content Asset

All import files are available in the site import folder (metadata) within the cartridge installation pack.

Note: Ensure content assets in the import has appropriate library Id of the site where XML to be imported

1. Navigate to*🡪Merchant Tools* *🡪 Content🡪Content Asset.*
2. Ensure that Content Assets with IDs “worldpayhelper” and “worldpay-elv-consent” are added.

### Managing Service

***Go to Services🡪Service Profile🡪 Worldpay Profile*** with the values listed against them:

|  |  |
| --- | --- |
| **Worldpay Profile** | **Description** |
| Name | Default “worldpayprofile”. Merchant can give any name as per need |
| Timeout (ms) | Default 30000 Timeout value in milli sec as per Merchant contract with Worldpay for webservice timeout |
| Circuit Breaker | Circuit break settings when connection not established for long time |
| Rate limits | Rate Limits after which service resumed back once circuit breaks |

Fill out the following properties under Worldpay credentials with the values listed against them:

All above service credentials are used by one Worldpay service only. (Refer Fig. 2.9)

|  |  |
| --- | --- |
| **Worldpay Credentials** | **Description** |
| Name | Merchant Hex Code provided by Worldpay |
| URL | This field represents the Web service URL that is invoked while calling Worldpay service. |
| User | This field represents the User Name provided specifically to the above mentioned Merchant Code. |
| Password | This field represents the Password provided specifically to the above mentioned Merchant Code. |

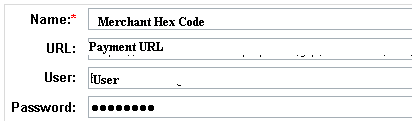


Fig 2.9

### IDEAL APM Integration

Bank configurations in Business Manager:

All the supported Bank List values are preconfigured. However, Modify/Delete the list if required. Follow the below steps to add additional banks:

1. Log into Business Manager
2. Navigate to *Administration 🡪 Site Development 🡪 System Object Definition 🡪 SitePreferences*
3. Open the tab Attribute Definition and find “*WorldpayIdealBankList*“
4. Add the required *Bank ID* and *Bank Name* in the *Value* and *Display Value* correspondingly.

(For more details refer Appendix B)

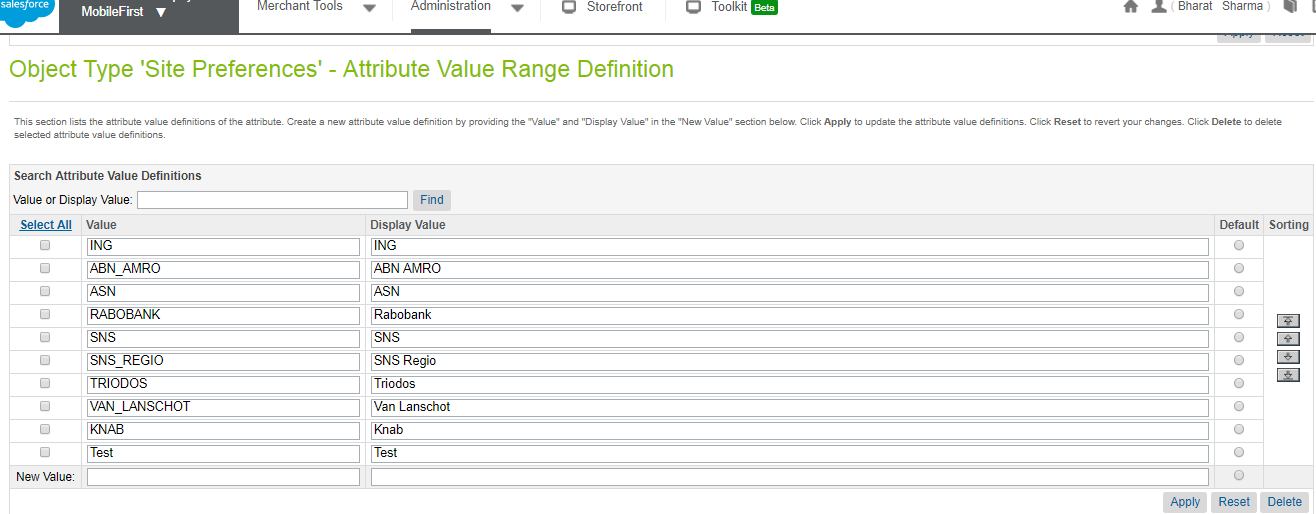


Fig: 2.6

Select the specific banks using the below steps:

1. Log into Business Manager
2. From the top bar select the required site for which configurations are to be done.
3. Navigate to SiteGenesis🡪Merchant Tools🡪 *Site Preferences 🡪 Custom Preferences 🡪 Worldpay*
4. Select the banks to be supported. (Multiple banks can be selected from the drop down)

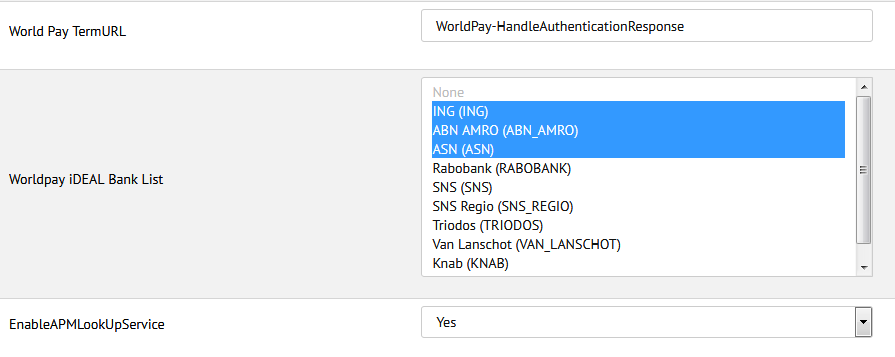


Fig: 2.7

**Note:** Please ensure list of bank should be updated; please contact Worldpay to get the latest list of Bank.

### BMultiple Merchant ID Support Integration

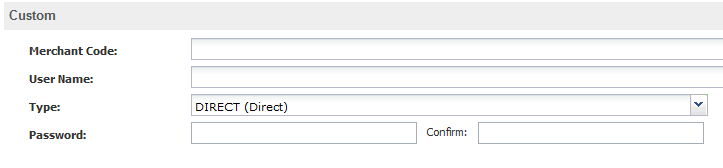
After importing metadata provided with Worldpay cartridge, additional fields (merchant ID, username, and password) will be created in payment methods in BM.

1. Log into Business Manager
2. From the top bar select the required site for which configurations are to be done
3. Navigate to SiteGenesis Site🡪Merchant Tools 🡪 Ordering🡪 Payment Methods.
4. Input the required details for the below field:

Merchant ID

Username

Password



### Look and Feel Customization for Worldpay Redirect Pages

Configurations that needs to be done in Business Manager to support look and feel customization:

1) Log into Business Manager

2) From the top bar select the required site for which configurations are to be done.

3) Navigate to SitePreferences 🡪 Custom Preferences 🡪 Worldpay

4) Input the required details for the below field:

*Worldpay Configurable HPP APM*

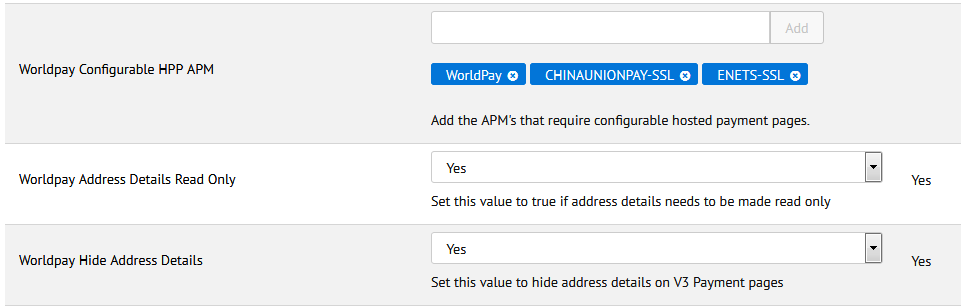
*Worldpay Installation Id*

*Worldpay Address Details Read Only*

*Worldpay Hide Address Details*

\*\* Ensure that the Payment Method ID (refer to APPENDIX A – KEY column), which needs to be supported for customizable is added to the **Worldpay Configurable HPP APM** sitepreference.

\*\*Please contact the Worldpay contact person for values of the above details.



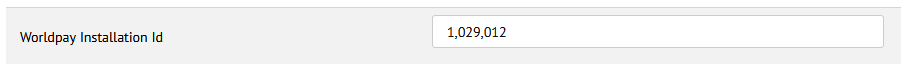


Fig 2.9

### Hosted Payment Pages iframe or lightbox for Worldpay Redirect Payment Method

Configurations that needs to be done in Business Manager to support look and feel customization:

1) Log into Business Manager

2) Navigate to SiteGenesis 🡪 Merchant Tools *🡪 Ordering 🡪 Payment Methods*

3) Select “Credit Card Redirect” payment method and add the following data under worldpay HPP CustomOptions JSON attribute.

4) Write JSON specifically type as **iframe** or **lightbox**. Click on Apply to enable the Iframe/Light box on HPP.

Example:

*{*

*“type”:”iframe”*

*}*

OR

*{*

*"type": "lightbox",*

*"lightboxMaskOpacity": 50,*

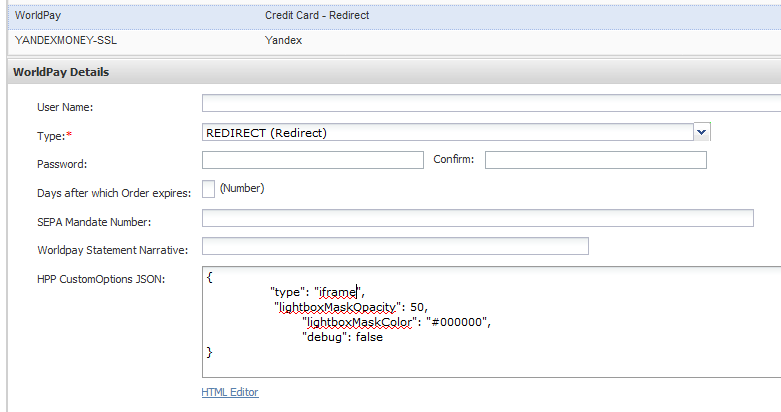
*"lightboxMaskColor": "#000000",*

*"debug": false*

*}*

For lightbox JSON settings below two attributes can be specified with merchant specific values.

* lightboxMaskOpacity
* lightboxMaskColor



### Country Currency Mapping

|  |  |
| --- | --- |
| **Country** | **Currency** |
| United States | US Dollars |
| UNITED KINGDOM | Pound Sterling |
| GERMANY | Euro |
| SINGAPORE | Singapore Dollar |
| JAPAN | Japanese Yen |
| SWEDEN | Yuan Renminbi |
| CHINA | Euro |
| FRANCE | Euro |
| RUSSIAN FEDERATION | Russian Ruble |
| BELGIUM | Euro |
| ITALY | Euro |
| CANADA | Canadian Dollar |
| AUSTRALIA | Australian Dollar |

### Country Currency and Locale Mapping for Klarna

|  |  |  |  |
| --- | --- | --- | --- |
| purchase\_country | purchase\_currency | locale | extra locale |
| AT | Euro | de-at | en-at |
| FI | Euro | fi-fi | en-fi , sv-fi |
| DE | Euro | de-de | en-de |
| NL | Euro | nl-nl | en-nl |
| NO | NOK | nb-no | en-no |
| SE | SEK | sv-se | en-se |
| GB | GBP | gb-gb | en-sb |

* Ensure the appropriate locale ,currencies and countries setting are available in business manager.

