



# Integration Document

SiteGenesis

**Version 22.2.1**



## Contents

1. Overview .....	4
2. Features .....	4
2.1. Payment Modes .....	5
2.1.1. Limepay One Time Payment .....	5
2.1.2. Limepay Pay Later Payment .....	5
2.1.3. Supported Payment Types .....	5
2.2. Limepay Payment Widget .....	6
2.3. Supported API .....	6
3. Limitations, Constraints .....	6
4. Privacy, Payment .....	7
4. Compatibility .....	8
5. Installation .....	8
5.1. Cartridges .....	8
5.2. Metadata .....	8
5.2.1. Services .....	8
5.2.2. Payment Processor .....	9
5.2.3. Payment Methods .....	9
5.2.4 Customer Groups .....	9
5.3. Google Phone Number Library – Node.js extension .....	10
6. Configurations .....	11
6.1. Configuring Site Preferences .....	11
6.2. Configuring Payment Methods .....	14
7. Storefront Code Changes .....	15
7.1. Controller Changes .....	15
7.2. Models Changes .....	21
7.3. Client Side JS Changes .....	23
7.4. Custom Files .....	32
7.5. ISML Changes .....	32
8. User Guide .....	41
8.1. Custom CSS .....	41
8.2. Limepay Widget .....	42
8.2.1. Toggle Widget .....	42
8.2.2. Textual Widget .....	44
8.2.3. Product Widget .....	44
8.2.4. Cart Widget .....	48

8.2.5. Checkout Content & Stages .....	52
8.2.6. Customer Applicable Checkout View .....	57
8.2.7. Payment Method Description .....	61
8.2.8. Changing Billing Email / Phone .....	62
8.2.9. Limepay Control Switch .....	62
9. Refunds & Backend Operations .....	63
9.1. Refunds .....	63
9.2. Payment Flow Processes.....	63
10. Testing.....	65
10.1. Checkout with One Time Payment .....	66
10.2. Checkout with Pay Later Payment .....	72
10.3. Test Cards.....	78
11. Operation, Maintenance.....	78
11.1. Availability.....	78
11.2. Failover / Recovery Process .....	78
11.3.Support .....	78
12. Appendix .....	78
12.1. System Extensions.....	78
12.1.1. Order.....	79
12.1.2. PaymentTransaction .....	79
12.1.3. Basket.....	79
12.1.4. Site Preference.....	79
12.1.5. Category.....	80
12.1.6. Product.....	80
12.2. Locating Code Changes .....	80
13. Known Issues.....	82
14. Release History.....	82

## 1. Overview

The Limepay payment cartridge enables commerce cloud to integrate with the Limepay payment service. It offers only non hosted checkout service to storefront. The purpose of the document is to guide through an easy installation of Limepay payment cartridge onto a commerce cloud store.

The integration is based on the SiteGenesis demo store, provided by SFCC.

The integration consists of an archive with contents as described in the below table.

Name	Purpose
cartridges	Contains Limepay integration cartridges
cartridges\int_limepay_sg	This cartridge contains the SiteGenesis specific changes required for Limepay integration
cartridges\int_limepay	This cartridge contains the API calls for Limepay integration and common code
cartridges\limepay_sg_changes	This cartridge references all the changes done to OOB SiteGenesis files. Please do not include this in the cartridge path and this is for reference purposes only
metadata	Contains system object extensions and configurations required for the integration
documentation	Contains this document "Limepay SiteGenesis Integration Document"

## 2. Features

The integration described in this document supports the following features:

- Limepay one time settlement and pay later instalment payment for checkout
- Pay later payment in 4 instalments over a frequency of 2 weeks each
- Adjustable initial amount and instalment due dates
- Limepay payment widgets on product and cart displays breakdown of prices for one time and pay later options
- Service for transactions in AUD currency
- Ability to enable/disable one time payments for store encouraging only pay later payments
- Ability to adjust the threshold value for which pay later payments would be allowed for store
- Ability for customer to select Limepay payment mode for checkout from product / cart pages through widget interaction
- Ability to disallow certain products or category from pay later payment
- Merchant refunds
- Ability to perform 3DS transactions.

## 2.1. Payment Modes

A store can be configured to allow any one of the following Limepay payment options for checkout :

- One time and pay later
- Pay later only

### 2.1.1. Limepay One Time Payment

Limepay one time settlement payment allows customer to pay the whole order total in one go entering card details (Refer section 2.1.3 for supported cards) and successfully place the order. Payment details are entered in a Limepay iFrame provided under the payment method section. Merchant can customize the iFrame through cartridge supported file system. One time payments can be disabled throughout for any store through configuration thereby allowing only pay later payment.

