Zip Payments

SiteGenesis Integration Guide

**Version <21.1.1>**





Contents

[General Information 5](#_Toc64479163)

[Components Overview 5](#_Toc64479164)

[Zip Client 5](#_Toc64479165)

[Zip Processor 5](#_Toc64479166)

[Zip Checkout 5](#_Toc64479167)

[Zip Checkout Experience 5](#_Toc64479168)

[Zip Payment Method 6](#_Toc64479169)

[PayFlex Payment Method 6](#_Toc64479170)

[QuadPay Payment Method 7](#_Toc64479171)

[Features Overview 7](#_Toc64479172)

[Checkout with Zip and Handle Zip Checkout Experience Outcomes 7](#_Toc64479173)

[Checkout with QuadPay and Handle QuadPay Checkout Experience Outcomes 8](#_Toc64479174)

[Checkout with PayFlex and Handle PayFlex Checkout Experience Outcomes 9](#_Toc64479175)

[Control Available Zip products via min/max ranges 10](#_Toc64479176)

[Handle Zip Checkout Experience cancellation 10](#_Toc64479177)

[Use tokens for delayed captures, order cancellations and refunds. 10](#_Toc64479178)

[Checkout with New Zip Account Registration 11](#_Toc64479179)

[Checkout with Zip Saved Payment method 11](#_Toc64479180)

[Hook capture, refund and cancel 11](#_Toc64479181)

[System Extensions 12](#_Toc64479182)

[Zippay payment 12](#_Toc64479183)

[QuadPay payment 12](#_Toc64479184)

[PayFlex payment 12](#_Toc64479185)

[Payment Processors 12](#_Toc64479186)

[Payment Methods 13](#_Toc64479187)

[System Object Type Extensions 14](#_Toc64479188)

[Site Preferences 14](#_Toc64479189)

[Customer Payment Instrument 15](#_Toc64479190)

[Order 15](#_Toc64479191)

[Order Address 17](#_Toc64479192)

[Order Payment Instrument 17](#_Toc64479193)

[Payment Transaction 18](#_Toc64479194)

[Profile 19](#_Toc64479195)

[Services 20](#_Toc64479196)

[Content Assets 21](#_Toc64479197)

[Installation 22](#_Toc64479198)

[Archive Contents 22](#_Toc64479199)

[Archive Preparation 22](#_Toc64479200)

[Cartridge Installation 23](#_Toc64479201)

[Custom Code 24](#_Toc64479202)

[Updates on your storefont core cartridge 24](#_Toc64479203)

[{your\_storefront\_core}/cartridge/templates/default/components/footer/footer\_UI.isml 24](#_Toc64479204)

[{your\_storefront\_core}/cartridge/templates/default/checkout/cart/cart.isml 25](#_Toc64479205)

[{your\_storefront\_core}/cartridge/templates/default/checkout/billing/billing.isml 26](#_Toc64479206)

[{your\_storefront\_core}/cartridge/templates/default/product/components/pricing.isml 27](#_Toc64479207)

[{your\_storefront\_core}/cartridge/templates/default/checkout/billing/paymentmethods.isml 29](#_Toc64479208)

[{your\_storefront\_core}/cartridge/templates/default/checkout/summary/summary.isml 31](#_Toc64479209)

[{your\_storefront\_core}/ cartridge/templates/default/components/order/orderdetailsemail.isml 33](#_Toc64479210)

[{your\_storefront\_core}/ cartridge/templates/default/components/order/orderdetails.isml 33](#_Toc64479211)

[{your\_storefront\_corecartridge/templates/default/account/payment/paymentinstrumentlist.isml 34](#_Toc64479212)

[{your\_storefront\_core}/cartridge/templates/default/account/orderhistory/orders.isml 34](#_Toc64479213)

[{your\_storefront\_corecartridge/cartridge/forms/default/profile.xml 35](#_Toc64479214)

[Updates on your storefront controller cartridge 36](#_Toc64479215)

[{your-controller-cartridge}/cartridge/controllers/COBilling.js 36](#_Toc64479216)

[{your-controller-cartridge}/cartridge/controllers/Cart.js 39](#_Toc64479217)

[{your-controller-cartridge}/cartridge/controllers/PaymentInstruments.js 40](#_Toc64479218)

[{your-controller-cartridge}//cartridge/controllers/COSummary.js 41](#_Toc64479219)

[Configuration 42](#_Toc64479220)

[Configure Zip Payments Processor 42](#_Toc64479221)

[Activate Production Mode 42](#_Toc64479222)

[Set API Key 42](#_Toc64479223)

[Enable/Disable Automatic Capture 43](#_Toc64479224)

[Enable/Disable Zip Tokenization 43](#_Toc64479225)

[Configure Zip Payment Method 43](#_Toc64479226)

[Update Payment Method Logo 43](#_Toc64479227)

[Configure PayFlex Payment Method 43](#_Toc64479228)

[Configure QuadPay Payment Method 43](#_Toc64479229)

[Allowed Countries 43](#_Toc64479230)

[Min/Max range for each currency 44](#_Toc64479231)

[Configure Zip Checkout 44](#_Toc64479232)

[Configure Zip Payment Declined Message 44](#_Toc64479233)

[Configure Thank You Page (Order Confirmation) 44](#_Toc64479234)

[Configure Zip API Client 44](#_Toc64479235)

[Configure Retries in case of General Errors 44](#_Toc64479236)

[Customizations and Extensions 46](#_Toc64479237)

[Use Back-End Hooks To Capture and Refund Amounts 46](#_Toc64479238)

[Operations, Maintenance 47](#_Toc64479239)

[Availability 47](#_Toc64479240)

[Support 47](#_Toc64479241)

[Known Issues 47](#_Toc64479242)

[Release History 48](#_Toc64479243)

# General Information

## Components Overview

### Zip Client

This component is used to connect and work with Zip API.

Each request and corresponding response made with the Zip Client is logged in a custom error log file.

The cartridge uses the following API functions:

* Checkout - Create a Checkout – when a customer clicks the button ( “Continue with Zip” for ZiP, “Continue with QuadPay” for QuadPay or “Continue with Pay Flex” for Pay Flex Payment).
* Checkout - Retrieve a Checkout – when a customer completes a Zip/QuadPay/PayFlex Checkout Experience with any status.
* Charge – Create a Charge – when payment is authorized on approved orders.
* Charge – Capture a Charge – when authorizing a payment (with automatic capture) or when executing a manual capture via the back-end hooks (refer to “Customizations and Extensions”).
* Charge – Cancel a Charge – when executing an order cancellation via the back-end hooks.
* Refund – Create a Refund – when executing a payment refund via the back-end hooks.
* Token – Create a Token – when handling approved order and tokenization is required (turned on by the merchant).

### Zip Processor

This component is used to handle payments including authorizations, captures and refunds. Zip allows partial captures and partial refunds as well.

### Zip Checkout

This component is used to extend and customize the SiteGenesis checkout with additional features to allow checking out using Zip Checkout Experience.

Two types of errors are handled when attempting to checkout with Zip:

* General, technical errors – due to unsuccessful order creation, local server issues.
* Zip Checkout errors – either due to unsuccessful connection with Zip API or bad parameters passed to it.

### Zip Checkout Experience

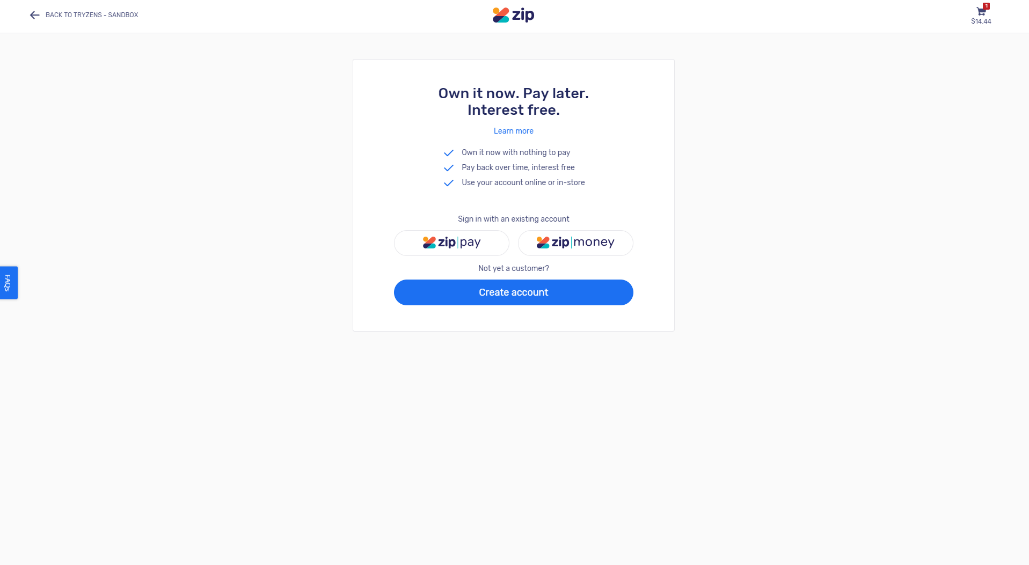
This component is provided by Zip when the customer clicks “Continue with Zip” button on last checkout step. Then Zip Checkout Experience opens as a stand-alone page.

The customer can:

* Login with an existing ZipMoney or ZipPay account to complete the order.
* Request a new customer account with ZipMoney or ZipPay.

When the customer requests a new account, there are three possibilities:

* Immediate Approval - the account gets approved immediately. The customer is then taken back to a “Thank You” page. The customer also receives a confirmation email of their completed order.
* Immediate Rejection – the account gets rejected immediately. The customer can attempt to create a new account or cancel the order.
* Referral – the account is pending approval/rejection. In such a case, the customer gets redirected back to payment page with referral error message and the order is treated as a cancelled order. Later, the customer may receive either an approval in which there would be a link to complete the order (in case of approval). Then the customer can proceed to account setup. If the cart item is still available, then the customer can continue checkout with Zip or any other payment method to place a new order.



Zip Checkout Experience, Stand-Alone Page

Please refer to <https://zip-online.api-docs.io/v1/checkout-and-registration/overview-use-cases> for more information on testing the Zip Checkout Experience: test credit card numbers, phone number, etc.

### Zip Payment Method

This component is used to show Zip as a payment method in checkout and allow customers to choose Zip as a payment method in Australia, New Zealand, United Kingdom regions.

### PayFlex Payment Method

This component is used to show PayFlex as a payment method in checkout and allow customers to choose PayFlex as a payment method South Africa region.

### QuadPay Payment Method

This component is used to show QuadPay as a payment method in checkout and allow customers to choose QuadPay as a payment method in US region.