## CUSTOM CODE

**Pre-Requisite:** Make sure the cartridge of site SiteGenesis “**int\_worldpay:int\_worldpay\_core:app\_storefront\_controllers:app\_storefront\_core”** is specified in Site Settings path under Manage Sites > Merchant Site

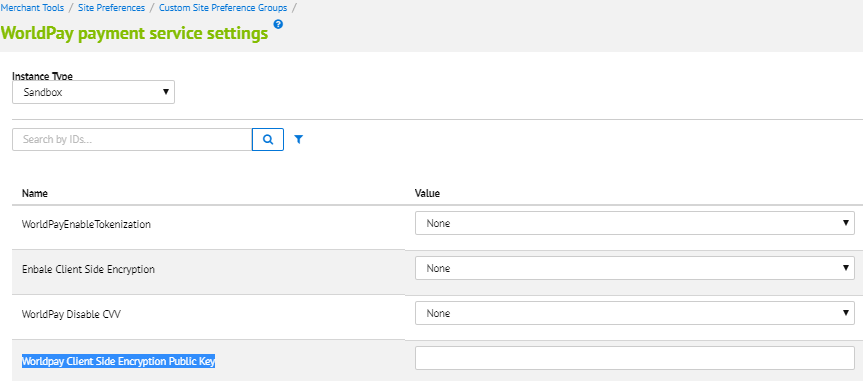
This section covers only SiteGenesis changes required in int\_worldpay to support controllers to work.

### Error Messaging

In cartridge, all the error messaging is achieved using properties file (worldpayerror.properties). These messages can be configured while installation of the cartridge and can be modified based upon the merchants business requirement. Using properties file for messaging gives the Merchant the advantage for supporting multi lingual site.

### Client Side Encryption

Please get the client side encryption from the worldpay merchant interface in order to encrypt the Card PII data. This will prevent the exposure of Card PII over the network. Download the public key of Client side encryption from worldpay interface and configure it in the site preference (‘Worldpay Client Side Encryption Public Key’) available in business manager (SiteGenesis Merchant Tools>Site Preferences>Custom Site Preference Groups).



### 3.3.3 Form File Changes

Disclaimer: shipping and billing address in forms must have country code in UPPERCASE as Worldpay mandatory requirement. We have delivered only default locale form files. If merchant require any specific locale they can copy the required form to respective locales.

Purpose: Worldpay XML request must have country code in UPPERCASE.

## CUSTOM CODE FOR CONTROLLERS

Pre-Requisite: Make sure the controller cartridge of site site-genesis is (say, e.g. app\_sitegenesis\_controllers) and “int\_worldpay” are specified in Site Settings path under Manage Sites > Merchant Site

This section covers only controller changes required in int\_worldpay to support controllers to work.

### Generic Section

**Replacing the references of SiteGenesiscontroller in WorldPay Cartridge**

Replace reference of storefront cartridge name in Global variables (say, app\_storefront\_controllers) with sitegenesis controller cartridge name (say, app\_sitegenesis\_controllers)

This modification is required in two controller files “WorldPay.ds” and “WorldPayHelper.ds”



Replace reference of storefront cartridge name in ISML with sitegenesis core cartridge name (say, app\_sitegenesis\_core)

This modification is required in ISML file “worldpaypaymentmethods.isml”, code specified below

var currentCountry = require('app\_sitegenesis\_core/cartridge/scripts/util/Countries').getCurrent(pdict);

#### Controller – COBilling.js

##### Update “returnToForm” Function

1. Include script module at the top of script include section

Include global variable below the var guard

|  |
| --- |
| var WorldpayHelper = require('int\_worldpay/cartridge/scripts/WorldpayHelper'); |

1. Replace params section by below code snippet under the function returnToForm to handle error cases

|  |
| --- |
| function returnToForm(cart, params) {  var pageMeta = require('~/cartridge/scripts/meta');  // if the payment method is set to gift certificate get the gift certificate code from the form  if (!empty(cart.getPaymentInstrument()) && cart.getPaymentInstrument().getPaymentMethod() === PaymentInstrument.METHOD\_GIFT\_CERTIFICATE) {  app.getForm('billing').copyFrom({  giftCertCode: cart.getPaymentInstrument().getGiftCertificateCode()  });  }  pageMeta.update({  pageTitle: Resource.msg('billing.meta.pagetitle', 'checkout', 'SiteGenesis Checkout')  });  if (params && params.errorMessage) {  app.getView(require('~/cartridge/scripts/object').extend(params, {  Basket : cart.object,  ContinueURL : URLUtils.https('COBilling-Billing'),  errorMessage : params.errorMessage  })).render('checkout/billing/billing');  } else if (!empty(params)) {  app.getView(require('~/cartridge/scripts/object').extend(params, {  Basket: cart.object,  ContinueURL: URLUtils.https('COBilling-Billing')  })).render('checkout/billing/billing');  } else {  app.getView({  Basket: cart.object,  ContinueURL: URLUtils.https('COBilling-Billing')  }).render('checkout/billing/billing');  }  } |

##### Update “initCreditCardList” function

* Update below code snippet under initCreditCardList to add worldpay available credit cards

|  |
| --- |
| function initCreditCardList(cart) {  var paymentAmount = cart.getNonGiftCertificateAmount();  var countryCode;  var applicablePaymentMethods;  var applicablePaymentCards;  var applicableCreditCards;  countryCode = cart.object.billingAddress && cart.object.billingAddress.countryCode.value ? cart.object.billingAddress.countryCode.value : Countries.getCurrent({  CurrentRequest: {  locale: request.locale  }  }).countryCode;    applicablePaymentMethods = PaymentMgr.getApplicablePaymentMethods(customer, countryCode, paymentAmount.value);  applicablePaymentCards = PaymentMgr.getPaymentMethod(PaymentInstrument.METHOD\_CREDIT\_CARD).getApplicablePaymentCards(customer, countryCode, paymentAmount.value);  app.getForm('billing').object.paymentMethods.creditCard.type.setOptions(applicablePaymentCards.iterator());  applicableCreditCards = null;  if (customer.authenticated) {  var profile = app.getModel('Profile').get();  if (profile) {  applicableCreditCards = profile.validateWalletPaymentInstruments(countryCode, paymentAmount.getValue()).ValidPaymentInstruments;  }  applicableCreditCards = WorldpayHelper.AvailableCreditCards(applicableCreditCards);  }  return {  ApplicablePaymentMethods: applicablePaymentMethods,  ApplicableCreditCards: applicableCreditCards  };  } |

##### Update “publicStart” function

* Update the method to handle error messages

|  |
| --- |
| function publicStart(errorMessage) {  var cart = app.getModel('Cart').get();  if (cart) {  // Initializes all forms of the billing page including: - address form - email address - coupon form  initAddressForm(cart);  initEmailAddress(cart);  var creditCardList = initCreditCardList(cart);  var applicablePaymentMethods = creditCardList.ApplicablePaymentMethods;  var billingForm = app.getForm('billing').object;  var paymentMethods = billingForm.paymentMethods;  if (paymentMethods.valid) {  paymentMethods.selectedPaymentMethodID.setOptions(applicablePaymentMethods.iterator());  } else {  paymentMethods.clearFormElement();  }  app.getForm('billing.couponCode').clear();  app.getForm('billing.giftCertCode').clear();  if(errorMessage && errorMessage.errorMessage){  start(cart, require('~/cartridge/scripts/object').extend(errorMessage, {ApplicableCreditCards: creditCardList.ApplicableCreditCards}));  }  else{  start(cart, {ApplicableCreditCards: creditCardList.ApplicableCreditCards});  }  } else {  app.getController('Cart').Show();  }  } |

* Also update creditcardlist under local methods

exports.InitCreditCardList=initCreditCardList; and exports.HandlePaymentSelection = handlePaymentSelection;

##### Update “validatePayment” function

* Update the method with the countrycode conditions.

|  |
| --- |
| function validatePayment(cart) {  var paymentAmount, countryCode, invalidPaymentInstruments, result;  if (app.getForm('billing').object.fulfilled.value) {  paymentAmount = cart.getNonGiftCertificateAmount();  countryCode = cart.object.billingAddress && cart.object.billingAddress.countryCode ? cart.object.billingAddress.countryCode : Countries.getCurrent({  CurrentRequest: {  locale: request.locale  }  }).countryCode;  invalidPaymentInstruments = cart.validatePaymentInstruments(customer, countryCode, paymentAmount.value).InvalidPaymentInstruments;  if (!invalidPaymentInstruments && cart.calculatePaymentTransactionTotal()) {  result = true;  } else {  app.getForm('billing').object.fulfilled.value = false;  result = false;  }  } else {  result = false;  }  return result;  } |

##### Update “saveCreditCard” function

* Update the method to update worldpay card details. This will ensure that duplicate card won’t be saved from My Account and order placement

|  |
| --- |
| function saveCreditCard() {  var i, creditCards, GetCustomerPaymentInstrumentsResult, newCreditCard;  if (customer.authenticated && app.getForm('billing').object.paymentMethods.creditCard.saveCard.value  && app.getForm('billing').object.paymentMethods.selectedPaymentMethodID.value != Resource.msg('orderdetails.paymentmethod.worldpay','worldpay',null)) {  // var creditCards = customer.getProfile().getWallet().getPaymentInstruments(PaymentInstrument.METHOD\_CREDIT\_CARD);  var ArrayList = require('dw/util/ArrayList');  var Pipelet = require('dw/system/Pipelet');  creditCards = new ArrayList();  GetCustomerPaymentInstrumentsResult = new Pipelet('GetCustomerPaymentInstruments').execute({  Customer: customer,  PaymentMethod: PaymentInstrument.METHOD\_CREDIT\_CARD  });  if (GetCustomerPaymentInstrumentsResult.result !== PIPELET\_ERROR) {  creditCards = GetCustomerPaymentInstrumentsResult.PaymentInstruments;  }  Transaction.wrap(function () {  newCreditCard = customer.getProfile().getWallet().createPaymentInstrument(PaymentInstrument.METHOD\_CREDIT\_CARD);  // copy the credit card details to the payment instrument  newCreditCard.setCreditCardHolder(app.getForm('billing').object.paymentMethods.creditCard.owner.value);  newCreditCard.setCreditCardNumber(app.getForm('billing').object.paymentMethods.creditCard.number.value);  newCreditCard.setCreditCardExpirationMonth(app.getForm('billing').object.paymentMethods.creditCard.expiration.month.value);  newCreditCard.setCreditCardExpirationYear(app.getForm('billing').object.paymentMethods.creditCard.expiration.year.value);  newCreditCard.setCreditCardType(app.getForm('billing').object.paymentMethods.creditCard.type.value);  for (i = 0; i < creditCards.length; i++) {  var creditcard = creditCards[i];  if (creditcard.getCreditCardExpirationMonth() === newCreditCard.getCreditCardExpirationMonth()  && creditcard.getCreditCardExpirationYear() === newCreditCard.getCreditCardExpirationYear()  && creditcard.creditCardNumber.substring(creditcard.creditCardNumber.length - 4).equals(newCreditCard.creditCardNumber.substring(newCreditCard.creditCardNumber.length - 4))  ) {  WorldpayHelper.ReplaceToken(newCreditCard, creditcard);  customer.getProfile().getWallet().removePaymentInstrument(creditcard);  }  }  });  }  return true;  } |

#### Controller – COPlaceOrder.js

##### Update “API Include” module

* Include script module at the top of script include section

|  |
| --- |
| var PaymentProcessor = app.getModel('PaymentProcessor');  var Worldpay = require('int\_worldpay/cartridge/controllers/Worldpay'); |