### 2.1.2. Limepay Pay Later Payment

Limepay pay later payment allows customer to pay the order amount in total of 4 instalments each defaulting to a value of one fourth the order amount due over a period of 2 weeks. However, Limepay provides the ability for the customer to change the first instalment amount up to a certain extent (as per Limepay threshold rules) and have the subsequent instalments and their due dates adjusted. Products and categories can be configured to be disabled for pay later payment at checkout. Merchant also has the control over the order amount threshold allowing pay later payment. Following scenarios disqualifies a customer from pay later payment mode at checkout :

- At least one pay later disabled product in cart
- Order total exceeds the pay later threshold range\*

\* Limepay configures the pay later threshold range for a merchant used for checkout. (Refer point 4 under section 8.2.4)

### 2.1.3. Supported Payment Types

Supported cards by Limepay:

- Visa
- Mastercard
- American Express

Limepay also provides support to 3D Secure cards

For any other supported cards please check with Limepay

## 2.2. Limepay Payment Widget

Payment widget for product and cart pages provides the customer a breakdown information of the price between one time and pay later payment modes and thereby providing a means for the customer to opt between the 2 modes for checkout. Customers get to know the instalments at product total or cart total level. Widget gets a toggle state when both payment modes are configured for the store as mentioned in the above section, allowing the customer to interact and he/she can preselect a particular payment mode for checkout from product or cart pages. User toggled state of the widget is remembered throughout the user session among product/cart widgets and checkout pages. Widget gets a simple textual state and calls out the instalment breakdown when one time is disabled and pay later is only configured for the store. However, cart and product pages can be configured by merchant to independently display either of the textual or toggle widget when payment mode selected for the site is one time and pay later. Whereas when payment mode is configured for only pay later, textual widget takes precedence and toggle widget shall not be seen anywhere across the storefront user flow.

## 2.3. Limepay 3DS

Limepay 3DS payments allow brands to perform 3DS transaction via Limepay payment methods. When place order on review page, an API call made to Limepay for creating payment and SFCC pass the 3DS true in API request and Limepay Payment API return a 403 response along a payment action which requires to perform 3DS challenge by customer. In this case an Popup widget appears on front end rendered by Limepay and customer have to go through all 3DS verification in Limepay 3DS Popup widget. Limepay either return error or success after verifying the 3DS that leads to place the order in case of success and show the error in case of any 3DS challenge failure.

## 2.4. Supported API

API	Description	Relative Endpoint
Create Order	Creates a new order at Limepay system during payment authorization	/orders
Pay Order	Pay for an order using existing payment token	/orders/{limepay_order_id}/pay
Create Refund	Create a new refund for Limepay transaction	/refunds
Upsert Customer	Create a new Limepay customer	/customers/plugin
Signin Customer	Sign in Limepay Customer With Customer ID and generate customer token	/authn/tenants/accounts:signinCustomerWithCustomerId

Also refer section 5.2.1. for complete endpoint details

## 3. Limitations, Constraints

Limepay service supports only transactions in AUD currency. Any other site currency is not supported and or need to be checked with Limepay service team.

## 4. Privacy, Payment

This integration requires access to the following customer data elements: Billing Address, Shipping Address, Order Details and Customer Profile.

No credit card details are stored within SFCC using this integration.

## 4. Compatibility

Compatible with Commerce Cloud Platform Release 21.5, Site Genesis 105.1.1.

## 5. Installation

### 5.1. Cartridges

Upload the following cartridges to the code version in Business Manager

- int\_limepay
- int\_limepay\_sg

Configure the cartridge path as shown below by adding to the beginning of the cartridge path

<b>Site Cartridge Path</b>	int_limepay_sg:int_limepay
----------------------------	----------------------------

### 5.2. Metadata

All BM configurations related to the below components have been configured within **metadata/site-template** folder

- System object extensions
- Services
- Payment Processors
- Payment Methods
- Customer Groups

Follow the below steps to import the BM configurations for the above-mentioned components

1. Locate the folder **metadata** in the installation package
2. Review the contents within **site-template** folder
3. Change the site ID under path **site-template\sites** to merchant site ID
4. Archive the folder to 'site-template.zip' and the Import the file via Site Import & Export

#### 5.2.1. Services

New service 'limepay.http.service' has been added as part the integration

Service ID	Service Profile	Service Credential
limepay.http.service	limepay.http.profile	limepay.http.credentials

Map the URLs for the sandbox and production endpoints as follows

Environment	URL
Non-production instances	<a href="https://api.sandbox.limepay.com.au">https://api.sandbox.limepay.com.au</a>
Production instances	<a href="https://api.limepay.com.au">https://api.limepay.com.au</a>

Service credentials requires no user/password rather refers the public/secret key combinations mapped to site preferences.