## Features Overview

The integration described in this document supports the following features:

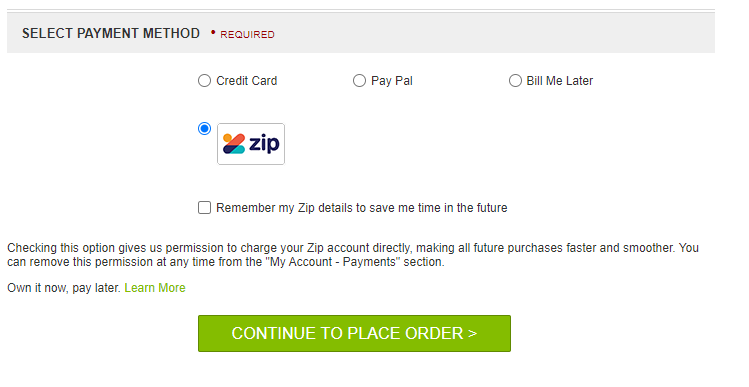
* Checkout with Zip and Handle Zip Checkout Experience Outcomes
* Checkout with QuadPay and Handle QuadPay Checkout Experience Outcomes
* Checkout with PayFlex and Handle PayFlex Checkout Experience Outcomes
* Control Available Zip products via min/max ranges.
* Use tokens for delayed captures, order cancellations and refunds.
* Checkout with New Zip Account Registration
* Handle Zip Checkout Experience cancellation.
* Handle Order Denial/Recoverable Errors.

Checkout with Zip Saved Payment method

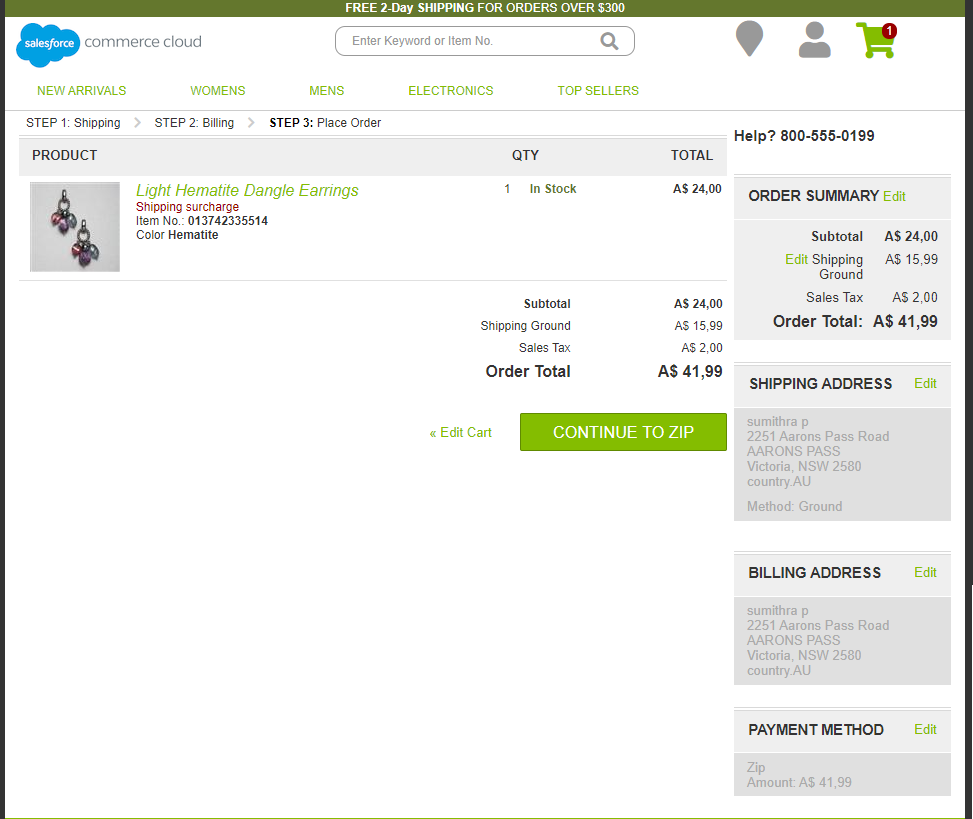
This section describes all the features listed above in detail.

### Checkout with Zip and Handle Zip Checkout Experience Outcomes

A customer visiting the merchant’s website may attempt to checkout with one of the available Zip products: ZipPay or ZipMoney by completing an enhanced SiteGenesis checkout. On the payment checkout stage, the customer selects Zip as a payment method:



The order review stage has been enhanced with a Zip-specific button “Continue to Zip”, which when clicked will attempt to create and initiate a new Zip Checkout Experience and create a new order within the merchant’s system with status CREATED.

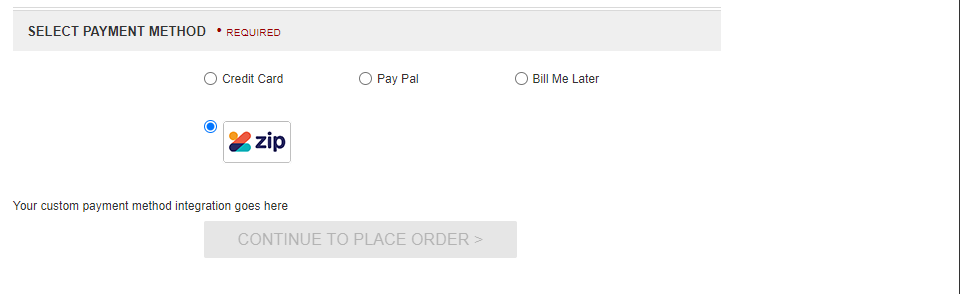


In case of order approval, the integration will attempt to place the order which will change the order’s status to NEW and authorize a new payment by creating a new charge via Zip API. If the merchant has enabled automatic capture, the payment amount associated with the newly created charge is also captured.

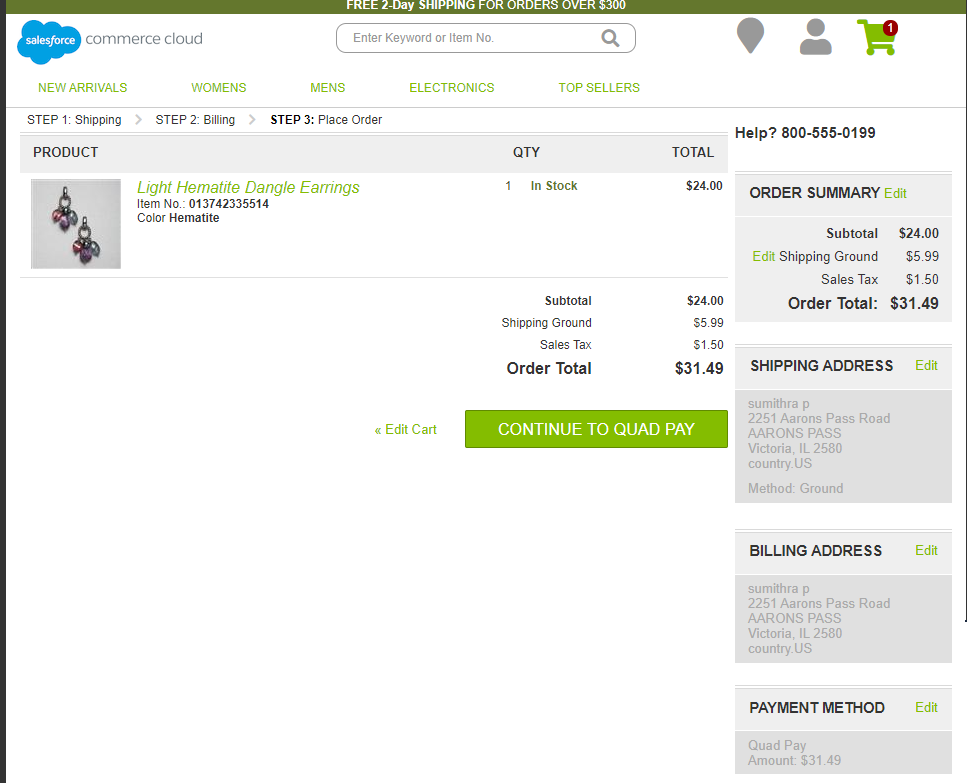
If not, the integration will redirect the customer back to the checkout’s order review stage and show a standard error message. The customer may then attempt to complete a new Zip Checkout as the “Continue to Zip” button is going to be enabled.

### Checkout with QuadPay and Handle QuadPay Checkout Experience Outcomes

A customer visiting the merchant’s website may attempt to checkout with QuadPay by completing an enhanced SiteGenesis checkout. On the payment checkout stage, the customer selects QuadPay as a payment method:



The order review stage has been enhanced with a QuadPay-specific button “Continue to QuadPay”, which when clicked will attempt to create and initiate a new QuadPay Checkout Experience and create a new order within the merchant’s system with status CREATED.

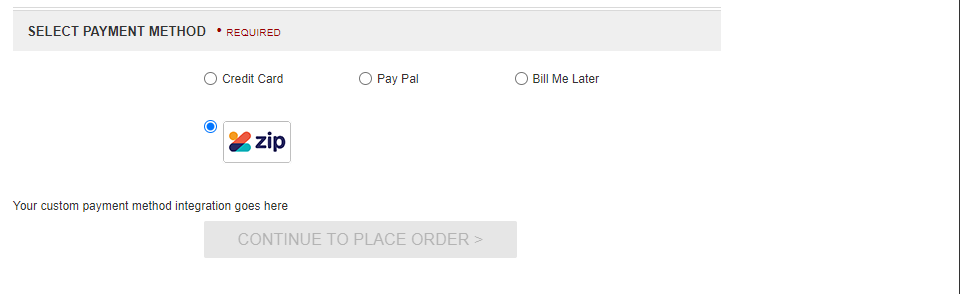


In case of order approval, the integration will attempt to place the order which will change the order’s status to NEW and authorize a new payment by creating a new charge via Zip API. If the merchant has enabled automatic capture, the payment amount associated with the newly created charge is also captured.