##### Update “handlePayments” method

* Add below section after authorizationResult received to handle the authorize result for Redirection and 3D and error message and konbini APM

|  |
| --- |
| function handlePayments(order) {  if (order.getTotalNetPrice() !== 0.00) {  var paymentInstruments = order.getPaymentInstruments();  if (paymentInstruments.length === 0) {  return {  missingPaymentInfo: true  };  }  /\*\*  \* Sets the transaction ID for the payment instrument.  \*/  var handlePaymentTransaction = function () {  paymentInstrument.getPaymentTransaction().setTransactionID(order.getOrderNo());  };  for (var i = 0; i < paymentInstruments.length; i++) {  var paymentInstrument = paymentInstruments[i];  if (PaymentMgr.getPaymentMethod(paymentInstrument.getPaymentMethod()).getPaymentProcessor() === null) {  Transaction.wrap(handlePaymentTransaction);  } else {  var authorizationResult = PaymentProcessor.authorize(order, paymentInstrument);  if(authorizationResult.returnToPage){  return {  returnToPage :true,  order : order,  klarnasnippet: authorizationResult.klarnasnippet  };  } else if(authorizationResult.redirect || authorizationResult.worldpayredirect){  return {  redirect :true,  redirectUrl : authorizationResult.redirectUrl  };  } else if(authorizationResult.is3D){  return {  is3D : true,  redirectUrl : authorizationResult.redirectUrl,  paRequest : authorizationResult.paRequest,  termUrl : authorizationResult.termUrl,  orderNo : order.orderNo  };  }  else if (authorizationResult.not\_supported || authorizationResult.error) {  return {  error : true,  errorCode : authorizationResult.errorCode,  errorMessage : authorizationResult.errorMessage  };  }else if (authorizationResult.authorized && !empty(authorizationResult.isPaymentKonbini) && authorizationResult.isPaymentKonbini) {  return {  isPaymentKonbini : true,  WorldpayRedirectURL : authorizationResult.redirectUrl  };  }  }  }  }  return {};  } |

##### Update “start” method

* Replace the initial code that return to Cart.Show() if cart does not exist with the code below by adding if condition
* Move SaveCreditCard functionality of COBilling and place the code just before submitImpl function at the end of function else condition for above cart and handle the APM’s redirections by replacing the code with below

|  |
| --- |
| function start() {  var cart = Cart.get();  if (!cart) {  app.getController('Cart').Show();  return {};  }  var COShipping = app.getController('COShipping');  // Clean shipments.  COShipping.PrepareShipments(cart);  // Make sure there is a valid shipping address, accounting for gift certificates that do not have one.  if (cart.getProductLineItems().size() > 0 && cart.getDefaultShipment().getShippingAddress() === null) {  COShipping.Start();  return {};  }  // Make sure the billing step is fulfilled, otherwise restart checkout.  if (!session.forms.billing.fulfilled.value) {  app.getController('COCustomer').Start();  return {};  }  Transaction.wrap(function () {  cart.calculate();  });  var COBilling = app.getController('COBilling');  Transaction.wrap(function () {  if (!COBilling.ValidatePayment(cart)) {  COBilling.Start();  return {};  }  });  // Recalculate the payments. If there is only gift certificates, make sure it covers the order total, if not  // back to billing page.  Transaction.wrap(function () {  if (!cart.calculatePaymentTransactionTotal()) {  COBilling.Start();  return {};  }  });  // Handle used addresses and credit cards.  var saveCCResult = COBilling.SaveCreditCard();  if (!saveCCResult) {  return {  error: true,  PlaceOrderError: new Status(Status.ERROR, 'confirm.error.technical')  };  }  // Creates a new order. This will internally ReserveInventoryForOrder and will create a new Order with status  // 'Created'.  var order = cart.createOrder();  if (!order) {  app.getController('Cart').Show();  return {};  }  var handlePaymentsResult = handlePayments(order);  if (handlePaymentsResult.error) {  Transaction.wrap(function(){  OrderMgr.failOrder(order);  });  app.getController('COBilling').Start({'errorMessage' : handlePaymentsResult.errorMessage});  return {error : false};  } else if(handlePaymentsResult.is3D) {  return {  is3D : true,  redirectUrl : handlePaymentsResult.redirectUrl,  paRequest : handlePaymentsResult.paRequest,  termUrl : handlePaymentsResult.termUrl,  orderNo :handlePaymentsResult.orderNo  };  } else if(handlePaymentsResult.returnToPage) {  app.getView({  Order : handlePaymentsResult.order,  klarnasnippet:handlePaymentsResult.klarnasnippet  }).render('checkout/summary/summary');  return {};  } else if(handlePaymentsResult.redirect) {  return {redirect :true,  redirectUrl:handlePaymentsResult.redirectUrl  };  } else if (handlePaymentsResult.missingPaymentInfo) {  Transaction.wrap(function(){  OrderMgr.failOrder(order);  });  app.getController('COBilling').Start({'errorMessage' : handlePaymentsResult.errorMessage});  return {error : true};  } else if (handlePaymentsResult.isPaymentKonbini) {  return {  Order : order,  order\_created : true,  WorldpayRedirectURL : handlePaymentsResult.WorldpayRedirectURL  };  }    // Handle used addresses and credit cards.  var saveCCResult = COBilling.SaveCreditCard();  if (!saveCCResult) {  return {  error: true,  PlaceOrderError: new Status(Status.ERROR, 'confirm.error.technical')  };  }  return submitImpl(order);  } |

##### Update “submit” method

* Replace the existing asynchronous callback for order placement submission flow submit() function with the code below:

|  |
| --- |
| function submit() {    var isWorldpayPaymentProcessor = false;  var WorldpayHelper = require('int\_worldpay/cartridge/scripts/WorldpayHelper');  var order = WorldpayHelper.GetOrder().Order;  var paymentMethod:String;  if (order!=null) {  //Check payment processor as worldpay  var paymentInstruments = order.getPaymentInstruments();  if (paymentInstruments.length > 0) {  for (var i = 0; i < paymentInstruments.length; i++) {  var paymentInstrument = paymentInstruments[i];  var paymentProcessor = PaymentMgr.getPaymentMethod(paymentInstrument.getPaymentMethod()).getPaymentProcessor();  if (paymentProcessor != null && paymentProcessor.getID().equals("Worldpay")) {  isWorldpayPaymentProcessor = true;  paymentMethod = paymentInstrument.paymentMethod;  break;  } } }  } else {  app.getController('Cart').Show();  return {success : false}; }  if (isWorldpayPaymentProcessor) {  /\*var Order = require('dw/order/Order');  var paymentStatus = request.querystring.paymentStatus;  Logger.getLogger('worldpay').debug(req.querystring.order\_id + ' orderid COPlaceOrder paymentStatus ' + paymentStatus);\*/  if(paymentMethod.equals(dw.order.OrderPaymentInstrument.METHOD\_CREDIT\_CARD)){  if (submitImpl(order).error) {  app.getController('COSummary').Start();  return;  }  app.getController('COSummary').ShowConfirmation(order);  return;  }  var redirectionResult = Worldpay.HandleRedirection(order);  if (redirectionResult.success) {  if (submitImpl(order).error) {  app.getController('COSummary').Start();  return;  }  app.getController('COSummary').ShowConfirmation(order, redirectionResult.klarnasnippet);  return;  }  return;  } else {  var Logger = require('dw/system/Logger');  Logger.getLogger("worldpay").debug("AuthorizeOrder.ds : not worldpay order anuj order" );  if (submitImpl(order.object).error) {  app.getController('COSummary').Start();  return;  }  app.getController('COSummary').ShowConfirmation(order);  return;  }  } |

##### Update “submitImpl” method

* Add submitImpl method in module export section at the end
* exports.SubmitImpl = guard.ensure(['https'], submitImpl);
* Add “submitImpl “function with the code below

|  |
| --- |
| function submitImpl(order) {  var orderPlacementStatus = Transaction.wrap(function () {  if (OrderMgr.placeOrder(order) === Status.ERROR) {  OrderMgr.failOrder(order);  app.getController('COBilling').Start();  return false;  }  order.setConfirmationStatus(order.CONFIRMATION\_STATUS\_CONFIRMED);  return true;  });  if (orderPlacementStatus === Status.ERROR) {  return {error: true};  }  // Creates purchased gift certificates with this order.  /\*if (!createGiftCertificates(order)) {  Transaction.wrap(function () {  OrderMgr.failOrder(order);  });  app.getController('COBilling').Start();  return {error: true};  }\*/  // Send order confirmation and clear used forms within the checkout process.  var Email = app.getModel('Email');  var Resource = require('dw/web/Resource');  Email.get('mail/orderconfirmation', order.getCustomerEmail())  .setSubject((Resource.msg('order.orderconfirmation-email.001', 'order', null) + ' ' + order.getOrderNo()).toString())  .send({  Order: order  });  // Mark order as EXPORT\_STATUS\_READY.  Transaction.wrap(function () {  order.setExportStatus(dw.order.Order.EXPORT\_STATUS\_READY);  order.setConfirmationStatus(dw.order.Order.CONFIRMATION\_STATUS\_CONFIRMED);  });  // Clears all forms used in the checkout process.  session.forms.singleshipping.clearFormElement();  session.forms.multishipping.clearFormElement();  session.forms.billing.clearFormElement();  return {  Order: order,  order\_created: true  };  } |

#### Controller – COSummary.js

##### Update “API Include” module

* Include script module at the top of script include section

|  |
| --- |
| var Cart = app.getModel('Cart');  var ISML = require('dw/template/ISML');  var OrderMgr = require('dw/order/OrderMgr');  var Order = app.getModel('Order'); |

##### Update “submit” method

* Replace the existing submit method with the code

|  |
| --- |
| function submit() {  // Calls the COPlaceOrder controller that does the place order action and any payment authorization.  // COPlaceOrder returns a JSON object with an order\_created key and a boolean value if the order was created successfully.  // If the order creation failed, it returns a JSON object with an error key and a boolean value.  var placeOrderResult = app.getController('COPlaceOrder').Start();  if (placeOrderResult.error) {  start();  return;  }else if(placeOrderResult.redirect){  app.getView({  ContinueURL: placeOrderResult.redirectUrl,  Location :placeOrderResult.redirectUrl  }).render('util/redirect');  return;  }else if(placeOrderResult.is3D){  var OrderObj: Order = OrderMgr.getOrder(placeOrderResult.orderNo);  var Resource = require('dw/web/Resource');  var error3D1 = Resource.msg('worldpay.redirect.error1','worldpayerror',null);  var error3D2 = Resource.msg('worldpay.redirect.error2','worldpayerror',null);  var message3D = Resource.msg('worldpay.3dsecure.message','worldpay',null);    ISML.renderTemplate('worldpayissuerredirect',{  ContinueURL: placeOrderResult.redirectUrl,  IssuerURL:placeOrderResult.redirectUrl,  PaRequest:placeOrderResult.paRequest,  TermURL:placeOrderResult.termUrl,  Order:OrderObj,  error3D1: error3D1,  error3D2: error3D2,  message3D: message3D  });  return;  }  else if (placeOrderResult.order\_created) {  if (placeOrderResult.WorldpayRedirectURL) {  showConfirmation(placeOrderResult.Order, placeOrderResult.WorldpayRedirectURL);  } else {  showConfirmation(placeOrderResult.Order);  }  return;  }  } |

##### Update “showConfirmation” method

* Add another parameter named “worldpayRedirectURL” to function “showConfirmation”and also pass the same to view. Code snippet specified below

|  |
| --- |
| function showConfirmation(order,worldPayRedirectURL) {  if (!customer.authenticated) {  // Initializes the account creation form for guest checkouts by populating the first and last name with the  // used billing address.  var customerForm = app.getForm('profile.customer');  customerForm.setValue('firstname', order.billingAddress.firstName);  customerForm.setValue('lastname', order.billingAddress.lastName);  customerForm.setValue('email', order.customerEmail);  customerForm.setValue('orderNo', order.orderNo);  }  app.getForm('profile.login.passwordconfirm').clear();  app.getForm('profile.login.password').clear();  var pageMeta = require('~/cartridge/scripts/meta');  pageMeta.update({pageTitle: Resource.msg('confirmation.meta.pagetitle', 'checkout', 'SiteGenesis Checkout Confirmation')});  app.getView({  Order: order,  WorldpayRedirectURL : worldPayRedirectURL,  ContinueURL: URLUtils.https('Account-RegistrationForm') // needed by registration form after anonymous checkouts  }).render('checkout/confirmation/confirmation');  } |