**Note** : Do not use hyphenated hostnames to access Open Commerce API (OCAPI) or Storefront while setting up the URL for the calls made internally for OCAPI and Storefront. Instead of use vanity host names such as brand.com , [www.brand.com](http://www.brand.com) etc.

#### 5.2.2. Payment Processor

New payment processor used by the cartridge

Payment Processor ID	Purpose
LIMEPAY	Unified payment processor for processing Limepay one time and pay later transactions

#### 5.2.3. Payment Methods

Two new payment methods are used by the cartridge

Payment Method ID	Purpose
Limepay	Main payment method used for Limepay one time and pay later checkout
Limepay_instalment	Dummy payment method used for configuring pay later threshold range. Not used for placing orders.

#### 5.2.4 Customer Groups

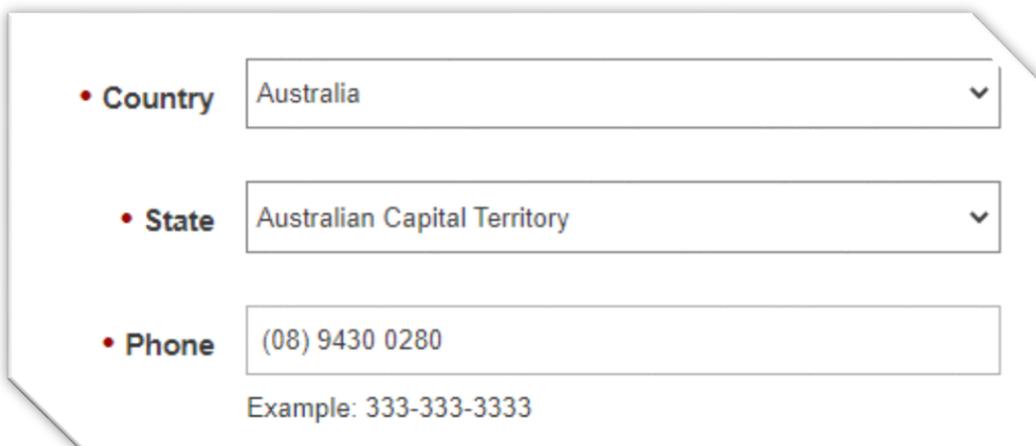
New dynamic customer group *'ExcludeLimepayPayLater'* is used by the cartridge based on user storefront custom session attribute. When a customer adds at least one pay later disabled product to cart he/she gets added to this customer group thereby being disqualified for pay later option at checkout.

### 5.3. Google Phone Number Library – Node.js extension

Limepay checkout API accepts customer billing country and phone number as input for rendering the checkout iFrame. A Node.js extension package *'google-libphonenumber'* is used by the cartridge to parse these two customer fields resulting in E164 formatted phone number as required by the API. For this add the node package *'google-libphonenumber'* to your project package.json dev dependencies and build

```
..
"author":
"devDependencies": {
  "google-libphonenumber": "^3.2.20",
  "node-sass": "^4.11.0",
  "postcss-loader": "^2.1.5",
  "eslint": "^4.19.1",
```

For example, customer selects billing country as *Australia* and enters billing phone number as shown in below image, the library then parses it to E164 format output as *'+61883942707'* where *'+61'* is mapped from billing country and the remaining digits from the actual billing phone field. The formatted number is then passed to Limepay checkout iFrame for rendering and proceeding with checkout by customer. Note, One Time Password (OTP) SMS are generated by Limepay systems onto this phone number while proceeding with pay later payment option and is required to be authenticated by the storefront customer before successfully placing order



The image shows a form with three main sections, each with a red bullet point label and a text input field with a dropdown arrow on the right:

- Country**: The input field contains "Australia".
- State**: The input field contains "Australian Capital Territory".
- Phone**: The input field contains "(08) 9430 0280".

Below the Phone field, there is an example text: "Example: 333-333-3333".