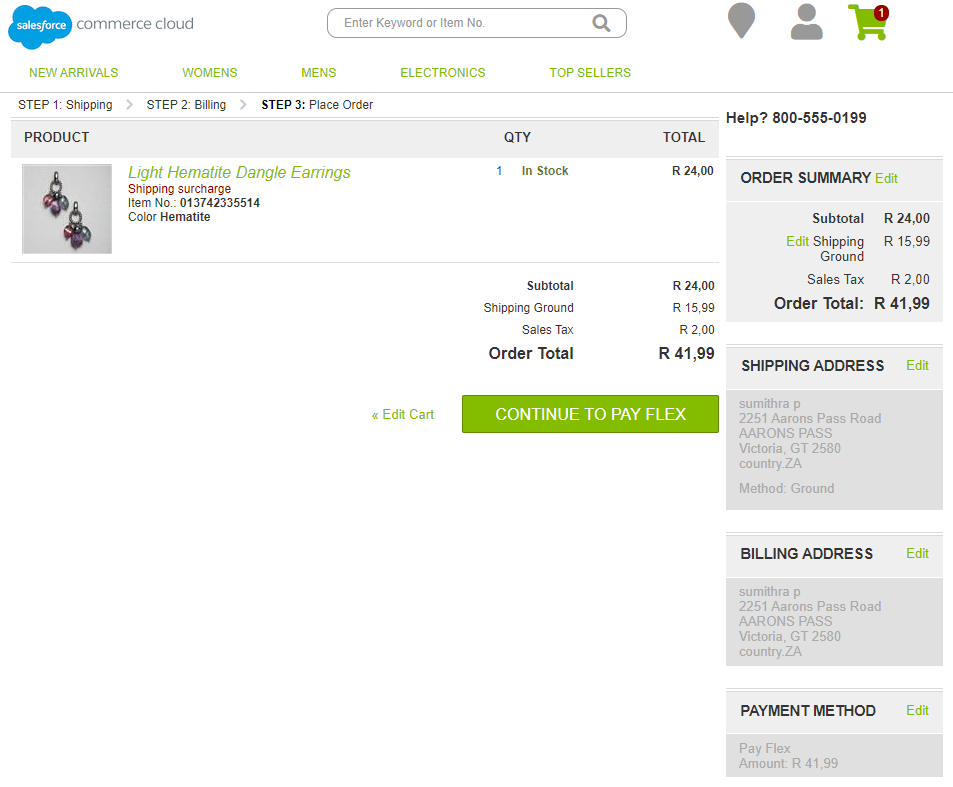
If not, the integration will redirect the customer back to the checkout’s order review stage and show a standard error message. The customer may then attempt to complete a new QuadPay Checkout as the “Continue to QuadPay” button is going to be enabled.

### Checkout with PayFlex and Handle PayFlex Checkout Experience Outcomes

A customer visiting the merchant’s website may attempt to checkout with PayFlex by completing an enhanced SiteGenesis checkout. On the payment checkout stage, the customer selects PayFlex as a payment method:



The order review stage has been enhanced with a PayFlex -specific button “Continue to Pay Flex”, which when clicked will attempt to create and initiate a new PayFlex Checkout Experience and create a new order within the merchant’s system with status CREATED.



In case of order approval, the integration will attempt to place the order which will change the order’s status to NEW and authorize a new payment by creating a new charge via Zip API. If the merchant has enabled automatic capture, the payment amount associated with the newly created charge is also captured.

If not, the integration will redirect the customer back to the checkout’s order review stage and show a standard error message. The customer may then attempt to complete a new PayFlex Checkout as the “Continue to Pay Flex” button is going to be enabled.

### Control Available Zip products via min/max ranges

The merchant can control Zip applicability via per-currency min/max range.

### Handle Zip Checkout Experience cancellation

A customer can cancel the Zip Checkout Experience by clicking a link within. If the merchant has enabled lightbox, it will close. If not, the customer is redirected back to the SiteGenesis checkout’s order review stage. Either way, the customer can attempt a new Zip Checkout Experience on the same page.

### Use tokens for delayed captures, order cancellations and refunds.

The merchant can activate tokenization which allows the system to keep a token within a successfully created order. Later, charges can be created, and payment amounts may be captured or refunded without customer’s involvement using the order’s token. Note that the created token does not expire.

The basic checkout creates a new checkout experience which later, when handling checkout completion/cancellation redirect, is used to create charges. This “checkout with tokenization” is a type of checkout which uses a token to create charges. The token is available for further captures and refunds for an order. The merchant can activate tokenization for each one of the Zip products to allow this payment flow.

### Checkout with New Zip Account Registration

A customer’s registration request may be approved within the same customer session (in a matter of seconds) or, it may require a manual approval. In case of manual approval, a Zip Agent will review the registration request and either approve or deny it. If the request has been approved, an email is sent to the customer with a link to finish the checkout process.

### Checkout with Zip Saved Payment method

This use case is applicable only for logged in customers. An additional checkbox is available when a Zip payment method is selected “Save payment method for future use”. The customer continues with checkout as usual. If the order is approved as a result of completing the Zip checkout experience, a new token is going to be requested via the ZIP API and saved to the customer’s profile.

Later, the token is used anytime the customer attempts to place an order, bypassing the need for a new Zip checkout experience altogether. On the final checkout stage, the button will read “Place Order”.

The token will be available in “My Profile” section where customers can remove the token should they want to discontinue using it. This feature is only available in Zip Australia region.

### Hook capture, refund and cancel

Merchant can programmatically hook to capture, refund and cancel charge. To do that, the merchant needs to disable automatic capture. When automatic capture is disabled, the charges are just authorized. Later, the merchant can use the hooks to trigger multiple captures of a single charge (partial capturing), refunds and charge cancellation.

# System Extensions

### Zippay payment

If you are installing the Zip payment method, install the meta files within metadata/site-template\_Zip

### QuadPay payment

For QuadPay payment, install meta files within metadata/site-template\_QuadPay

### PayFlex payment

For PayFlex payment method, install meta files within the metadata/site-template\_PayFlex

## Payment Processors

The system has been extended with a new payment processor: **Zip.**

**Settings are accessible here: “**Merchant Tools – Ordering – Payment Processors – Zip – Settings**”.**

These settings are used for Zip payment method.

|  |  |  |
| --- | --- | --- |
| **Setting** | **Value** | **Description** |
| Zip API Mode Production  (zipApiMode) | Yes/No | If set to yes, the Zip API Client will load the Zip Production Service Profile. If no, Zip API Client will load the Zip Sandbox Service Profile. |
| Zip Tokenization  (zipTokenization) | Yes/No | Enable/disable tokenization for this processor.  Default: Yes |
| Zip API Key  (zipApiKey) | String | API Key for this payment processor. |
| Automatic Capture  (zipAutoCapture) | Yes/No | Enable/disable automatic captures of charges for this processor.  Default: Yes |
| Enable Zip  (enableZippay) | Yes/No | Enable/disable Zip |

These settings are used for QuadPay payment method.

|  |  |  |
| --- | --- | --- |
| **Setting** | **Value** | **Description** |
| QuadPay API Mode Production  (zipApiMode) | Yes/No | If set to yes, the Zip API Client will load the Zip Production Service Profile. If no, Zip API Client will load the Zip Sandbox Service Profile. |
| Zip Tokenization  (zipTokenization) | Yes/No | Enable/disable tokenization for this processor.  Default: Yes |
| QuadPay API Key  (zipApiKey) | String | API Key for this payment processor. |
| Automatic Capture  (zipAutoCapture) | Yes/No | Enable/disable automatic captures of charges for this processor.  Default: Yes |
| Enable QuadPay  (enableZippay) | Yes/No | Enable/disable QuadPay. |

These settings are used for PayFlex payment method.

|  |  |  |
| --- | --- | --- |
| **Setting** | **Value** | **Description** |
| PayFlex API Mode Production  (zipApiMode) | Yes/No | If set to yes, the Zip API Client will load the Zip Production Service Profile. If no, Zip API Client will load the Zip Sandbox Service Profile. |
| Zip Tokenization  (zipTokenization) | Yes/No | Enable/disable tokenization for this processor.  Default: Yes |
| PayFlex API Key  (zipApiKey) | String | API Key for this payment processor. |
| Automatic Capture  (zipAutoCapture) | Yes/No | Enable/disable automatic captures of charges for this processor.  Default: Yes |
| Enable PayFlex  (enableZippay) | Yes/No | Enable/disable PayFlex. |

## Payment Methods

Three payment method has been added: **Zip**, **Pay Flex**, **QuadPay** linked to the payment processor described in previous section.

**Zip payment method**

|  |  |  |
| --- | --- | --- |
| **Setting** | **Values** | **Description** |
| Payment Processor | Zip |  |
| Countries | AU, NZ, UK | This payment method is applicable for Australia, New Zealand and United Kingdom by default.  Selected Countries: AU, NZ, UK. |
| Min/Max Payment Ranges |  | The merchant can set a specific range for which Zip is applicable as a payment method. |

**PayFlex payment method**

|  |  |  |
| --- | --- | --- |
| **Setting** | **Values** | **Description** |
| Payment Processor | Zip |  |
| Countries | ZA | This payment method is applicable for South Africa by default.  Selected Countries: ZA. |
| Min/Max Payment Ranges |  | The merchant can set a specific range for which Zip is applicable as a payment method. |

**QuadPay payment method**

|  |  |  |
| --- | --- | --- |
| **Setting** | **Values** | **Description** |
| Payment Processor | Zip |  |
| Countries | US | This payment method is applicable for United States by default.  Selected Countries: US. |
| Min/Max Payment Ranges |  | The merchant can set a specific range for which Zip is applicable as a payment method. |

## System Object Type Extensions

### Site Preferences

Custom preferences have been added to a new “**Zip**” group for Zip payment method. Those preferences apply to the zip integration (they are not per-processor) for Zip payment method.

|  |  |
| --- | --- |
| **Preference** | **Description** |
| Enable Zip  (enableZippay) | Enable/disable Zip |
| Zip API Mode Production  (zipApiMode) | API Mode Production for Zip |
| Zip Tokenization  (zipTokenization) | Enable/disable tokenization for this processor. Only applicable for Zip in AU region |
| Automatic Capture  (zipAutoCapture) | Enable/disable automatic captures of charges for this processor |
| Zip API Key  (zipApiKey) | API Key for Zip |

Custom preferences have been added to a new “**PayFlex**” group for PayFlex payment method. Those preferences apply to the zip integration (they are not per-processor) for PayFlex payment method.

|  |  |
| --- | --- |
| **Preference** | **Description** |
| Enable PayFlex  (enableZippay) | Enable/disable PayFlex |
| PayFlex API Mode Production  (zipApiMode) | API Mode Production for PayFlex |
| Zip Tokenization  (zipTokenization) | Enable/disable tokenization for this processor. Only applicable for Zip in AU region |
| Automatic Capture  (zipAutoCapture) | Enable/disable automatic captures of charges for this processor |
| PayFlex API Key  (zipApiKey) | API Key for PayFlex |

Custom preferences have been added to a new “**QuadPay**” group for QuadPay payment method. Those preferences apply to the zip integration (they are not per-processor) for QuadPay payment method.

|  |  |
| --- | --- |
| **Preference** | **Description** |
| Enable QuadPay  (enableZippay) | Enable/disable QuadPay |
| QuadPay API Mode Production  (zipApiMode) | API Mode Production for QuadPay |
| Zip Tokenization  (zipTokenization) | Enable/disable tokenization for this processor. Only applicable for Zip in AU region |
| Automatic Capture  (zipAutoCapture) | Enable/disable automatic captures of charges for this processor |
| QuadPay API Key  (zipApiKey) | API Key for QuadPay |

### Customer Payment Instrument

This custom preference is added to a new “**Zip - SFRA**” group for Zip payment method.