##### Add “submitOrder” method

* Add new flow “COSummary-SubmitOrder” which gets invoked when user click “Place order” button on order review page for “Credit Card Redirect” payment method implementation where merchant desire to implement Hosted Payment Pages rendering as IFRAME or LIGHTBOX.
* Add below new code snippet under COSummary and export the method at the end

exports.SubmitOrder = guard.ensure(['https'], submitOrder);

|  |
| --- |
| function submitOrder() {  var cart = Cart.get();  if (cart) {  submit();  return;  } else if (!empty(session.privacy.order\_id)) {  app.getView({  Order : Order.get(session.privacy.order\_id).object  }).render('checkout/summary/summary');  return;  } else {  app.getController('Cart').Show();  return {};  }  } |

#### Controller – PaymentInstruments.js

##### Update “create” method

* Modify create function by adding below code to stop adding duplicate cards in worldPay integration

|  |
| --- |
| if (isDuplicateCard) {  wallet.removePaymentInstrument(oldCard);  }    var WorldpayHelper = require('int\_worldpay/cartridge/scripts/WorldpayHelper');  WorldpayHelper.ReplacePaymentInstrument(paymentInstrument, paymentInstruments);  Transaction.commit(); |

#### ISML Template changes

##### Update “summary.isml”

* Remove below code snippet in summary.isml

|  |
| --- |
| <isif condition="${pdict.PlaceOrderError != null}">  <div class="error-form">${Resource.msg(pdict.PlaceOrderError.code,'checkout',null)}</div> |

* Also replace “pdict.Basket” with “LineCntr” in file.
* Replace <isreportcheckout tag and <ischeckoutprogressindicator tag sections in summary.isml on the summary page with below code snippet. This code snippet determine if the page is summary page or summary page with IFRAME/Lightbox

|  |
| --- |
| <isif condition="${!empty(pdict.Basket)}">  <isset name="LineCntr" value="${pdict.Basket}" scope="page"/>  <iselseif condition="${!empty(pdict.Order)}">  <isset name="LineCntr" value="${pdict.Order}" scope="page"/>  </isif>  <isset name="summaryaction" value="${URLUtils.https('COSummary-Submit')}" scope="page" />  <script src="${URLUtils.staticURL('/lib/jquery/jquery-1.11.1.min.js')}" type="text/javascript"></script>  <isif condition="${!empty(LineCntr.getPaymentInstruments())}">  <isloop items="${LineCntr.getPaymentInstruments()}" var="paymentInstr" status="loopstate">  <isif condition="${dw.order.PaymentMgr.getPaymentMethod(paymentInstr.paymentMethod).ID=='Worldpay'}">  <isscript>  var isValidCustomOptionsHPP = require('int\_worldpay\_core/cartridge/scripts/common/Utils').isValidCustomOptionsHPP(dw.order.PaymentMgr.getPaymentMethod(paymentInstr.paymentMethod));  if (!empty(isValidCustomOptionsHPP) && empty(pdict.Basket)) {  var CustomOptionsHPPJSON = require('int\_worldpay\_core/cartridge/scripts/common/Utils').getCustomOptionsHPP(dw.order.PaymentMgr.getPaymentMethod(paymentInstr.paymentMethod),  paymentInstr.custom.worldpayRedirectURL, LineCntr.getOrderNo(), LineCntr.getOrderToken(), paymentInstr.custom.preferedCard);  }  </isscript>  </isif>  </isloop>  </isif>  <isif condition="${typeof isValidCustomOptionsHPP !== 'undefined' && !empty(isValidCustomOptionsHPP)}">  <isset name="summaryaction" value="${URLUtils.https('COSummary-SubmitOrder')}" scope="page" />  </isif>  <isif condition="${!empty(pdict.Basket)}">  <isreportcheckout checkoutstep="${5}" checkoutname="${'OrderSummary'}"/>  <isif condition="${!pdict.CurrentForms.multishipping.entered.value}">  <ischeckoutprogressindicator step="3" multishipping="false" rendershipping="${LineCntr.productLineItems.size() == 0 ? 'false' : 'true'}"/>  <iselse/>  <ischeckoutprogressindicator step="4" multishipping="true" rendershipping="${LineCntr.productLineItems.size() == 0 ? 'false' : 'true'}"/>  </isif>  </isif> |

* Replace form action “ <isset name="summaryaction" value="${summaryaction}" scope="page" />” with “<isset name="summaryaction" value="${URLUtils.https('COSummary-Submit')}" scope="page" />” and wrap the form tag having Place Order button inside if condition where basket not empty as <isif condition="${!empty(pdict.Basket)}"></isif>.
* At end of file before closing </isdecorate> tag insert below line as

|  |
| --- |
| <div class="order-summary-footer">  <div class="place-order-totals">  <isordertotals p\_lineitemctnr="${LineCntr}" p\_showshipmentinfo="${false}" p\_shipmenteditable="${false}" p\_totallabel="${Resource.msg('summary.ordertotal','checkout',null)}"/>  </div>  <isif condition="${!empty(pdict.Basket)}">  <form action="${summaryaction}" method="post" class="submit-order">  <fieldset>  <div class="form-row">  <a class="back-to-cart" href="${URLUtils.url('Cart-Show')}">  <isprint value="${Resource.msg('summary.editcart','checkout',null)}" encoding="off" />  </a>  <isif condition="${pdict.Basket.paymentInstruments[0].getPaymentMethod().equals('KLARNA-SSL') || pdict.Basket.paymentInstruments[0].getPaymentMethod().equals('Worldpay')} ">  <button class="button-fancy-large" type="submit" name="submit" value="${Resource.msg('global.submitorderklarna','locale',null)}">  ${Resource.msg('global.submitorderklarna','locale',null)}  </button>  <iselse>  <button class="button-fancy-large" type="submit" name="submit" value="${Resource.msg('global.submitorder','locale',null)}">  ${Resource.msg('global.submitorder','locale',null)}  </button>  </isif>  </div>  <input type="hidden" name="${dw.web.CSRFProtection.getTokenName()}" value="${dw.web.CSRFProtection.generateToken()}"/>    </fieldset>  </form>  </isif>  </div>  <isinclude template="worldpaysummary"/>  </isdecorate> |

##### Update “minisummary.isml”

* Replace “pdict.Basket” with “lineCtnr” in file.
* Add below code snippet in minisummary.isml after <isinclude> tag of modules inclusion. This code snippet determine basket or order whichever available taken as lineCtntr and it determine if the page is summary page or summary page with IFRAME/lightbox

|  |
| --- |
| <isif condition="${!empty(pdict.Basket)}">  <isset name="lineCtnr" value="${pdict.Basket}" scope="page"/>  <iselseif condition="${!empty(pdict.Order)}">  <isset name="lineCtnr" value="${pdict.Order}" scope="page"/>  </isif> |

* Replace <isif condition="${checkoutstep <= 5}"> with <isif condition="${checkoutstep <= 6}">
* Wrap editURL <a> tag with non-empty basket if condition

|  |
| --- |
| <isset name="editUrl" value="${URLUtils.url('Cart-Show')}" scope="page"/>  <isif condition="${lineCtnr.productLineItems.size() == 0 && lineCtnr.giftCertificateLineItems.size() == 1}">  <isset name="editUrl" value="${URLUtils.url('GiftCert-Edit','GiftCertificateLineItemID', lineCtnr.giftCertificateLineItems[0].UUID)}" scope="page"/>  </isif>  ${Resource.msg('summary.title','checkout',null)} <isif condition="${!empty(pdict.Basket)}"><a class="section-header-note" href="${editUrl}">${Resource.msg('global.edit','locale',null)}</a></isif>  </h3>    <div class="checkout-mini-cart">  <isif condition="${checkoutstep != 5 && checkoutstep != 6}">  <isminilineitems p\_lineitemctnr="${lineCtnr}"/>  </isif> |

* Replace line <isif condition="${checkoutstep != 5}"> inside div checkout-mini-cartwith <isif condition= <isif condition="${checkoutstep != 5 && checkoutstep != 6}">

<isminilineitems p\_lineitemctnr="${lineCtnr}"/>

* Replace <isif condition="${checkoutstep > 3}"> with <isif condition="${checkoutstep == 6}">

|  |
| --- |
| <iscomment>render the order totals</iscomment>  <div class="checkout-order-totals">  <iscomment><isif condition="${checkoutstep == 6}">  <isordertotals p\_lineitemctnr="${lineCtnr}" p\_showshipmentinfo="${true}" p\_shipmenteditable="${false}" p\_totallabel="${Resource.msg('global.ordertotal','locale',null)}"/></iscomment>  <isif condition="${checkoutstep > 3}">  <isordertotals p\_lineitemctnr="${lineCtnr}" p\_showshipmentinfo="${true}" p\_shipmenteditable="${true}" p\_totallabel="${Resource.msg('global.ordertotal','locale',null)}"/>  <iselse/>  <isordertotals p\_lineitemctnr="${lineCtnr}" p\_showshipmentinfo="${false}" p\_shipmenteditable="${false}" p\_totallabel="${Resource.msg('global.estimatedtotal','locale',null)}"/>  </isif>  </div> |

##### Update “minishipments.isml”

* Replace “Shipments” variable set via <isset tag, with below code snippet in minishipments.isml. This code snippet determine basket or order whichever available taken as lineCtntr and it determine if the page is summary page or summary page with IFRAME/lightbox

|  |
| --- |
| <isif condition="${!empty(pdict.Basket)}">  <isset name="lineCtnr" value="${pdict.Basket}" scope="page"/>  <isset name="Shipments" value="${pdict.Basket.shipments}" scope="page"/>  <iselseif condition="${!empty(pdict.Order)}">  <isset name="lineCtnr" value="${pdict.Order}" scope="page"/>  <isset name="Shipments" value="${pdict.Order.shipments}" scope="page"/>  </isif> |

* Replace “pdict.Basket” with “lineCtnr” inside <isif condition="${!empty(Shipments)}"> condition i.e. <isif condition="${shipment.productLineItems.length <= 0 || shipment.custom.shipmentType == null && shipment.UUID==lineCtnr.defaultShipment.UUID && !empty(shipment.shippingAddress) && empty(shipment.shippingAddress.address1)}">

And “ <isif condition="${Shipments.size() > 1 && pdict.Basket.productLineItems.size() > 0}"><div class="name">${Resource.msgf('multishippingshipments.shipment','checkout',null, shipmentCount)}</div></isif> with “<isif condition="${Shipments.size() > 1 && lineCtnr.productLineItems.size() > 0}"><div class="name">${Resource.msgf('multishippingshipments.shipment','checkout',null, shipmentCount)}</div></isif>”

* Wrap edit link <a> tag with non-empty basket if condition everywhere in file

|  |
| --- |
| <h3 class="section-header">  <isif condition="${shipment.giftCertificateLineItems.size() > 0}">  ${Resource.msg('minishipments.shipping','checkout',null)} <span>${Resource.msg('minishipments.giftcertdelivery','checkout',null)}</span>  <iselseif condition="${shipment.custom.shipmentType == 'instore'}"/>  <isset name="editUrl" value="${URLUtils.https('Cart-Show')}" scope="page"/>  <isif condition="${!empty(pdict.Basket)}"> <a href="${editUrl}" class="section-header-note">${Resource.msg('global.edit','locale',null)}</a></isif>  ${Resource.msg('cart.store.instorepickup','checkout',null)}  <iselseif condition="${shipment.shippingAddress != null && lineCtnr.productLineItems.size() > 0}"/>  <isif condition="${!empty(pdict.Basket)}"> <a href="${editUrl}" class="section-header-note">${Resource.msg('global.edit','locale',null)}</a></isif>  ${Resource.msg('minishipments.shippingaddress','checkout',null)}  </isif>  </h3> |