For more details on *'google-libphonenumber'* library, [see](#)

## 6. Configurations

The below sections explain how to set up the Limepay integration related configurations within Business Manager

### 6.1. Configuring Site Preferences

Refer to the following table for the different site preferences used by Limepay integration

Site Preference	Value	Description
Limepay Enabled	Yes	Centralized control to turn on/off Limepay payments for the site
Limepay Client Public Key	<i>Merchant's public key</i> <i>Limepay</i>	Key for rendering Limepay payment iFrame at checkout page under payment method section, used by Limepay checkout API. Also utilized for acquiring payment tokens for completing an order
Limepay Client Secret Key	<i>Merchant's secret key</i> <i>Limepay</i>	Key for authenticating a merchant for Limepay service by backend server. Place order works with correct pair of merchant specific public/secret keys.
Limepay Payment Options	<ul style="list-style-type: none"> <li><i>Full Payment &amp; Split Payment (multiple)</i></li> <li><i>Split Payment Only (splitOnly)</i></li> </ul>	Allowed Limepay payment options for site. Full Payment & Split Payment provide the customer with the choice to opt between the 2 for checkout. Whereas Split Payment Only allows pay later as a payment and no one time settlements allowed.
Limepay Multi Option Default	<ul style="list-style-type: none"> <li><i>Full Payment (full)</i></li> <li><i>Split Payment (split)</i></li> </ul>	This preference is used only when 'multiple' is selected for the above preference 'Limepay Payment Options'. Lets merchants decide which among the 2 modes would be defaulted as payment option for a site.
Limepay PDP Widget Mode	<ul style="list-style-type: none"> <li><i>Textual (textual)</i>      <i>Message</i></li> <li><i>Toggle (toggle)</i>      <i>Button</i></li> </ul>	Decides product page / quick view Limepay widget behavior. Toggle button displays breakdown of one time and pay later options and allows customer to choose between 2 for checkout. Toggle default state matches to above preference 'Limepay Multi Option Default'. Toggle widget is applicable for a site when both payments are allowed as per preference 'Limepay Payment Options'. However, a merchant may also configure to display textual widget despite both payments are allowed. The textual widget shows a

		breakdown of pay later payment. When 'splitOnly' is configured for 'Limepay Payment Options' irrespective of this preference widget behavior shall always fall back to textual mode as customer has no options to choose between.
Limepay Cart Widget Mode	<ul style="list-style-type: none"> <li>• <i>Textual (textual) Message</i></li> <li>• <i>Toggle (toggle) Button</i></li> </ul>	Decides cart page Limepay widget behavior. Toggle button displays breakdown of one time and pay later options and allows customer to choose between 2 for checkout. Toggle default state matches to above preference 'Limepay Multi Option Default'. Toggle widget is applicable for a site when both payments are allowed as per preference 'Limepay Payment Options'. However, a merchant may also configure to display textual widget despite both payments are allowed. The textual widget shows a breakdown of pay later payment. When 'splitOnly' is configured for 'Limepay Payment Options' irrespective of this preference widget behavior shall always fall back to textual mode as customer has no options to choose between.
Lime Pay Script API URL	<i>(See below)</i>	Limepay API url for rendering iFrame payment section at checkout and acquiring payment token for processing order based on storefront basket details. This URL works with merchant's public key preference mentioned above
Limepay Customer Service Email	<i>(Merchant email id's)</i>	Comma or semi colon separated email id's to which Limepay refund failure notification would be sent
3DS Enabled	<i>Yes/No depending on enable/disable 3DS Flow</i>	Control to turn off/on the Limepay 3DS functionality.
Limepay 3DS Minimum Amount	<i>Any value in number.</i>	<i>Minimum Total Amount added to enable the Limepay 3DS Verification.</i>  Only check if 3DS is enable and see if order total is greater or equal to this amount then 3DS verification is allowed.
Primary Color	<i>Any Hex Color Code.</i>	Color that loads on Limepay render widget
Apple Pay Domain Association	<i>Domain Association Code</i>	Used to Limepay Integration with Apple Pay

Environment	Limepay Script API URL
Non-production instances	<a href="https://checkout.limepay.com.au/v2/checkout-v2.1.0.min.js">https://checkout.limepay.com.au/v2/checkout-v2.1.0.min.js</a>
Production instances	<a href="https://checkout.limepay.com.au/v2/checkout-v2.1.0.min.js">https://checkout.limepay.com.au/v2/checkout-v2.1.0.min.js</a>