*Note: For QuadPay payment method, this custom preference is added to a new “****QuadPay- SFRA****” group and for PayFlex payment method, this custom preference is added to a new “****PayFlex- SFRA****” group*.

|  |  |
| --- | --- |
| **ID** | **Description** |
| Zip Token  (zipToken) | Used when tokenization has been turned on and an order has been approved. Before placing the order, a new token is created and saved in this attribute for the specific order. Later, the token may be used to create charges for this specific order. |

### Order

This custom preference can be added to a new “**Zip - SFRA**” group for Zip payment method.

|  |  |
| --- | --- |
| **ID** | **Description** |
| Zip Token  (zipToken) | Used when tokenization has been turned on and an order has been approved. Before placing the order, a new token is created and saved in this attribute for the specific order. Later, the token may be used to create charges for this specific order. Only for Zip AU. |
| Zip Require Approval  (zipRequireApproval) | Flag to indicate if the Zip order is linked to a customer account which requires approval. Only for Zip AU. |
| Zip Receipt Number  (ZipReceiptNumber) | Holding the Zip receipt number for this order. |
| Zip Captured Amount  (ZipCapturedAmount) | Holding captured amount for a Zip order. |
| Zip Refunded Amount  (ZipRefundedAmount) | Holding refunded amount for a Zip order. |
| Zip Checkout Id  (ZipCheckoutId) | Holding Zip Checkout Id related to this order. |

This custom preference can be added to a new “**QuadPay- SFRA**” group for QuadPay payment method.

|  |  |
| --- | --- |
| **ID** | **Description** |
| Zip Token  (zipToken) | Used when tokenization has been turned on and an order has been approved. Before placing the order, a new token is created and saved in this attribute for the specific order. Later, the token may be used to create charges for this specific order. Only for Zip AU. |
| Zip Require Approval  (zipRequireApproval) | Flag to indicate if the Zip order is linked to a customer account which requires approval. Only for Zip AU. |
| QuadPay Receipt Number  (ZipReceiptNumber) | Holding the QuadPay receipt number for this order. |
| QuadPay Captured Amount  (ZipCapturedAmount) | Holding captured amount for a QuadPay order. |
| QuadPay Refunded Amount  (ZipRefundedAmount) | Holding refunded amount for a QuadPay order. |
| QuadPay Checkout Id  (ZipCheckoutId) | Holding QuadPay Checkout Id related to this order. |

This custom preference can be added to a new “**PayFlex- SFRA**” group for PayFlex payment method.

|  |  |
| --- | --- |
| **ID** | **Description** |
| Zip Token  (zipToken) | Used when tokenization has been turned on and an order has been approved. Before placing the order, a new token is created and saved in this attribute for the specific order. Later, the token may be used to create charges for this specific order. Only for Zip AU. |
| Zip Require Approval  (zipRequireApproval) | Flag to indicate if the Zip order is linked to a customer account which requires approval. Only for Zip AU. |
| PayFlex Receipt Number  (ZipReceiptNumber) | Holding the PayFlex receipt number for this order. |
| PayFlex Captured Amount  (ZipCapturedAmount) | Holding captured amount for a PayFlex order. |
| PayFlex Refunded Amount  (ZipRefundedAmount) | Holding refunded amount for a PayFlex order. |
| PayFlex Checkout Id  (ZipCheckoutId) | Holding PayFlex Checkout Id related to this order. |

### Order Address

AU, NZ, UK, US and ZA are added as allowed countries in this system object type, so they appear as valid values for selection within the payment method’s configuration settings.

|  |  |
| --- | --- |
| **ID** | **Description** |
| Country (countryCode) | Allowed countries for billing address (see the Countries setting within the ZIP Payments payment method).  Values:   |  |  | | --- | --- | | United States | US | | France | FR | | Italy | IT | | Japan | JP | | China | CN | | United Kingdom | GB | | Australia | AU | | New Zealand | NZ | | South Africa | ZA | |

### Order Payment Instrument

The following custom attributes have been added to a “**Zip**-**SFRA**” attribute group for Zip payment method:

|  |  |
| --- | --- |
| **ID** | **Description** |
| Zip Email  (zipEmail) | This field holds the Zip email, which a customer entered when checking out with one of the available Zip payment methods. The field is shown in the order’s payment tab in BM. |
| Zip Phone  (zipPhone) | This field holds the Zip phone, which a customer entered when checking out with one of the available Zip payment methods. The field is shown in the order’s payment tab in BM. |

The following custom attributes have been added to a “**QuadPay**-**SFRA**” attribute group for QuadPay payment method:

|  |  |
| --- | --- |
| **ID** | **Description** |
| QuadPay Email  (zipEmail) | This field holds the QuadPay email, which a customer entered when checking out with the QuadPay payment method. The field is shown in the order’s payment tab in BM. |
| QuadPay Phone  (zipPhone) | This field holds the QuadPay phone, which a customer entered when checking out with the QuadPay payment method. The field is shown in the order’s payment tab in BM. |

The following custom attributes have been added to a “**PayFlex**-**SFRA**” attribute group for PayFlex payment method:

|  |  |
| --- | --- |
| **ID** | **Description** |
| PayFlex Email  (zipEmail) | This field holds the PayFlex email, which a customer entered when checking out with the PayFlex payment method. The field is shown in the order’s payment tab in BM. |
| PayFlex Phone  (zipPhone) | This field holds the PayFlex phone, which a customer entered when checking out with the PayFlex payment method. The field is shown in the order’s payment tab in BM. |

### Payment Transaction

The following custom attributes have been added to a “**Zip**-**SFRA**” attribute group for Zip payment method:

|  |  |
| --- | --- |
| **ID** | **Description** |
| Zip Charge Id  (ZipChargeId) | Holding Zip Charge ID. |
| Zip Receipt Number (ZipReceiptNumber) | Holding the Zip receipt number for this order. |
| Zip Product  (zipProduct) | Holding the Zip Product, the customer placed. |
| Is Automatic Capture  (isZipAutoCapture) | Holds the state of AutoCapture param in the capture request. |

The following custom attributes have been added to a “**QuadPay**-**SFRA**” attribute group for QuadPay payment method:

|  |  |
| --- | --- |
| **ID** | **Description** |
| QuadPay Charge Id  (ZipChargeId) | Holding QuadPay Charge ID. |
| QuadPay Receipt Number (ZipReceiptNumber) | Holding the QuadPay receipt number for this order. |
| QuadPay Product  (zipProduct) | Holding the QuadPay Product, the customer placed. |
| Is Automatic Capture  (isZipAutoCapture) | Holds the state of AutoCapture param in the capture request. |

The following custom attributes have been added to a “**PayFlex**-**SFRA**” attribute group for PayFlex payment method:

|  |  |
| --- | --- |
| **ID** | **Description** |
| PayFlex Charge Id  (ZipChargeId) | Holding PayFlex Charge ID. |
| PayFlex Receipt Number (ZipReceiptNumber) | Holding the PayFlex receipt number for this order. |
| PayFlex Product  (zipProduct) | Holding the PayFlex Product, the customer placed. |
| Is Automatic Capture  (isZipAutoCapture) | Holds the state of Auto Capture param in the capture request. |

### Profile

The following custom attributes have been added to a “**Zip-SFRA**” attribute group for Zip payment method.

*Note: For QuadPay payment method, this custom preference is added to a new “****QuadPay- SFRA****” group and for PayFlex payment method, this custom preference is added to a new “****PayFlex- SFRA****” group*.

|  |  |
| --- | --- |
| **ID** | **Description** |
| Zip Save Token  (zipSaveToken) | A flag programmatically set if the customer wants to save a Zip token. Only for Zip AU. |

## Services

An HTTP service has been added: “zip.http.defaultendpoint” with a profile “zip.http.service” and two credentials: “zip.http.credentials.production” and “zip.http.credentials.sandbox” for each API mode respectively.

## Content Assets

Two content assets have been added as part of the cartridge integration.

|  |  |
| --- | --- |
| zip-declined | This message appears when a Zip payment has been declined. |
| zip-referral-error | This message appears when checkout proceed with Zip referral account and the order became under review. |

# Installation

This section describes the contents of the archive you downloaded from Salesforce as well as the way to prepare and install the archive to support payments with Zip.

## Archive Contents

The archive contains the following folders within:

* “build” – contains front-end build tools and configuration scripts. Note that, building the cartridge front-end resources is unnecessary as they have been added to the archive as well.
* “cartridges” – contains the Zip integration cartridges.
* “cartridges/int\_zipmoney” – contains SiteGenesis code extensions.
* “cartridges/int\_zippay” – contains core code extension
* “cartridges/bm\_zippay” – contains Business Manager UI BM-related system extensions to help with QA testing and validating delayed payment flows (partial/full refund and capture of charges).
* “documentation” – contains this “IntegrationGuide.docx” document which describes integration of Zip Payments with the SiteGenesis.
* “metadata” – contains meta-data to be imported when installing the cartridge.

## Archive Preparation

Before following the installation process in this section, please copy-paste the metadata folder in a temporary directory of your choosing. Then, follow these steps to ensure the archive is prepared for site import:

Note: Please choose site-template folder based on the Zip payment method you are implementing

* If you are installing Zip payment, choose folder site-template\_Zip
* For QuadPay, choose folder site-template\_QuadPay
* For PayFlex, choose folder site-template\_PayFlex

1. Within the site-template\_Zip/sites, there is a RefArch folder. Rename the folder so its name corresponds to the name of your SFRA site onto which you want the cartridge installed. For example, if you have a site called RefArchGlobal, you must rename the RefArch folder into a new folder called RefArchGlobal in the same directory.
2. Within the site-template/libraries directory, there is the RefArchSharedLibrary folder. Copy-paste the folder into a new one whose name corresponds to the <your-library-id> content library assigned to your site.
3. Then, within the site-template/libraries/<your-library-id> directory, open library.xml file and look to find library-id="RefArchSharedLibrary" at the first opening tag. Replace that with the name of your library <your-library-id>.

Note that if you have more than one sites onto which you want the cartridge installed, you must follow the same steps and just copy-paste the folders.

## Cartridge Installation

Please, make sure the archive contents you received corresponds to the one described in “Archive Contents” and that you have followed the steps in “Archive Preparation” first.