##### Update “minibillinginfo.isml”

* Replace “billingAddress” and “paymentInstruments” variables set via <isset tag, with below code snippet in minibillinginfo.isml. This code snippet determine basket or order whichever available taken as lineCtntr and it determine if the page is summary page or summary page with IFRAME/lightbox

|  |
| --- |
| <isif condition="${!empty(pdict.Basket)}">  <isset name="lineCtnr" value="${pdict.Basket}" scope="page"/>  <isset name="billingAddress" value="${pdict.Basket.billingAddress}" scope="page"/>  <isset name="paymentInstruments" value="${pdict.Basket.paymentInstruments}" scope="page"/>  <iselseif condition="${!empty(pdict.Order)}">  <isset name="lineCtnr" value="${pdict.Order}" scope="page"/>  <isset name="billingAddress" value="${pdict.Order.billingAddress}" scope="page"/>  <isset name="paymentInstruments" value="${pdict.Order.paymentInstruments}" scope="page"/>  </isif> |

* Replace ”<a href="${URLUtils.https('COBilling-Start')}" class="section-header-note">${Resource.msg('global.edit','locale',null)}</a>” with if condition everywhere in file. <isif condition="${!empty(pdict.Basket)}"> <a href="${URLUtils.https('COBilling-Start')}" class="section-header-note">${Resource.msg('global.edit','locale',null)}</a></isif>
* Add a selectedPaymentMethodMiniSection as follows

<div class="selectedPaymentMethodMiniSection" data-paymentmethodid="${paymentInstr.paymentMethod}"><isprint value="${dw.order.PaymentMgr.getPaymentMethod(paymentInstr.paymentMethod).name}" /></div>

##### Update “PaymentMethods.isml”

* Add the below code snippet to paymentMethods.isml, after legend block for displaying the Error Messages in case of a failed transaction.

|  |
| --- |
| <isif condition="${pdict.errorMessage != null}">  <div class="error-form">${pdict.errorMessage}</div>  </isif> |

**\*\* Merchant shall take responsibility to add this section at the top of the Page of above the Payment Method selection section.**

* Replace all payment method specific code snippet like credit card block, Bill me later paypal etc till the closing of </fieldset> with the <div class="payment-method-container"></div> from paymentMethods.isml as we have already taken this section inside worldpay cartridge in file named “worldpaypaymentmethods.isml”

|  |
| --- |
| **<iscomment>**  **Credit card block**  **--------------------------------------------------------------**  **</iscomment>**  <iscomment>display select box with stored credit cards if customer is authenticated</iscomment>  <isif condition="${pdict.CurrentCustomer.authenticated && !empty(pdict.ApplicableCreditCards)}">  <div class="form-row">  <label class="label">${Resource.msg('billing.selectcreditcard','checkout',null)}</label>  <div class="field-wrapper">  <select name="${pdict.CurrentForms.billing.paymentMethods.creditCardList.htmlName}" id="creditCardList" class="input-select">  <option value="" selected="selected">${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="${creditCardInstr.creditCardExpirationYear}" formatter="0000" />  </a>  </isloop>  </iscomment>  </isif>  <isinputfield formfield="${pdict.CurrentForms.billing.paymentMethods.creditCard.owner}" type="input"/>  <isinputfield formfield="${pdict.CurrentForms.billing.paymentMethods.creditCard.type}" type="select"/>  <isinputfield formfield="${pdict.CurrentForms.billing.paymentMethods.creditCard.number}" type="input" dynamicname="true"/>  <div class="form-row required">  <label>  <span class="required-indicator">${Resource.msg('billing.requiredindicator','checkout',null)}</span>  <span>${Resource.msg('billing.creditcardlistexpdate', 'checkout', null)}</span>  </label>  <isscript>  var currentCountry = require('~/cartridge/scripts/util/Countries').getCurrent(pdict);  </isscript>  <isdynamicform formobject="${pdict.CurrentForms.billing.paymentMethods.creditCard.expiration}" formdata="${currentCountry.dynamicForms.expirationInfo}"/>  </div>  <isscript>  var help = {  label: Resource.msg('billing.linkcvn', 'checkout', null),  cid: 'checkout-security-code'  };  </isscript>  <isinputfield formfield="${pdict.CurrentForms.billing.paymentMethods.creditCard.cvn}" type="input" rowclass="cvn" dynamicname="true" help="${help}"/>  <isif condition="${pdict.CurrentCustomer.authenticated}">  <isinputfield formfield="${pdict.CurrentForms.billing.paymentMethods.creditCard.saveCard}" type="checkbox"/>  </isif>    </div>  <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> |

##### Update “creditcardjson.isml”

* Add below code snippet to creditcardjson.isml file

|  |
| --- |
| <isscript>  var cc = {  maskedNumber:pdict.SelectedCreditCard.maskedCreditCardNumber,  holder:pdict.SelectedCreditCard.creditCardHolder,  type:pdict.SelectedCreditCard.creditCardType,  expirationMonth:pdict.SelectedCreditCard.creditCardExpirationMonth,  expirationYear:pdict.SelectedCreditCard.creditCardExpirationYear,  showCvvField:dw.system.Site.getCurrent().getCustomPreferenceValue('WorldpayEnableTokenization') && !empty(pdict.SelectedCreditCard.creditCardToken) && dw.system.Site.getCurrent().getCustomPreferenceValue('WorldpayDisableCVV')  }  var json = JSON.stringify(cc);  </isscript> |

##### Update “ReportCheckout.isml”

Add below code snippet to ReportCheckout.isml file for tracking

* Replace “pdict.Basket” with “LineCntr” at two places and check for LineCntr for null

|  |
| --- |
| 'BasketID', LineCntr != null ? LineCntr.UUID:null, |

* Add below code snippet after <isset> variable declared in file ReportCheckout.isml. This will help in ensuring checkout tracking of new page displayed for IFRAME/lightbox in redirect flow

|  |
| --- |
| <isset name="checkoutstep" value="${pdict.checkoutstep}" scope="page"/>  <isset name="checkoutname" value="${pdict.checkoutname}" scope="page"/>  <isset name="LineCntr" value="${pdict.Basket}" scope="page"/>  <isif condition="${empty(pdict.Basket) && !empty(pdict.Order)}">  <isset name="LineCntr" value="${pdict.Order}" scope="page"/>  </isif> |

##### Update “style.css/\_checkout.scss”

* Update style.css or ideally in file \_checkout.scss in order to make lightbox center align on page add below css in file \_checkout.scss which gets compiled to style.css

|  |
| --- |
| /\* lightbox center align \*/  div#custom-html #wp-cl-lightbox {margin: 0 !important; left: 8.5%; top: 25%;}  div#wp-cl { text-align: center;}  /\*ipad\*/  @media screen and (min-width:768px) and (max-width: 959px)  {  div#custom-html #wp-cl-lightbox {left: 20%; top: 20%;position: fixed; width:60% !important;}  } |

#### APM specific changes

##### For Credit Card Redirect

###### Update “billing.xml”

* Add below code snippet in billing.xml Form inside form group “paymentMethods”

|  |
| --- |
| <group formid="paymentMethods">  <field formid="cards" type="string" mandatory="false" binding="cards"></field> |

##### For IDEAL Forms

###### Update “billing.xml”

* Add the below code snippet in billing.xml within the group with formid=”paymentMethods"

|  |
| --- |
| <group formid="paymentMethods">  <include formid="idealFields" name="ideal"/> |

##### For GiroPay Forms

###### Update “billing.xml”

* Add the below code snippet in billing.xml within the group with formid= “paymentMethods" (Screenshot attached for better understanding)

|  |
| --- |
| <group formid="paymentMethods">  <include formid="giropayFields" name="giropay"/> |

##### For SEPA DD Forms

###### Update “billing.xml”

* Add the below code snippet in billing.xml within the group with formid= “paymentMethods" (Screenshot attached for better understanding)

|  |
| --- |
| <group formid="paymentMethods">  <include formid="elvFields" name="elv"/> |



##### For Konbini

###### Update “orderconfirmation.isml”

* Add below code snippet before ThankYou section <table> tag starts in orderconfirmation.isml and after corresponding </table> tag add closing </isif> tag

|  |
| --- |
| <td colspan="2" style="font-size:12px;font-family:arial;padding:20px 10px;vertical-align:top;">  <isinclude template="worldpayconfirmation"/>  <table style="background:#ffffff;border:1px solid #999999;width:680px;">  <tr>  <th style="background:#cccccc;padding:5px 20px;font-size:12px;font-family:arial;text-align:left;">${Resource.msg('confirmation.thankyou','checkout',null)}</th>  </tr>  <tr> |

###### Update “confirmation.isml”

* Add below code section for Konbini payment method before thank you message

|  |
| --- |
| <div class="confirmation <isif condition="${!pdict.CurrentCustomer.authenticated}">create-account</isif>">  <isinclude template="worldpayconfirmation"/>  <div class="confirmation-message">  <h1>${Resource.msg('confirmation.thankyou','checkout',null)}</h1> |

#### Property file changes

##### Update “forms.properties”

* Update the country supported in the form.properties files.

|  |
| --- |
| country.ru=Russia  country.pl=Poland  country.eg=Egypt |

* Update the years in the form.properties files.

|  |
| --- |
| year.2037=2037  year.2036=2036  year.2035=2035  year.2034=2034  year.2033=2033  year.2032=2032  year.2031=2031  year.2030=2030  year.2029=2029  year.2028=2028  year.2027=2027  year.2026=2026  year.2025=2025  year.2024=2024  year.2023=2023  year.2022=2022 |

* Update the form.properties files for all the relevant countries.

##### Update “Resources.ds”

* Add below code under Resource helper URL section

|  |
| --- |
| rateLimiterReset : URLUtils.url('RateLimiter-HideCaptcha').toString(),  csrffailed : URLUtils.url('CSRF-Failed').toString(),  apmLookUp : URLUtils.https('Worldpay-APMLookupService').toString(),  loadPaymentMethods : URLUtils.https('Worldpay-LoadPaymentMethods').toString() |

* Replace CHECK\_TLS with the below code.

|  |
| --- |
| ResourceHelper.getPreferences = function(pageContext) {  var cookieHintAsset = ContentMgr.getContent('cookie\_hint');  var consentTrackingHintAsset = ContentMgr.getContent('consent\_tracking\_hint');  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'),  WP\_ENABLE\_CLIENT\_SIDE\_ENCRYPTION : (dw.system.Site.getCurrent().getCustomPreferenceValue('WorldpayEnableClientSideEncryption') ? true : false),  WP\_CSE\_PUBLIC\_KEY : dw.system.Site.getCurrent().getCustomPreferenceValue('WorldpayClientSideEncryptionPublicKey'),  WP\_DISABLE\_CVV : dw.system.Site.getCurrent().getCustomPreferenceValue('WorldpayDisableCVV')    }; |

* Update the ConsentTrackingHintAsset with the below code snippet

|  |
| --- |
| ResourceHelper.getPreferences = function(pageContext) {  var cookieHintAsset = ContentMgr.getContent('cookie\_hint');  var consentTrackingHintAsset = ContentMgr.getContent('consent\_tracking\_hint'); |

### APM Lookup Support

#### billing.js (compiled into app.js)

* Replace existing **function “setCCFields”** with below code snippet.