## 6.2. Configuring Payment Methods

Refer below table for configuring Limepay payment methods

Payment Method Setting	Payment Method : Limepay	Payment Method : Limepay_instalment	Description
Enabled	Yes	Yes	Though <i>Limepay_instalment</i> is a dummy payment method it requires to be enabled to allow threshold price range validation by storefront. However the same payment method has been skipped from rendering as a payment method at checkout.
Payment Processor	LIMEPAY	<i>Do not configure</i>	Only payment method used for checkout is 'Limepay' and is mapped to a valid processor. 'Limepay_instalment' is only for configurations
Min/Max Payment Ranges	<i>Do not configure</i>	<i>Min/Max payment ranges allowed for pay later payment (Use the min/max range provided by Limepay)</i>	Do not configure the min/max payment ranges for 'Limepay' as it is the main payment method used to place orders. Set it for supported AUD currency against 'Limepay_instalment' to limit payment range allowed for pay later.
Currencies	AUD	AUD	Please configure the applicable currencies based on your requirements.
Countries	<i>Based on requirements</i>	<i>Based on requirements</i>	Please configure the applicable countries based on your requirements.

## 7. Storefront Code Changes

Make the following code changes on your SiteGenesis cartridge to integrate with Limepay

### 7.1. Controller Changes

#### **COPlaceOrder.js**

**File : (*merchant\_controllers*)\cartridge\controllers\COPlaceOrder.js**

1. Add the below code changes to the function, handlePayments(order)

```
// Limepay, return response error to checkout page - STARTif (authorizationResult.error &&
authorizationResult.PlaceOrderError) {return {
error: true,
PlaceOrderError: authorizationResult.PlaceOrderError
};
}
// Limepay, return response error to checkout page - EN
```

Below the following lines

```
Transaction.wrap(handlePaymentTransaction);  
  
} else {  
  
var authorizationResult = PaymentProcessor.authorize(order,  
paymentInstrument);
```

Changes to reflect as shown below

```
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);

    // Limepay, return response error to checkout page - START
    if (authorizationResult.error && authorizationResult.PlaceOrderError) {
      return {
        error: true,
        PlaceOrderError: authorizationResult.PlaceOrderError
      };
    }
    // Limepay, return response error to checkout page - END

    if (authorizationResult.not_supported || authorizationResult.error) {
      return {
        error: true
      };
    }
  }
}
```

```
// Limepay, return response error to checkout page - START
if(handlePaymentsResult.error &&
handlePaymentsResult.PlaceOrderError) { return Transaction.wrap(function () {
OrderMgr.failOrder(order); return {
error: true,
PlaceOrderError: handlePaymentsResult.PlaceOrderError
};
});
}
// Limepay, return response error to checkout page - END
```

2. Add the below code changes to the function, start()

Below the following lines

```
if (!order) {  
    app.getController('Cart').Show();  
    return {};  
}  
var handlePaymentsResult = handlePayments(order);
```

Changes to reflect as shown below

```
if (!order) {  
    // TODO - need to pass BasketStatus to Cart-Show ?  
    app.getController('Cart').Show();  
    return {};  
}  
var handlePaymentsResult = handlePayments(order);  
  
// Limepay, return response error to checkout page - START  
if(handlePaymentsResult.error && handlePaymentsResult.PlaceOrderError) {  
    return Transaction.wrap(function () {  
        OrderMgr.failOrder(order);  
        return {  
            error: true,  
            PlaceOrderError: handlePaymentsResult.PlaceOrderError  
        };  
    });  
}  
// Limepay, return response error to checkout page - END  
  
if (handlePaymentsResult.error) {  
    return Transaction.wrap(function () {  
        OrderMgr.failOrder(order);  
        return {  
            error: true,  
            PlaceOrderError: handlePaymentsResult.PlaceOrderError  
        };  
    });  
}
```

3. Add 3DS payment condition to send payment action result on front end.

```
} else if (handlePaymentsResult.paymentResult) {  
    //LimePay 3DS Response  
    return {  
        paymentResult: handlePaymentsResult,  
        threeDSURL: URLUtils.url('Limepay-Limepay3DS', 'threeDSCallback', true, 'OrderNo', order.orderNo).toString()  
    };  
}
```

4. Add 3DS Response from payment handler method
- ```
//Add LimePay 3DS if 3DS redirect is enabled
var paymentResult = authorizationResult.paymentResult;

if (paymentResult && paymentResult.threeDSRedirect) {
    return {
        paymentResult: paymentResult.object,
        threeDSURL: URLUtils.url('Limepay-Limepay3DS', 'threeDScallback', true, 'OrderNo', order.orderNo).toString()
    };
}
```