Cartridge installation is done via following the steps laid out in the installation plan shown below:

1. Add currencies (AUD, NZD, GBP for Zip, USD for QuadPay and ZAR for PayFlex).
2. Add Locales (AU, NZ, GB, US, ZA) – the AU, NZ and GB locales need to be configured for Zip, US locale need to be configured for QuadPay and ZA locale need to be configured for PayFlex.
3. Import cartridge metadata via site import – the system integration expert is required to import metadata via “Site Import/Export” section in the business manager. Beforehand, the “sites/RefArch ” folder may need to be renamed or duplicated so configuration is applied to the right site(s), based on SiteGenesis, within the site structure of the merchant’s system.
4. Update cartridge path – for each SiteGenesis-enabled site which is going to support zip payments, the system integration expert needs to prepend zip cartridge name to the cartridge path. In other words, “int\_ zipmoney” and “int\_zippay” needs to come before “app\_storefront\_controllers” and “app\_storefront\_core” in the path.
5. Update BM cartridge path – Add “int\_zippay” and “bm\_zippay” to BM cartridge path
6. For QuadPay payment methods, replace bm\_zippay\cartridge\bm\_extensions.xml and bm\_zippay\cartridge\templates\resources\zippaybm.properties with files from bm-changes\_QuadPay
7. For PayFlex payment methods, replace bm\_zippay\cartridge\bm\_extensions.xml and bm\_zippay\cartridge\templates\resources\zippaybm.properties with files from bm-changes\_PayFlex
8. Update SiteGenesis – the base cartridges must be updated with the custom code from section “Custom Code”.

# Custom Code

## Updates on your storefont core cartridge

Please follow steps as below:

* {your\_storefront\_core}/cartridge/templates/default/components/footer/footer\_UI.isml
* {your\_storefront\_core}/cartridge/templates/default/checkout/cart/cart.isml
* {your\_storefront\_core}/cartridge/templates/default/checkout/billing/billing.isml
* {your\_storefront\_core}/cartridge/templates/default/product/components/pricing.isml
* {your\_storefront\_core}/cartridge/templates/default/checkout/billing/paymentmethods.isml
* {your\_storefront\_core}/cartridge/templates/default/checkout/summary/summary.isml
* {your\_storefront\_core}/cartridge/templates/default/components/order/orderdetailsemail.isml
* {your\_storefront\_core}/ cartridge/templates/default/components/order/orderdetails.isml
* {your\_storefront\_core}/cartridge/templates/default/account/payment/paymentinstrumentlist.isml
* {your\_storefront\_core}/cartridge/templates/default/account/orderhistory/orders.isml
* {your\_storefront\_core}/cartridge/cartridge/forms/default/profile.xml

### {your\_storefront\_core}/cartridge/templates/default/components/footer/footer\_UI.isml



Code updated:

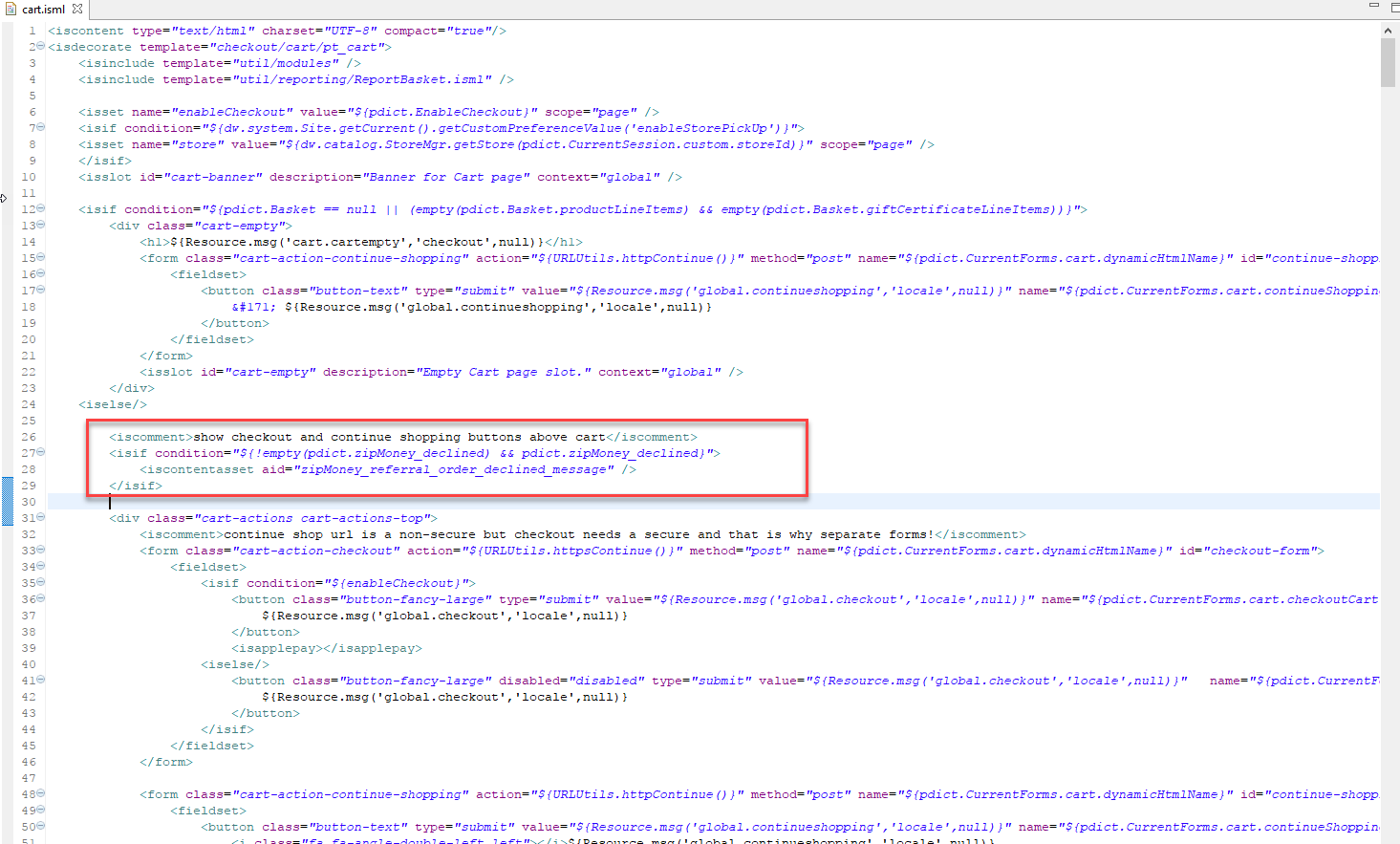
<!-- zipMoney Checkout Scripts -->

<isinclude template = "zipMoney/checkout/zipMoney\_checkout" />

<!-- zipMoney Widget API -->

<isinclude template = "zipMoney/widgets/zipMoney\_widget\_api"/>

### {your\_storefront\_core}/cartridge/templates/default/checkout/cart/cart.isml



Code updated:

<iscomment>show checkout and continue shopping buttons above cart</iscomment>

<isif condition="${!empty(pdict.zipMoney\_declined) && pdict.zipMoney\_declined}">

<iscontentasset aid="zipMoney\_referral\_order\_declined\_message" />

</isif>

### {your\_storefront\_core}/cartridge/templates/default/checkout/billing/billing.isml



Code updated:

<isif condition="${session.privacy.zipMoneyDeclined}">

<div class="error-form">

<iscontentasset aid="zip-declined" />

</div>

<isscript>

var Transaction = require('dw/system/Transaction');

Transaction.wrap(function(){

session.privacy.zipMoneyDeclined = null;

});

</isscript>

</isif>

<isif condition="${session.privacy.zipMoneyReferred}">

<div class="error-form">

<iscontentasset aid="zip-referral-error" />

</div>

<isscript>

var Transaction = require('dw/system/Transaction');

Transaction.wrap(function(){

session.privacy.zipMoneyReferred = null;

});

</isscript>

</isif>

### {your\_storefront\_core}/cartridge/templates/default/product/components/pricing.isml

****

**Code updated:**

<isset name="CurrencyCode" value="${session.getCurrency().getCurrencyCode()}" scope="pdict" />

<isif condition="${pdict.ProductMinPrice > 0 && pdict.ProductMaxPrice > 0}">

<div class="pricerange"><isprint value="${dw.util.StringUtils.formatMoney(dw.value.Money(pdict.ProductMinPrice, pdict.CurrencyCode))}" /> - <isprint value="${dw.util.StringUtils.formatMoney(dw.value.Money(pdict.ProductMaxPrice, pdict.CurrencyCode))}" /></div>

<iselseif condition="${pdict.ProductSetSalesPrice > pdict.ProductSetStandardPrice}">

<div class="standardprice"><isprint value="${dw.util.StringUtils.formatMoney(dw.value.Money(pdict.ProductSetStandardPrice, pdict.CurrencyCode))}" /></div>

<isset name="zipMoney\_amount" value="${pdict.ProductSetStandardPrice.value}" scope="page"/>

<iselseif condition="${pdict.ProductSetStandardPrice >= pdict.ProductSetSalesPrice}">

<div class="salesprice"><isprint value="${dw.util.StringUtils.formatMoney(dw.value.Money(pdict.ProductSetSalesPrice, pdict.CurrencyCode))}" /></div>

<isset name="zipMoney\_amount" value="${pdict.ProductSetSalesPrice.value}" scope="page"/>

</isif>

<isif condition="${ShowStandardPrice}">

<span class="price-standard">

<isif condition="${StandardPrice.valueOrNull != null && StandardPrice.valueOrNull > 0}">

<isprint value="${StandardPrice}"/><iselse/>${Resource.msg('product.pricing.noprice','product',null)}

<isset name="zipMoney\_amount" value="${StandardPrice}" scope="page"/>

</isif>

</span>

</isif>

<iscomment>Let promotional price of 0 display as is, in this case isPromoPrice is true</iscomment>

<span class="price-sales">

<isif condition="${(SalesPrice.valueOrNull != null && SalesPrice.valueOrNull > 0) || isPromoPrice}">

<isprint value="${SalesPrice}"/>

<isset name="zipMoney\_amount" value="${SalesPrice}" scope="page"/>

<iselse/>

${Resource.msg('product.pricing.noprice','product',null)}</isif>

</span>

<isscript>

var TemplateUtils = require('\*/cartridge/scripts/util/template');

</isscript>

<isif condition=*"${TemplateUtils.isZippayEnabled() && TemplateUtils.isShowZipWidgets()}"*>