|  |
| --- |
| function setCCFields(data) {  var $selectPaymentMethod = $('.payment-method-options');  var selectedPaymentMethod = $selectPaymentMethod.find(':checked').val();  var $creditCard = $('[data-method="'+selectedPaymentMethod+'"]');  $creditCard.find('input[name$="creditCard\_owner"]').val(data.holder).trigger('change');  $creditCard.find('select[name$="\_type"]').val(data.type).trigger('change');  $creditCard.find('input[name\*="\_creditCard\_number"]').val(data.maskedNumber).trigger('change');  $creditCard.find('[name$="\_month"]').val(data.expirationMonth).trigger('change');  $creditCard.find('[name$="\_year"]').val(data.expirationYear).trigger('change');  if(selectedPaymentMethod && selectedPaymentMethod == 'Worldpay'){  // hide CVV  if (window.SitePreferences.WP\_DISABLE\_CVV) {  $creditCard.find('input[name\*="\_cvn"]').removeClass('required');  $creditCard.find('div.cvn').hide();  } else {  $creditCard.find('input[name\*="\_cvn"]').addClass('required');  $creditCard.find('div.cvn').show();  }  $creditCard.find('input[name$="creditCard\_owner"]').prop('readonly', true);  $creditCard.find('select[name$="\_type"]').attr("disabled", true);  $creditCard.find('input[name\*="\_creditCard\_number"]').prop('readonly', true);  $creditCard.find('[name$="\_month"]').attr("disabled", true);  $creditCard.find('[name$="\_year"]').attr("disabled", true);  $creditCard.find('input[name$="\_saveCard"]').val('false');  $creditCard.find('input[name$="\_saveCard"]').removeProp('checked');  if ($('select[id$="\_country"]').val() == "BR") {  $("#Payment\_Brazil").show();  }  }else{ // show CVV  $creditCard.find('input[name\*="\_cvn"]').addClass('required');  $creditCard.find('input[name$="\_cvn"]').val('').trigger('change');  }  } |

* Replace existing function “**updatePaymentMethod**” with below code snippet

|  |
| --- |
| function updatePaymentMethod(paymentMethodID) {  $.ajax({  url:Urls.loadPaymentMethods,  type:"Post",  data:{"selectedPaymentMethodId":paymentMethodID},  success:function(response){  $(".payment-method-container").html(response);  if($('.payment-method-options').find(':checked').val()=='Worldpay' && $('select[id$="\_country"]', $form).val() == "BR"){  $("#Payment\_Brazil").hide();  }  // select credit card from list  $('.creditCardList').on('change', function () {  var cardUUID = $(this).val();  if (!cardUUID) {  $("#clearpaymentform").trigger('click');  return;  }  // remove server side error  $('.required.error').removeClass('error');  $('.error-message').remove();  var selectedPaymentMethod = $('.payment-method-options').find(':checked').val();  if(selectedPaymentMethod=='Worldpay'){  if(cardUUID!=""){  $(".card-block").show();  //$(".save-card-block").hide();  }else{  $(".card-block").hide();  $(".save-card-block").show();  }  }  populateCreditCardForm(cardUUID);  var $creditCardSection = $('[data-method="Worldpay"]');  $creditCardSection.find('[name$="\_cards"]').val('');  var $form = $('.address');  var countryCode = $('select[id$="\_country"]', $form).val();  if(paymentMethodID == "Worldpay" && countryCode =="BR"){  $("#Payment\_Brazil").show();  $creditCardSection.find('input[name$="creditCard\_cpf"]').removeClass('required');  $creditCardSection.find('input[name$="creditCard\_installments"]').removeClass('required');  } else if(paymentMethodID == "CREDIT\_CARD" && countryCode =="BR"){  var $creditCardSection = $('[data-method="CREDIT\_CARD"]');  $creditCardSection.find('input[name$="creditCard\_cpf"]').addClass('required');  $creditCardSection.find('input[name$="creditCard\_installments"]').removeClass('required');  }  });  if(paymentMethodID=='Worldpay'){  var $creditCard = $('[data-method="Worldpay"]');  $creditCard.find('input,select').val('');  $creditCard.find('input[name$="creditCard\_owner"]').removeClass('required');  $creditCard.find('select[name$="\_type"]').removeClass('required');  $creditCard.find('input[name\*="\_creditCard\_number"]').removeClass('required');  $creditCard.find('[name$="\_month"]').removeClass('required');  $creditCard.find('[name$="\_year"]').removeClass('required');  $creditCard.find('input[name$="\_cvn"]').removeClass('required');  $('.creditCardList').val('');  $creditCard.find('input[name$="\_saveCard"]').val('false');  $creditCard.find('input[name$="\_saveCard"]').removeProp('checked');  $(".card-block").hide();  $(".save-card-block").show();  }  var $form = $('.address');  var countryCode = $('select[id$="\_country"]', $form).val();  fieldForBrazil(paymentMethodID,countryCode);    // select prefered credit card from list  $('#preferedCreditCardList').on('change', function (e) {  e.preventDefault();  var $creditCard = $('[data-method="Worldpay"]');  $creditCard.find('input[name$="creditCard\_owner"]').removeClass('required');  $creditCard.find('input[name$="creditCard\_owner"]').val('');  $creditCard.find('select[name$="\_type"]').removeClass('required');  $creditCard.find('input[name\*="\_creditCard\_number"]').removeClass('required');  $creditCard.find('input[name\*="\_creditCard\_number"]').val('');  $creditCard.find('[name$="\_month"]').removeClass('required');  $creditCard.find('[name$="\_year"]').removeClass('required');  $creditCard.find('input[name$="\_cvn"]').removeClass('required');  $('.creditCardList').val('');  $creditCard.find('input[name$="\_saveCard"]').val('true');  $creditCard.find('input[name$="\_saveCard"]').prop('checked',true);  $(".card-block").hide();  $(".save-card-block").show();  $("#Payment\_Brazil").hide();  });  $("#clearpaymentform").on('click',function(e){  e.preventDefault();  var $creditCard = $('#PaymentMethod\_Worldpay');  if ($('input#is-Worldpay').prop('checked')) {  $creditCard.find('input,select').val('');  $creditCard.find('input[name$="creditCard\_owner"]').removeClass('required');  $creditCard.find('select[name$="\_type"]').removeClass('required');  $creditCard.find('input[name\*="\_creditCard\_number"]').removeClass('required');  $creditCard.find('[name$="\_month"]').removeClass('required');  $creditCard.find('[name$="\_year"]').removeClass('required');  $creditCard.find('input[name$="\_cvn"]').removeClass('required');  $('.creditCardList').val('');  $creditCard.find('input[name$="\_saveCard"]').val('true');  $creditCard.find('input[name$="\_saveCard"]').prop('checked',true);  $(".card-block").hide();  $(".save-card-block").show();  $("#Payment\_Brazil").hide();  } else {  $creditCard = $('[data-method="CREDIT\_CARD"]');  $creditCard.find('input,select').val('');  $creditCard.find('input[name$="creditCard\_owner"]').addClass('required');  $creditCard.find('select[name$="\_type"]').addClass('required');  $creditCard.find('input[name\*="\_creditCard\_number"]').addClass('required');  $creditCard.find('[name$="\_month"]').addClass('required');  $creditCard.find('[name$="\_year"]').addClass('required');  $creditCard.find('input[name$="\_cvn"]').addClass('required');  $('.creditCardList').val('');  $creditCard.find('input[name$="\_saveCard"]').val('true');  $creditCard.find('input[name$="\_saveCard"]').prop('checked',true);  $creditCard.find('div.cvn').show();  $(".card-block").show();  $(".save-card-block").show();  if (countryCode === 'BR') {  $("#Payment\_Brazil").show();  }  }  });  }  });  $('input[name$="\_selectedPaymentMethodID"]').removeAttr('checked');  $('input[value=' + paymentMethodID + ']').prop('checked', 'checked');  //formPrepare.validateForm();  } |

* Add the below function **populateApmMethods** after function updatePaymentMethod.

|  |
| --- |
| function populateApmMethods() {  // load APM details  var countryCode = $('select[id$="\_country"]').val();  if(countryCode == undefined || countryCode ==''){  return; }  var url = Urls.apmLookUp + '?billingCountry=' + countryCode;    $.ajax({  url: url,  method: "POST",  dataType:"html"  }).done(function(data){  var result = new Array();  var foundSelectedPaymentMethod = false;  var selectedPaymentMethod = 'CREDIT\_CARD';  var paymentMethods= $(".payment-method-options");  $(paymentMethods).replaceWith(data);  if ($('.selectedPaymentMethodMiniSection').length>0) {  var datapaymentmethodid = $('.selectedPaymentMethodMiniSection').attr('data-paymentmethodid');  if ($('.payment-method-options input[value='+datapaymentmethodid+']')) {  foundSelectedPaymentMethod = true;  $('.payment-method-options input[value='+datapaymentmethodid+']').prop('checked', 'checked');  selectedPaymentMethod = datapaymentmethodid;  }  }  if (!foundSelectedPaymentMethod) {  $('input[value=CREDIT\_CARD]').prop('checked', 'checked');  }  updatePaymentMethod(selectedPaymentMethod);  });  } |

* **Remove below function creditCardList'->onChange** below inside the exports.init

$('#creditCardList').on('change', function ()

|  |
| --- |
| // select credit card from list  $('#creditCardList').on('change', function () {  var cardUUID = $(this).val();  if (!cardUUID) {return;}  populateCreditCardForm(cardUUID);  // remove server side error  $('.required.error').removeClass('error');  $('.error-message').remove(); |

* **Add new function** **checkoutForm->onsubmit** after removal of above function. This handles the client Side Encryption.

|  |
| --- |
| $checkoutForm.on('submit',function(e){  if($('.payment-method-options').find(':checked').val()=='CREDIT\_CARD'){  var $creditCard = $('[data-method="CREDIT\_CARD"]');  $creditCard.find('[name$="\_encryptedData"]').val('');  var data = {  cvc: $creditCard.find('input[name\*="\_cvn"]').val(),  cardHolderName: $creditCard.find('input[name$="creditCard\_owner"]').val(),  cardNumber: $creditCard.find('input[name\*="\_creditCard\_number"]').val(),  expiryMonth: $creditCard.find('[name$="\_month"]').val() < 10 ? '0'+$creditCard.find('[name$="\_month"]').val():$creditCard.find('[name$="\_month"]').val(),  expiryYear: $creditCard.find('[name$="\_year"]').val(),  }  Worldpay.setPublicKey(window.SitePreferences.WP\_CSE\_PUBLIC\_KEY);  var encryptedData = Worldpay.encrypt(data, function(e){  console.log("Worldpay Client Side Encryption validation error "+e);  });  if (encryptedData) {  $creditCard.find('[name$="\_encryptedData"]').val(encryptedData);  }  } else if($('.payment-method-options').find(':checked').val()=='Worldpay'){  var $creditCard = $('[data-method="Worldpay"]');  $creditCard.find('select[name$="\_type"]').removeAttr("disabled");  $creditCard.find('[name$="\_month"]').removeAttr("disabled");  $creditCard.find('[name$="\_year"]').removeAttr("disabled");  }  }); |

* Replace the code line “var selectedPaymentMethod = $selectPaymentMethod.find(':checked').val(); “ with “var selectedPaymentMethod = $('.payment-method-options').find(':checked').val();” in exports init function.
* **Make a call to new function** **populateApmMethods** **on country change** inside function exports.init and below formPrepare.init({

|  |
| --- |
| var selectedPaymentMethod = $('.payment-method-options').find(':checked').val();  formPrepare.init({  formSelector: 'form[id$="billing"]',  continueSelector: '[name$="billing\_save"]'  });  var $countrySelector= $('select[name$="\_country"]');  $countrySelector.on('change',function(e){  populateApmMethods();  selectedPaymentMethod = $('.payment-method-options').find(':checked').val();  var $form = $('.address');  var countryCode = $('select[id$="\_country"]', $form).val();  fieldForBrazil((selectedPaymentMethod) ? selectedPaymentMethod : 'CREDIT\_CARD',countryCode);  //updatePaymentMethod((selectedPaymentMethod) ? selectedPaymentMethod : 'CREDIT\_CARD');  });  populateApmMethods();  selectedPaymentMethod = $('.payment-method-options').find(':checked').val();    // default payment method to 'CREDIT\_CARD'  //updatePaymentMethod((selectedPaymentMethod) ? selectedPaymentMethod : 'CREDIT\_CARD');  $checkoutForm.on('click', 'input[type="radio"][name$="\_selectedPaymentMethodID"]', function(e){  updatePaymentMethod($(this).val());  }); |

### Brazil Integration

#### Update “creditcard.xml “

* Add below code snippet in **creditcard.xml** inside the form tag

|  |
| --- |
| <!-- field for CPF incase of Brazil -->  <field formid="cpf" label="label.cpf" type="string" max-length="25" binding="CPFNumber" missing-error="creditcard.numbermissingerror" value-error="creditcard.numbervalueerror"/>    <!-- field for Installments incase of brazil -->  <field formid="installments" label="label.installments" type="string" max-length="2" binding="Installments" missing-error="creditcard.numbermissingerror" value-error="creditcard.numbervalueerror"/>  <field formid="cards" type="string" mandatory="false" binding="cards"></field> |

\*\*The merchant need to ensure and provide validation and the max limit of the installments they want to support.

#### Update “billing.js” (compiled into app.js)

Add below code snippet to billing.js for Brazil integration:

* Add below function in billing.js after populateCreditCardForm function and call this method on change of country code:

|  |
| --- |
| function fieldForBrazil(paymentMethodID,countryCode){  if(countryCode =="BR"){  if(paymentMethodID == "BOLETO-SSL"){  $("#Payment\_Brazil").show();  var $fieldsSection = $("#Payment\_Brazil");  $fieldsSection.find('input[name$="creditCard\_cpf"]').addClass('required');  $fieldsSection.find('input[name$="creditCard\_cpf"]').parents('.form-row').find('label').prepend('<span class="required-indicator">â€¢ </span>');  } else if(paymentMethodID == "CREDIT\_CARD"){  $("#Payment\_Brazil").show();  var $fieldsSection = $("#Payment\_Brazil");  $fieldsSection.find('input[name$="creditCard\_cpf"]').addClass('required');  $fieldsSection.find('input[name$="creditCard\_cpf"]').parents('.form-row').find('label').prepend('<span class="required-indicator">â€¢ </span>');  $fieldsSection.find('input[name$="creditCard\_installments"]').removeClass('required');  } else if(paymentMethodID == "Worldpay"){  var $fieldsSection = $("#Payment\_Brazil");  $fieldsSection.find('input[name$="creditCard\_cpf"]').removeClass('required');  $fieldsSection.find('input[name$="creditCard\_installments"]').removeClass('required');  } else {  $("#Payment\_Brazil").hide();  }  }else{  $("#Payment\_Brazil").hide();  }  } |

### Street Number Implementation

In order to supply the street number (in Worldpay documentation also referred as house number) for the redirect payment model, it is required to implement an appropriate parsing algorithm in the CreateRequestHelper.ds script file. Line number 83 and 145 contains the appropriate attribute that needs to be configured. The street number can appear for example in different configurations:

In front of the street name (USA, Canada)

After the street name (Europe)

As an extra field

Therefore, the appropriate parsing logic has to be implemented during installation and according to the used scenario. If this step is not taken, the customer will have to resupply the street number at the Worldpay payment page. For more information, please refer to the WorldPay documentation.

### Error Messaging

In cartridge all the error messaging is achieved using properties file (worldpayerror.properties). These messages can be configured while installation of the cartridge and can be modified based upon the merchants business requirement. Using properties file for messaging gives the Merchant the advantage for supporting multi lingual site.

### Client Side Encryption

#### Update “pt\_checkout.isml”

Add below code snippet to pt\_checkout.isml for Client Side Encryption before </head> tag closing.

|  |
| --- |
| <iscomment>Add template-specific header information here.</iscomment>    <isif condition="${dw.system.Site.getCurrent().getCustomPreferenceValue('WorldpayEnableClientSideEncryption')}">  <script src="https://payments.worldpay.com/resources/cse/js/worldpay-cse-1.0.1.min.js"></script>  </isif> |

#### Update “creditcard.xml”

* Add the below code snippet in creditcard.xml inside form tag

|  |
| --- |
| <!-- field for credit card encrypted data -->  <field formid="encryptedData" type="string" mandatory="false" /> |

Make CVN field as non mandatory

|  |
| --- |
| <!-- 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 the below snippet in creditcard.xml

|  |
| --- |
| <field formid="cards" type="string" mandatory="false" binding="cards"></field> |

* Add the below snippet to increase the size of the card

|  |
| --- |
| <!-- field for credit card number -->  <field formid="number" label="creditcard.number" type="string" mandatory="true" masked="4" max-length="23"  description="creditcard.numberexample" binding="creditCardNumber"  missing-error="creditcard.numbermissingerror" value-error="creditcard.numbervalueerror"/> |

* Add the below snippet to increase the expiration years.

|  |
| --- |
| <field formid="year" label="resource.year" type="integer" mandatory="true" binding="creditCardExpirationYear"  missing-error="creditcard.yearmissingerror">  <options>  <option optionid="2018" label="year.2018" value="2018"/>  <option optionid="2019" label="year.2019" value="2019"/>  <option optionid="2020" label="year.2020" value="2020"/>  <option optionid="2021" label="year.2021" value="2021"/>  <option optionid="2022" label="year.2022" value="2022"/>  <option optionid="2023" label="year.2023" value="2023"/>  <option optionid="2024" label="year.2024" value="2024"/>  <option optionid="2025" label="year.2025" value="2025"/>  <option optionid="2026" label="year.2026" value="2026"/>  <option optionid="2027" label="year.2027" value="2027"/>  <option optionid="2028" label="year.2028" value="2028"/>  <option optionid="2029" label="year.2029" value="2029"/>  <option optionid="2030" label="year.2030" value="2030"/>  <option optionid="2031" label="year.2031" value="2031"/>  <option optionid="2032" label="year.2032" value="2032"/>  <option optionid="2033" label="year.2033" value="2033"/>  <option optionid="2034" label="year.2034" value="2034"/>  <option optionid="2035" label="year.2035" value="2035"/>  <option optionid="2036" label="year.2036" value="2036"/>  <option optionid="2037" label="year.2037" value="2037"/>  </options> |

* Update the credit.xml files for all the respected countries.

#### Update “Resource.ds”

* Add below code under Resource helper preference section

|  |
| --- |
| CHECK\_TLS: Site.getCurrent().getCustomPreferenceValue('checkTLS'),  WP\_ENABLE\_CLIENT\_SIDE\_ENCRYPTION : (dw.system.Site.getCurrent().getCustomPreferenceValue('WorldpayEnableClientSideEncryption') ? true : false),  WP\_CSE\_PUBLIC\_KEY : dw.system.Site.getCurrent().getCustomPreferenceValue('WorldpayClientSideEncryptionPublicKey'),  WP\_DISABLE\_CVV : dw.system.Site.getCurrent().getCustomPreferenceValue('WorldpayDisableCVV') |

### Form File Changes

Disclaimer: shipping and billing address in forms must have country code in UPPERCASE as WorldPay mandatory requirement.

Purpose: WorldPay XML request must have contry code in UPPERCASE.

### Update shippingaddress.xml and billingaddress.xml

* Update the shipping and billing address with the relevant countries.
* All the countries files should also be updated for all the relevant countries.

### Update the paymentinstrumentdetails.isml

* Update the maximum length and size.

|  |
| --- |
| <isscript>  var ownerAttributes = {  maxlength: 40,  size: 40  };  var numberAttributes = {  maxlength: 23,  size: 23  };  </isscript> |

### Update the locale.properties and countries.json

* In locale.properties files update the submit order for the Klarna.
* Add the respective countries configuration in the countries.json file.

Note: There can be some existing app.js error coming from default siteGenesis. You need to check in console for app.js. That might affect credit card form

# PRODUCTION SETUP



### Production Service Setup

Steps for setting up the production service in business manager

1. In business manager navigate to “Administration > Operations > Services”
2. Click on “Credentials” Tab
3. Create new credentials by providing the merchant code, production URL, username and password
4. Now, navigate to “Merchant Tools > Site Preferences > Custom Preferences”
5. Inside “Worldpay” group “Merchant Code for Worldpay” as created in step 3. Also modify the other desired preferences on need basis
6. Worldpay Client Side Encryption Public Key
7. Worldpay MAC Secret Code
8. Worldpay Merchant Number
9. Worldpay Installation Id
10. In case if for separate merchant to be used at APM level, then service detail modification to be done in “Merchant Tools > Ordering > Payment Methods”

# OPERATIONS, MAINTAINENCE



### Order Payment Instrument Attributes

Apart from capturing the Order Attributes we also capture some Order Payment Instrument attributes. These are specific to the Payment Method and are captured in <payment> tag.

|  |  |  |
| --- | --- | --- |
| Sr. No. | Additional Custom Fields | Attribute Id |
| 1 | Bank | bank |
| 2 | Installments | installments  Note: Please contact Worldpay for further information on the max number of installments. |
| 3 | CPF | cpf |
| 4 | BankCode | bankCode |
| 5 | Worldpay Merchant ID | Merchant Hex Code whose order is placed |
| 6 | Worldpay Token Requested | Token requested value where selected for cards |

### Order Notification Custom Object

The attributes for custom Object ‘**OrderNotifyUpdates’** are defined in the below table. A new custom Object is created each time a notification is received with all these attributes and the custom object will be retained until it is deleted by the Notification job.

|  |  |  |
| --- | --- | --- |
| Sr. No. | Additional Custom Fields | Attribute Id |
| 1 | Order No | orderNo |
| 2 | Xml String | xmlString |
| 3 | Custom Object ID | ID |
| 4 | Time Stamp | timeStamp |

### Order Payment Instrument Attributes

|  |  |  |
| --- | --- | --- |
| **Name** | **Email** | **Support Type** |
| Lochan Sim | Lochan.Sim@worldpay.com | Primary |
| Rhuta Patel | Rhuta.Patel@worldpay.com | Secondary |
| Jonathan Berry | jonathan.berry@worldpay.com | Secondary |

# USER GUIDE

### Production Service Setup

Typically, the backend developer does most of the integration works. We expect that the person doing this integration is familiar with the web service, xml processing and has hands on experience with the SFCC platform

# APPENDIX A: APM/Card mapping keys

Supported Cards Mapping table:

|  |  |  |
| --- | --- | --- |
| **Card Name** | **Key Value** | **Test card number** |
| Airplus | AIRPLUS-SSL | 122000000000003 |
| American Express | AMEX-SSL | 34343434343434 |
| Dankort | [DANKORT-SSL](http://dtd.worldpay.com/v1/paymentService_v1.dtd.html#ElementDef161) | 5019717010103742 |
| Diners | DINERS-SSL | 36700102000000 |
| Discover card | DISCOVER-SSL | 6011000400000000 |
| JCB | JCB-SSL | 3528000700000000 |
| Laser | LASER-SSL | 630495060000000000 630490017740292441 |
| Maestro | MAESTRO-SSL | 6759649826438453, 6799990100000000019 |
| MasterCard | ECMC-SSL | 5555555555554444, 5454545454545454 |
| Visa | VISA-SSL | 4444333322221111,  4911830000000 |