<isslot id=*"zip-pdp"* description=*"Zip PDP"* context=*"global"* />

</isif>

### {your\_storefront\_core}/cartridge/templates/default/checkout/billing/paymentmethods.isml

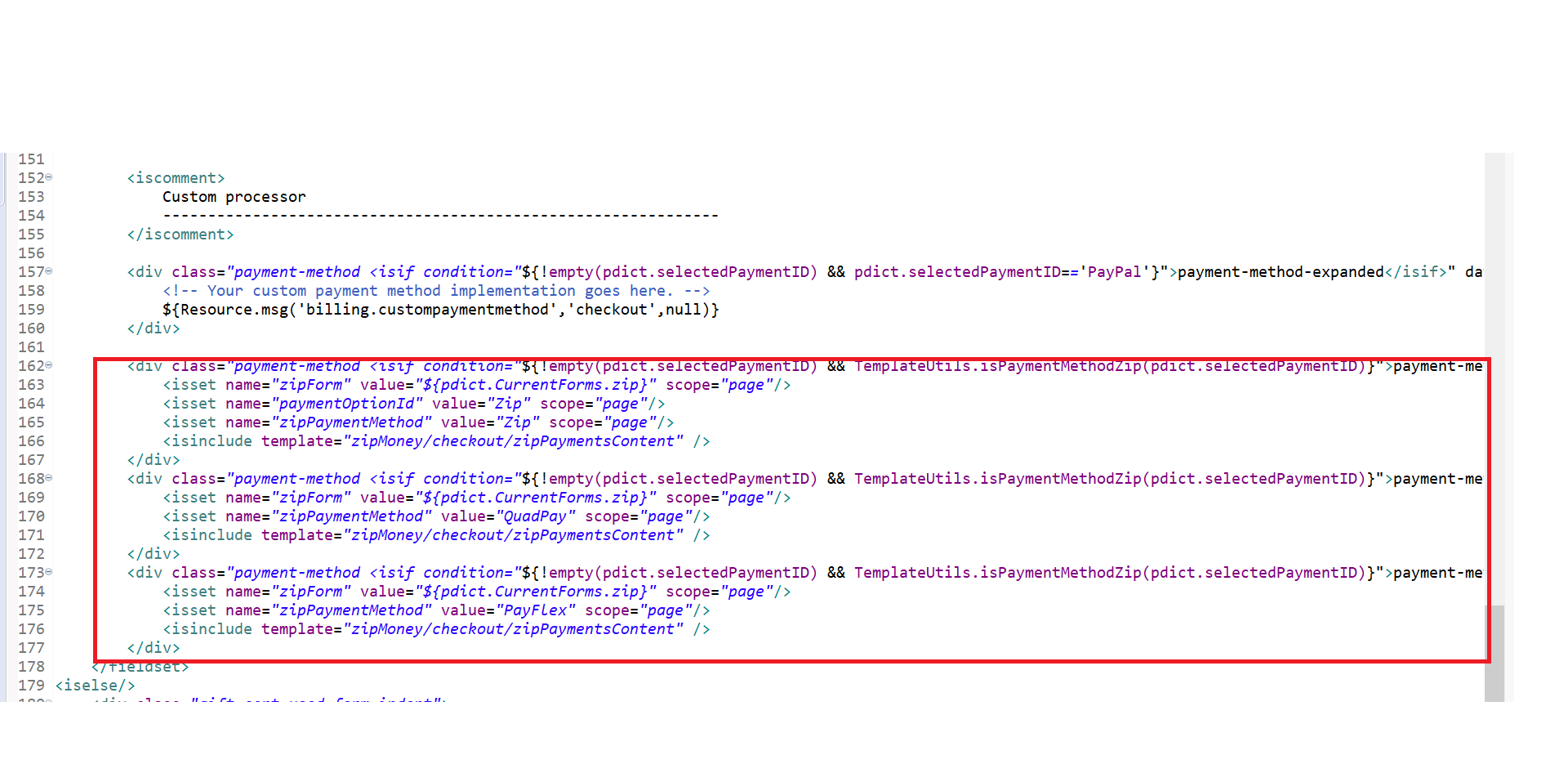


First, add zipMoney.css and place the <isscript> code snippet as shown in the screenshot above:

|  |
| --- |
| <link rel="stylesheet" href="${URLUtils.staticURL('/css/zipMoney.css')}" />  <isscript>  var TemplateUtils = require('\*/cartridge/scripts/util/template');  </isscript> |

Then, replace the label’s inner HTML with the following code:

|  |
| --- |
| <label for="is-${radioID}">  <isif condition="${TemplateUtils.isZippayEnabled() && TemplateUtils.isPaymentMethodZip(radioID) && TemplateUtils.getPaymentMethodLogoUrl(paymentMethodType.value)}">  <img class="${radioID}-logo" src="${TemplateUtils.getPaymentMethodLogoUrl(paymentMethodType.value)}" alt="${Resource.msg(paymentMethodType.label, 'forms', null)}">  <iselse>  <isprint value="${Resource.msg(paymentMethodType.label,'forms',null)}"/>  </isif>  </label> |



Last, the following code snipper must be inserted after the last *div* with class payment-method within the fieldset:

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

<isset name="zipForm" value="${pdict.CurrentForms.zip}" scope="page"/>

<isset name="paymentOptionId" value="Zip" scope="page"/>

<isset name="zipPaymentMethod" value="Zip" scope="page"/>

<isinclude template="zipMoney/checkout/zipPaymentsContent" />

</div>

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

<isset name="zipForm" value="${pdict.CurrentForms.zip}" scope="page"/>

<isset name="zipPaymentMethod" value="QuadPay" scope="page"/>

<isinclude template="zipMoney/checkout/zipPaymentsContent" />

</div>

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

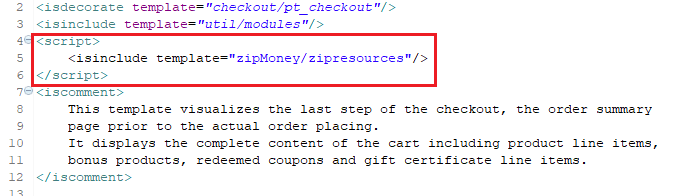
<isset name="zipForm" value="${pdict.CurrentForms.zip}" scope="page"/>

<isset name="zipPaymentMethod" value="PayFlex" scope="page"/>

<isinclude template="zipMoney/checkout/zipPaymentsContent" />

</div>

### {your\_storefront\_core}/cartridge/templates/default/checkout/summary/summary.isml



First, place the <isscript> code snippet as shown in the screenshot above:

<script>

<isinclude template="zipMoney/zipresources"/>

</script>



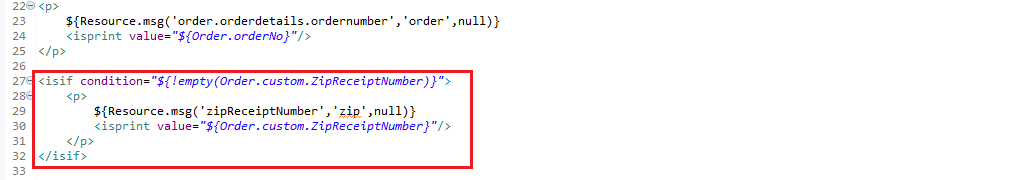
Replace the “submit-order” form opening tag with the following code:

|  |
| --- |
| <isscript>  var TemplateUtils = require("\*/cartridge/scripts/util/template");  </isscript>  <form action="${TemplateUtils.getPlaceOrderFormUrl()}" method="post" class="submit-order"> |

Replace the code for the Place Order button with:

|  |
| --- |
| <isif condition="${TemplateUtils.isZipPaymentMethodSelected(pdict.Basket) && !TemplateUtils.canTokenize(pdict.Basket)}">  <button id="zip-checkout" class="button-fancy-large continue-with-zip" type="button" name="submit" value="">  ${Resource.msg('continueToZip', 'zip', null)}  </button>  <iselse>  <button class="button-fancy-large" type="submit" name="submit" value="${Resource.msg('global.submitorder','locale',null)}">  ${Resource.msg('global.submitorder','locale',null)}  </button>  </isif> |

### {your\_storefront\_core}/ cartridge/templates/default/components/order/orderdetailsemail.isml



Add the following code to this template:

<isif condition="${!empty(Order.custom.ZipReceiptNumber)}">

<p>

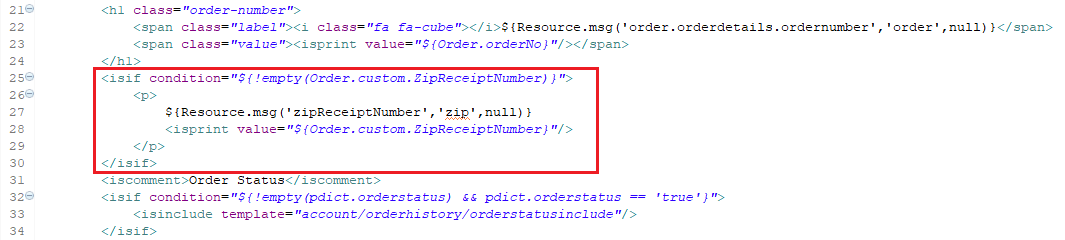
${Resource.msg('zipReceiptNumber','zip',null)}

<isprint value="${Order.custom.ZipReceiptNumber}"/>

</p>

</isif>

### {your\_storefront\_core}/ cartridge/templates/default/components/order/orderdetails.isml



Add the following code to this template:

<isif condition="${!empty(Order.custom.ZipReceiptNumber)}">

<p>

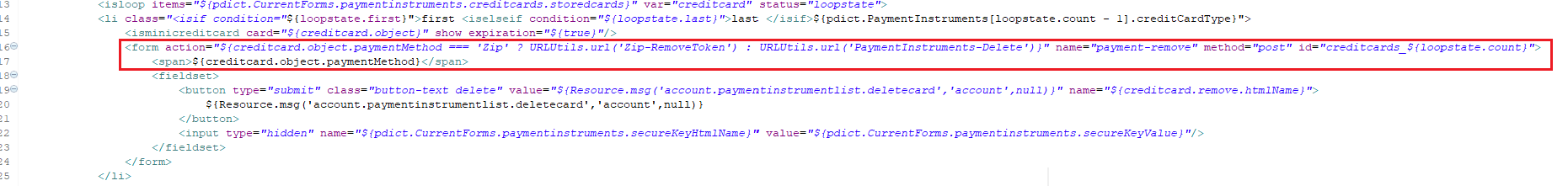
${Resource.msg('zipReceiptNumber','zip',null)}

<isprint value="${Order.custom.ZipReceiptNumber}"/>

</p>

</isif>

### {your\_storefront\_corecartridge/templates/default/account/payment/paymentinstrumentlist.isml



Add the following code to this template:

<form action="${creditcard.object.paymentMethod === 'Zip' ? URLUtils.url('Zip-RemoveToken') : URLUtils.url('PaymentInstruments-Delete')}" name="payment-remove" method="post" id="creditcards\_${loopstate.count}">

<span>${creditcard.object.paymentMethod}</span>

### {your\_storefront\_core}/cartridge/templates/default/account/orderhistory/orders.isml



Add the following code to this template:

<isif condition="${!empty(order.object.custom.ZipReceiptNumber)}">

<div class="order-zipreceipt">

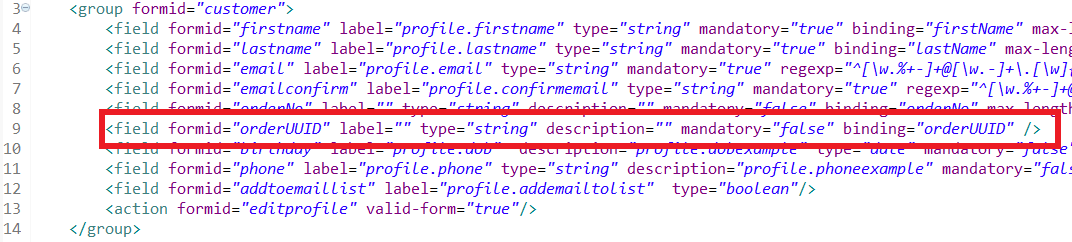
<span class="label">${Resource.msg('zipReceiptNumber','zip',null)}:</span>

<span class="value"><isprint value="${order.object.custom.ZipReceiptNumber}" /></span>

</div>

</isif>

### {your\_storefront\_corecartridge/cartridge/forms/default/profile.xml



Add the following code to this template:

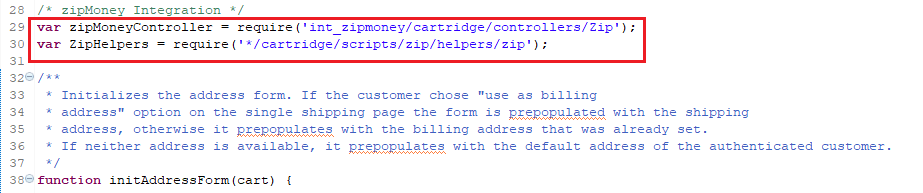
<field formid="orderUUID" label="" type="string" description="" mandatory="false" binding="orderUUID" />

## Updates on your storefront controller cartridge

* **{your-controller-cartridge}/cartridge/controllers/COBilling.js**
* **{your-controller-cartridge}/cartridge/controllers/Cart.js**
* **{your-controller-cartridge}/cartridge/controllers/PaymentInstruments.js**
* **{your-controller-cartridge}/cartridge/controllers/COSummary.js**

### ****{your-controller-cartridge}/cartridge/controllers/COBilling.js****

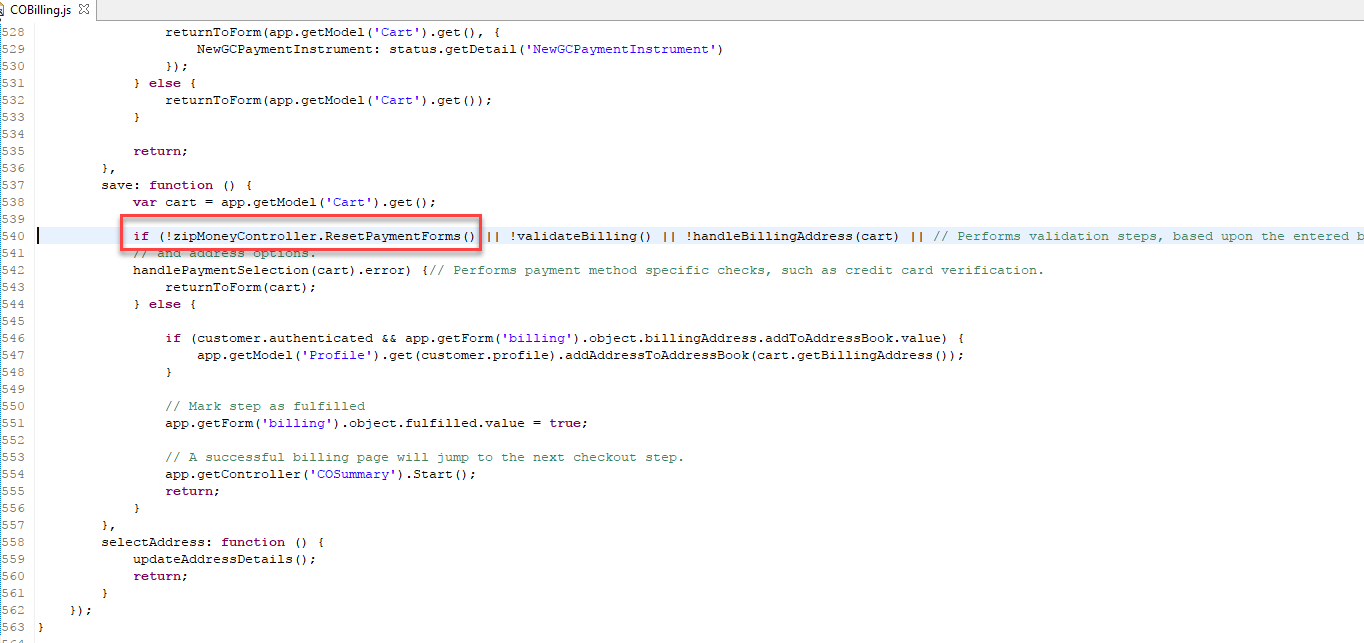
First, please add a reference of ZipMoney controller to the top of the **COBilling** controller as below.



Code updated:

|  |
| --- |
| /\* zipMoney Integration \*/  var zipMoneyController = require('int\_zipmoney/cartridge/controllers/Zip');  var ZipHelpers = require('\*/cartridge/scripts/zip/helpers/zip'); |

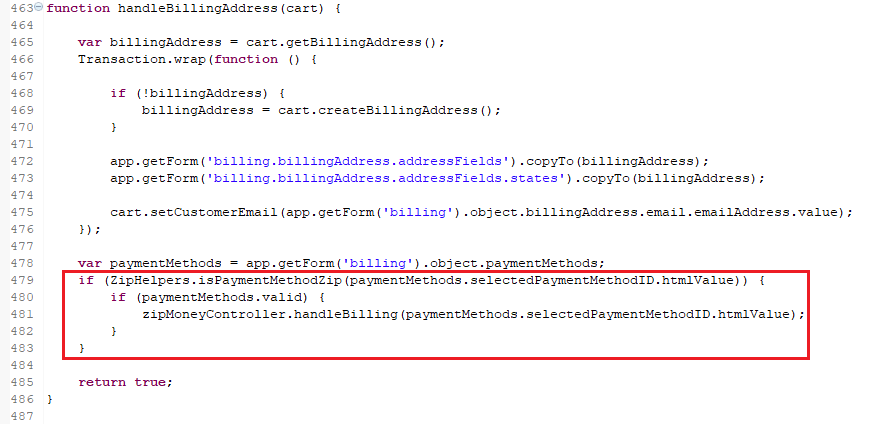
Then, update the logic code as below screenshot:



Code updated:

|  |
| --- |
| !zipMoneyController.ResetPaymentForms |

Then, update the logic in handleBillingAddress function as below screenshot:



Code Updated:

**if** (ZipHelpers.isPaymentMethodZip(paymentMethods.selectedPaymentMethodID.htmlValue)) {

**if** (paymentMethods.valid) {

zipMoneyController.handleBilling(paymentMethods.selectedPaymentMethodID.htmlValue);

}

}

Then, update the logic in start function as below screenshot:

****

Code Updated:

var CustomerTokenInWalletModel = require('\*/cartridge/models/zip/customerTokenInWallet');

var TempateUtils = require('\*/cartridge/scripts/util/template');

var zippayEnabled = TempateUtils.isZippayEnabled();

//update profile custom attribute ZipSaveToken as false if customer cancelled payment.

if (zippayEnabled && session.privacy.zipMoneyDeclined) {

var customerProfile = customer.getProfile();

var customerTokenInWalletModel = new CustomerTokenInWalletModel(customerProfile.getWallet());

if (customerProfile && customer.authenticated && !customerTokenInWalletModel.hasToken()) {

Transaction.wrap(function () {

customerProfile.custom.ZipSaveToken = false;

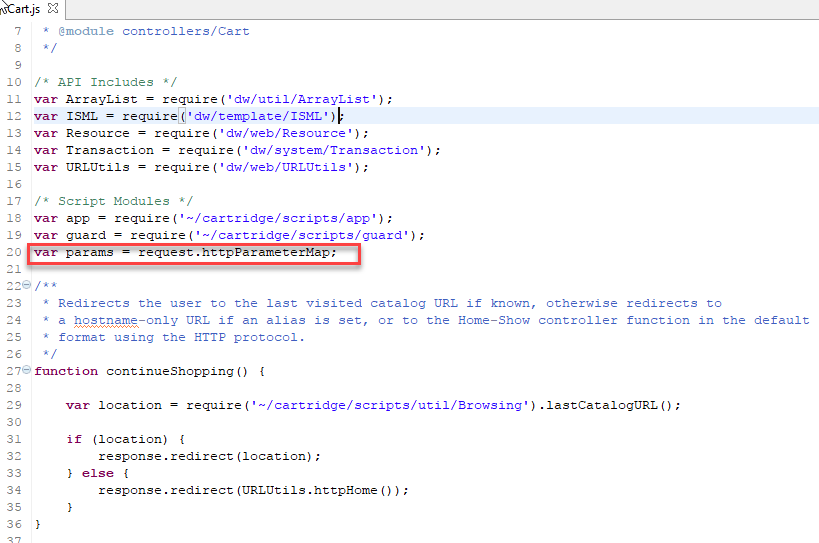
});

}

}

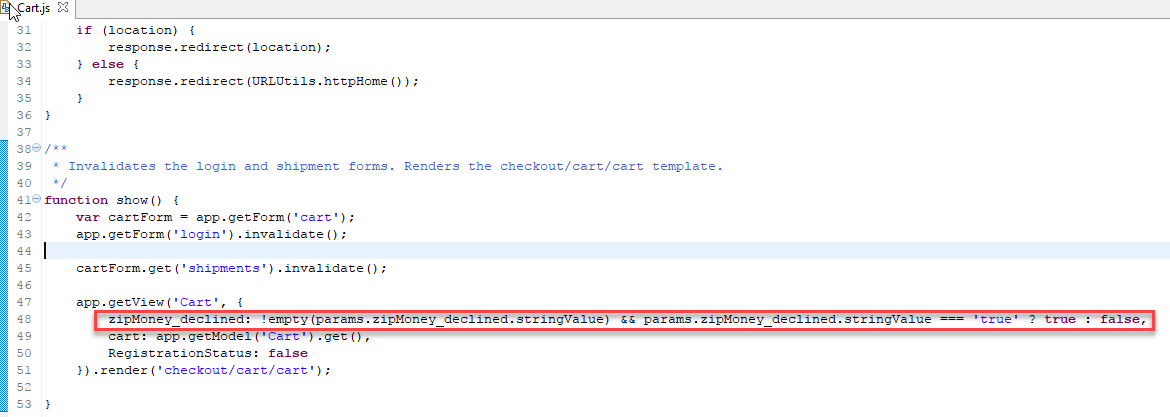
### ****{your-controller-cartridge}/cartridge/controllers/Cart.js****