Supported Payment Method Mapping table:

|  |  |
| --- | --- |
| **Payment Method Name** | **Key Value** |
| Konbini | KONBINI-SSL |
| Poli | POLI-SSL |
| Poli NZ | POLINZ-SSL |
| Przelewy24 | PRZELEWY-SSL |
| SEPA-DD | ELV-SSL |
| Alipay | ALIPAY-SSL |
| Boleto | BOLETO-SSL |
| CashU | CASHU-SSL |
| Credit Card – Direct | CREDIT\_CARD |
| Credit Card – Redirect | Worldpay |
| China Union Pay | CHINAUNIONPAY-SSL |
| ENETS | ENETS-SSL |
| Giropay | GIROPAY-SSL |
| IDEAL | IDEAL-SSL |
| Mistercash | MISTERCASH-SSL |
| Paypal | PAYPAL-EXPRESS |
| Sofort | SOFORT-SSL |
| Qiwi | QIWI-SSL |
| Yandex | YANDEXMONEY-SSL |

# APPENDIX B: IDEAL Bank List

|  |  |
| --- | --- |
| **Bank Name** | **Bank Code** |
| ABN | ABN\_AMRO |
| ASN | ASN |
| ING | ING |
| Knab | KNAB |
| Rabobank | RABOBANK |
| SNS | SNS |
| SNS Regio | SNS\_REGIO |
| Triodos | TRIODOS |
| Van Lanschot | VAN\_LANSCHOT |

# APPENDIX C: Error Codes and Error Messages for Transaction Notification

|  |  |
| --- | --- |
| **Master Error Code** | **Error Message** |
| 111 | XML Parse error has occurred |
| 112 | XML Corrupted, Could not find Order No. |
| 113 | XML in Custom Object Corrupted |
| 114 | Error occurred while deleting Custom Object |
| 115 | Error occurred while reading status history from Order Object |
| 116 | Error occurred while Updating Order Object |
| 117 | Error occurred while reading Custom Object |
| 118 | No Transaction History Available |
| 119 | Last status Not available as No Transaction History is Available |
| 120 | Wrong Order Number |

# APPENDIX D: Order Notification and SFCC Order status mapping

|  |  |
| --- | --- |
| **Order Notifications Received From Worldpay** | **Changes in Order object in Business Manager** |
| **AUTHORISED** | Order Status: NEW                  Payment Status: no change                  Export Status: READY FOR EXPORT                  Confirmation Status : CONFIRMED |
| **CANCELLED** | Order Status: FAILED/CANCELLED  Payment Status: NOT PAID                  Export Status: NOT EXPORTED  Confirmation Status : NOT CONFIRMED |
| **CAPTURED** | Order Status: COMPLETED                  Payment Status: PAID                  Export Status:  no change                  Confirmation Status :  no change |
| **SENT\_FOR \_REFUND** | Order Status:  no change                  Payment Status:  no change                  Export Status: no change                  Confirmation Status :  no change |
| **REFUSED** | Order Status: FAILED  Payment Status: NOT PAID                   Export Status: NOT EXPORTED  Confirmation Status : NOT CONFIRMED |
| **SETTLED** | Order Status: no change                  Payment Status: no change                  Export Status:  no change                  Confirmation Status :  no change |
| **INFORMATION\_REQUESTED** | Order Status:  no change                  Payment Status:  no change                  Export Status: no change                  Confirmation Status : no change |
| **CHARGED\_BACK** | Order Status:  no change                  Payment Status:  no change                  Export Status: no change                  Confirmation Status : no change |
| **EXPIRED** | Order Status:   FAILED                  Payment Status:   NOT PAID                  Export Status:   NOT EXPORTED           Confirmation Status :  NOT CONFIRMED |

# APPENDIX E: Error Codes and Error Messages

|  |  |
| --- | --- |
| **Master Error Code** | **Error Message** |
| worldpay.error.code1 | Internal error has occurred , please choose a different Payment Method or try again later. |
| worldpay.error.code2 | Parse error has occurred, please choose a different Payment Method or try again later. |
| worldpay.error.code3 | Invalid amount error, please choose a different Payment Method or try again later. |
| worldpay.error.code4 | Security error , please choose a different Payment Method or try again later. |
| worldpay.error.code5 | Invalid request, please choose a different Payment Method or try again later. |
| worldpay.error.code6 | Please choose a different Payment Method or try again later. |
| worldpay.error.code7 | Payment details in the order element are incorrect, please choose a different Payment Method or try again later. |
| worldpay.error.code8 | Submission error , please choose a different Payment Method or try again later. |
| worldpay.error.generalerror | Please choose a different Payment Method or try again later. |
| worldpay.error.cancelerror | Your transaction has been cancelled. |

# APPENDIX F: Enabling and disabling worldpay integration

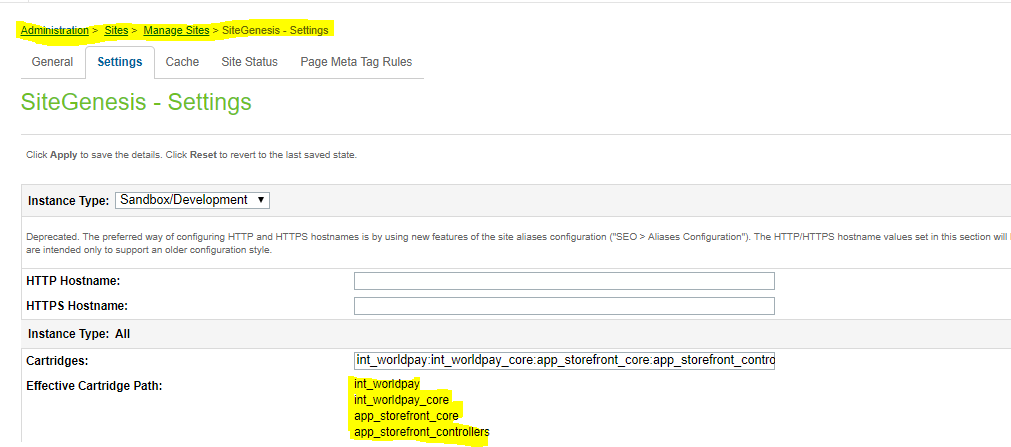
* Go to Merchant tools > Ordering > Payment Methods
* Disable all the worldpay specific payment methods
* Also cross check one by one for the type of processor, the payment method is using (To check it, click on any of the payment method lets say, CREDIT\_CARD, and see the Payment Pocessor). If it is worldpay, change it to some othe desired processor( eg. For CREDIT\_CARD, BASIC\_CREDIT).
* Enable the payment methods we want to use with the type of processor.
* To enable it again change the processor type to worldapy.
* Go to [Merchant Tools](https://worldpay02-tech-prtnr-eu04-dw.demandware.net/on/demandware.store/Sites-Site/default/ViewApplication-ExpandMenuGroup?MenuGroupID=SiteMenu&OverviewPage=SiteNavigationBar-ShowSiteOverview&csrf_token=J3jXHp4dwSBWpuaRTzOUTrs6vGohZn7ad2s3-pYNg6YDwoYT6ntqxJZdivSIfyCzzKwcICillPMHO-KHy-gvqE_nIve3vxlhtmUH1QW5V-uxdyw0UfzUHLmc3MFCB1kuhT-UwLvUA3pd27IKSC-A4qrF8ngGXYE3-077CND_5AalPq5yoY0) > [Site Preferences](https://worldpay02-tech-prtnr-eu04-dw.demandware.net/on/demandware.store/Sites-Site/default/SiteNavigationBar-ShowMenuitemOverview?CurrentMenuItemId=site-prefs&itemType=Site&csrf_token=J3jXHp4dwSBWpuaRTzOUTrs6vGohZn7ad2s3-pYNg6YDwoYT6ntqxJZdivSIfyCzzKwcICillPMHO-KHy-gvqE_nIve3vxlhtmUH1QW5V-uxdyw0UfzUHLmc3MFCB1kuhT-UwLvUA3pd27IKSC-A4qrF8ngGXYE3-077CND_5AalPq5yoY0) > custom Preferences > worldpay and disable ‘Enable Apm look up service’.

# APPENDIX G: CHECKLIST

This Section will help you ensure that all the Business Manager required configurations for worldpay are in Place and you are able to proceed with the trasactions.

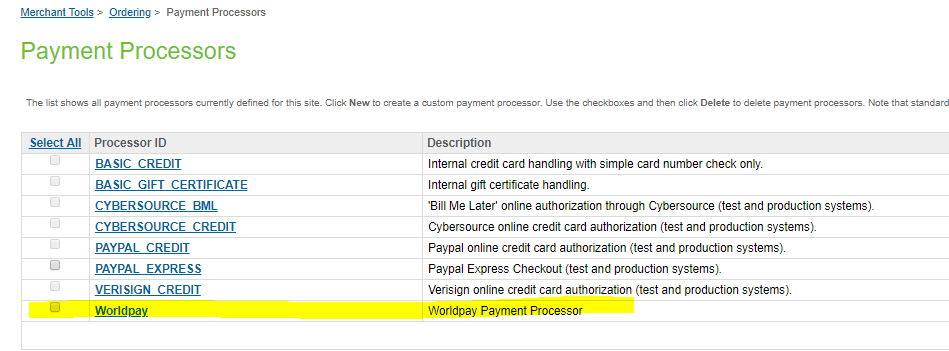
1. Cartridge Path is set properly

Go to Administration > Sites > Manage Sites > Your site - Settings



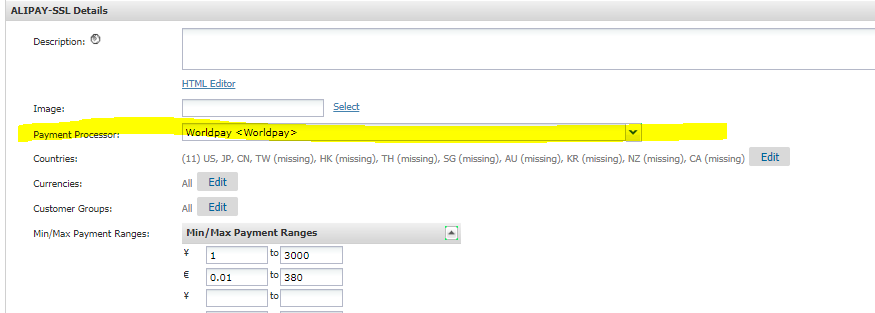
1. Payment Processor with id Worldpay is available

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

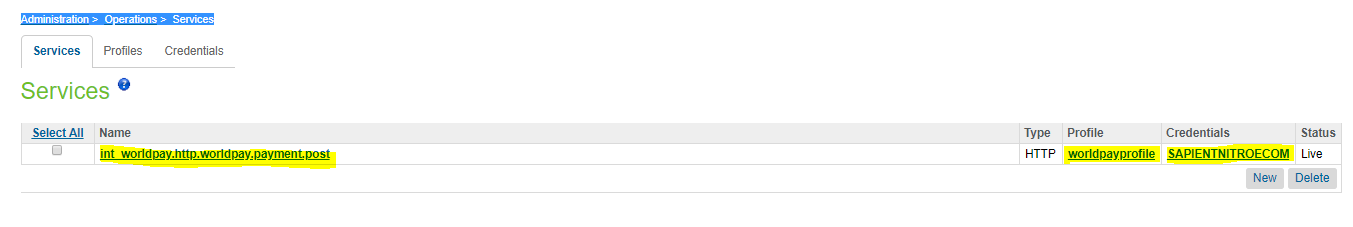


1. Payment methods are enabled and have worldpay as their processors set

* Go to **Merchant Tools > Ordering > Payment Methods,** and check for the availability of payment methods
* Open the Payment method and check if processor is worldpay or not.



1. Go to Administration > Operations > Services and check if Service is available or not



* Navigate to Merchant Tools > Content > Content Asset.
* Ensure that Content Assets with IDs “worldpayhelper” and “worldpay-elv-consent” are added.

1. Navigate to Merchant Tools > Site Preferences > Custom Preferences

Check for worldpay group availability.

If it’s there, open the group to see for all the custom preferences

1. Go to Administration > Operations > Custom Log Settings, and cross check if worldpay debug is on or not.