First, please define a HTTP Parameter Map as below:



Code updated:

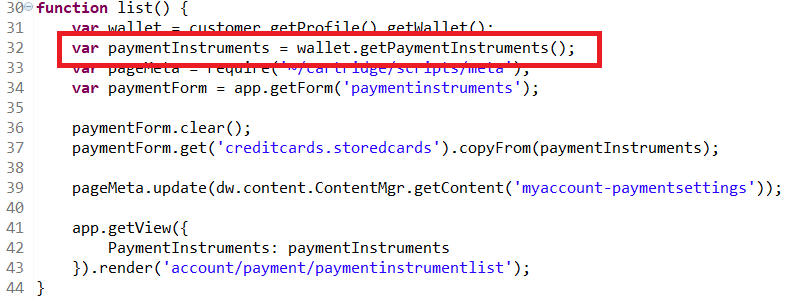
|  |
| --- |
| **var params = request.httpParameterMap;** |



Add the zipMoney\_declined parameter:

zipMoney\_declined: !empty(params.zipMoney\_declined.stringValue) && params.zipMoney\_declined.stringValue === 'true' ? true : false,

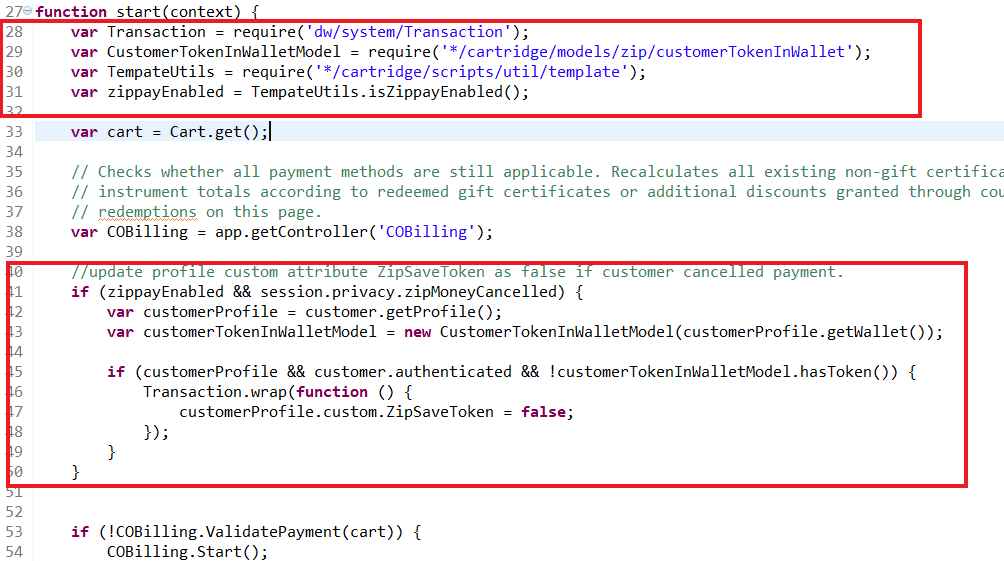
### ****{your-controller-cartridge}/cartridge/controllers/PaymentInstruments.js****



Remove the *dw.order.PaymentInstrument.METHOD\_CREDIT\_CARD* parameter from getPaymentInstruments() for displaying all saved card.

var paymentInstruments = wallet.getPaymentInstruments();

### ****{your-controller-cartridge}//cartridge/controllers/COSummary.js****



Code updated:

|  |
| --- |
| **var** Transaction = require('dw/system/Transaction');  **var** CustomerTokenInWalletModel = require('\*/cartridge/models/zip/customerTokenInWallet');  **var** TempateUtils = require('\*/cartridge/scripts/util/template');  **var** zippayEnabled = TempateUtils.isZippayEnabled();    //update profile custom attribute ZipSaveToken as false if customer cancelled payment.  **if** (zippayEnabled && session.privacy.zipMoneyCancelled) {  **var** customerProfile = customer.getProfile();  **var** customerTokenInWalletModel = **new** CustomerTokenInWalletModel(customerProfile.getWallet());  **if** (customerProfile && customer.authenticated && !customerTokenInWalletModel.hasToken()) {  Transaction.wrap(**function** () {  customerProfile.custom.ZipSaveToken = **false**;  });  }  } |
|  |

# Configuration

This section of the integration guide describes how merchants can configure the features of this integration.

## Configure Zip Payments Processor

#### **Enable/Disable Zip**

To configure this feature, the merchant needs to use the setting located in “Merchant Tools – Custom Preferences – Zip – Enable Zip” for Zip payment.

#### **Enable/Disable PayFlex**

To configure this feature, the merchant needs to use the setting located in “Merchant Tools – Custom Preferences – PayFlex – Enable PayFlex” for PayFlex payment.

#### **Enable/Disable QuadPay**

To configure this feature, the merchant needs to use the setting located in “Merchant Tools – Custom Preferences – QuadPay – Enable QuadPay” for QuadPay payment.

### Activate Production Mode

To configure this feature, the merchant needs to use the setting located in “Merchant Tools – Custom Preferences – Zip – Zip API Mode Production”.

(when the merchant is ready for production, the API Key also needs to be updated.)

To configure this feature, the merchant needs to use the setting located in “Merchant Tools – Custom Preferences – PayFlex – PayFlex API Mode Production” for PayFlex payment.

(when the merchant is ready for production, the API Key also needs to be updated.)

To configure this feature, the merchant needs to use the setting located in “Merchant Tools – Custom Preferences – QuadPay – QuadPay API Mode Production” for QuadPay payment.

(when the merchant is ready for production, the API Key also needs to be updated.)

### Set API Key

To configure this feature, the merchant needs to use the setting located in “Merchant Tools – Custom Preferences – Zip – Zip API Mode Production”.

To configure this feature, the merchant needs to use the setting located in “Merchant Tools – Custom Preferences – PayFlex – PayFlex API Mode Production” for PayFlex payment.

To configure this feature, the merchant needs to use the setting located in “Merchant Tools – Custom Preferences – QuadPay – QuadPay API Mode Production” for QuadPay payment.

### Enable/Disable Automatic Capture

To configure this feature, the merchant needs to use the setting located in “Merchant Tools – Custom Preferences – Zip – Automatic Capture”.

### Enable/Disable Zip Tokenization

To configure this feature, the merchant needs to use the setting located in “Merchant Tools – Custom Preferences – Zip – Zip Tokenization”.

## Configure Zip Payment Method

### Update Payment Method Logo

The Zip payment method logo can be updated.

The setting is in “Merchant Tools – Ordering – Payment Methods”. On the payment method screen, choose “Zip”. Then, use the “Image” setting and upload a new logo.

### Configure PayFlex Payment Method

The PayFlex payment method logo can be updated.

The setting is in “Merchant Tools – Ordering – Payment Methods”. On the payment method screen, choose “PayFlex”. Then, use the “Image” setting and upload a new logo.

### Configure QuadPay Payment Method

The QuadPay payment method logo can be updated.

The setting is in “Merchant Tools – Ordering – Payment Methods”. On the payment method screen, choose “QuadPay”. Then, use the “Image” setting and upload a new logo.

### Allowed Countries

The setting is in “Merchant Tools – Ordering – Payment Methods”. On the payment method screen, choose “Zip” and then use the “Countries” setting to configure allowed countries. Zip payment method will only be available for those countries you select.

### Min/Max range for each currency

The setting is in “Merchant Tools – Ordering – Payment Methods”. On the payment method screen, choose “Zip” and then use the “Min/Max Payment Ranges” setting for each currency.

## Configure Zip Checkout

### Configure Zip Payment Declined Message

The message appearing on payment stage when Zip payment is declined is configurable via a content asset: “zip-declined”.

## Configure Thank You Page (Order Confirmation)

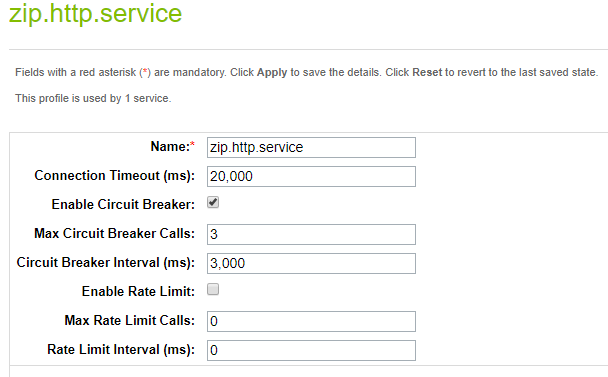
#### **Configure Zip Referral Order Error Message**

The message appearing on payment stage when the Zip referral order is cancelled. It is configurable via a content asset: “zip-referral-error”.

## Configure Zip API Client

### Configure Retries in case of General Errors

In case of general errors, the Zip API Client will retry requests according to configuration values specified in the Zip Service Profile. To access those configuration values, navigate to “Administration – Operations – Services” and then select “Profiles” tab. Choose the “zip.http.service” profile.



The **Max Circuit Breaker Calls** determine the number of attempts the Service is going to make in case of general errors. The **Circuit Breaker Interval** determine the interval in which those attempts will be made.

General errors are:

* Issues at initial connection.
  + Unknown host
  + Connection timeout
  + Connection refused
* Protocol-specific errors, such as HTTP 500 - internal server error or HTTP 503 - service unavailable.
* Exceptions explicitly thrown as part of the service call

# Customizations and Extensions

## Use Back-End Hooks To Capture and Refund Amounts

To implement capture, please use this line of code:

var hooksHelper = require('\*/cartridge/scripts/zip/helpers/hooks');

hooksHelper ('zippay.order.processor', 'Capture', orderNo, amount, orderProcess.Capture );

To implement refund, please use this line of code:

var hooksHelper = require('\*/cartridge/scripts/zip/helpers/hooks');

hooksHelper ('zippay.order.processor', 'Refund', orderNo, amount, reason, orderProcess.Refund );

# Operations, Maintenance

## Availability

In case of problems with the connection to ZipMoney please contact the ZipMoney support department.

Please supply as much information as possible (Merchant account, time, order number, ZipMoney reference etc.). Also, please check the log file mentioned above.

## Support

In case of problems with the integration, missing features, etc. please contact the ZipMoney support department or your ZipMoney account manager.

# Known Issues

None.

# Release History

|  |  |  |
| --- | --- | --- |
| **Version** | **Date** | **Changes** |
| 19.1.0 |  | Initial release |
| 21.1.0 | 30-Nov-2020 | 1. Added new payment methods - QuadPay for US regions and Payflex for South Africa region 2. Removed Zip footer content asset and logo 3. Moved common functionality to a core cartridge, int\_zippay |
| 21.1.1 | 24-May-2021 | Updated meta file to update default value of site preference ‘zipLightbox’ to false |