## COSummary.js

File : (merchant\_controllers)\cartridge\controllers\COSummary.js

```
/**
 * Renders the order confirmation page after successful order
 * creation. If a nonregistered customer has checked out, the confirmation page
 * provides a "Create Account" form. This function handles the
 * account creation.
 */
function showConfirmation(order) {
    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,
        ContinueURL: URLUtils.https('Account-RegistrationForm') // needed by registration form after anonymous checkouts
    }).render('checkout/confirmation/confirmation');
}
```

## COBilling.js

File : (merchant\_controllers)\cartridge\controllers\COBilling.js

Add below code in function returnToForm

```

// Limepay Signin Customer With Customer ID
var customerToken = '';
var limepayEnabled = limepayUtilsHelpers.isLimepayPaymentEnabled();
if (limepayEnabled) {
    if (customer.authenticated) {
        var profile = app.getModel('Profile').get();
        var limepayCustomerId = profile.object.custom.limepayCustomerId;
        if (limepayCustomerId) {
            // Limepay customer login with customer id API call
            var response = limepaySignInCustomerWithCustomerId.signInLimepayCustomerWithCustomerId(limepayCustomerId);
            if (response.ok && response.object) {
                customerToken = response.object.customToken;
            }
        }
    }
}

if (params) {
    app.getView(require('~/cartridge/scripts/object').extend(params, {
        limepayCustomerToken: customerToken,
        Basket: cart.object,
        ContinueURL: URLUtils.https('COBilling-Billing')
    })).render('checkout/billing/billing');
} else {
    app.getView({
        limepayCustomerToken: customerToken,
        Basket: cart.object,
        ContinueURL: URLUtils.https('COBilling-Billing')
    }).render('checkout/billing/billing');
}

```

### Account.js

File : (merchant\_controllers)\cartridge\controllers\Account.js

Add below code in function registrationForm() after

profileValidation = Customer.createAccount(email, password, app.getForm('profile'));

```

// Limepay new customer details to Limepay
var limepayEnabled = limepayUtilsHelpers.isLimepayPaymentEnabled();
if (limepayEnabled) {
    var customer = CustomerMgr.getCustomerByLogin(email);
    if (customer) {
        // Limepay create upsert customer API call
        limepayUpsertCustomerService.upsertLimepayCustomer(customer);
    }
}

```

You, 13 hours ago • Customer Token Generation SG

### Login.js

File : (merchant\_controllers)\cartridge\controllers>Login.js

Add Below code in function handleLoginForm () before

```
loginForm.clear();
```

```
//LimePay Upsert Customer Id If not already
var limepayEnabled = limepayUtilsHelpers.isLimepayPaymentEnabled();
if (limepayEnabled) {
    var email = loginForm.getValue('username');
    var customer = CustomerMgr.getCustomerByLogin(email);
    var limepayCustomerId = customer.profile.custom.limepayCustomerId;
    if (!limepayCustomerId) {
        // Limepay create upsert customer API call
        limepayUpsertCustomerService.upsertLimepayCustomer(customer);
    }
}
```

7.2.

Models Changes

### CartModel.js

File : (merchant\_controllers)\cartridge\scripts\models

1. Add the below code changes to the function, calculate()

```
// Limepay changes - START updatePayLaterEligibility(this.object);  
// Limepay changes - END
```

Below the following lines

```
calculate: function () { dw.system.HookMgr.callHook('dw.ocapi.shop.basket.calculate',  
'calculate', this.object);
```

Changes to reflect as shown below

```
    */  
    calculate: function () {  
        dw.system.HookMgr.callHook('dw.ocapi.shop.basket.calculate', 'calculate', this.object);  
  
        // Limepay changes - START  
        updatePayLaterEligibility(this.object);  
        // Limepay changes - END  
    },  
  
    addProductToCart: function() {  
        var cart = this;
```

2. Add the following **NEW Function** definition towards the end of file before line `module.exports = CartModel;`

```
// Limepay changes - START

/**
 * Iterate basket to find any disqualifying item Limepay pay later option
 * and update customer session
 * @param {object} basket The basket containing the elements which are
 * looped to find eligibility
 */
function updatePayLaterEligibility (basket) {
  var limepayUtilsHelpers = require('*/cartridge/scripts/helpers/
    limepayUtilsHelpers');
  var disableLimepayPayLater = false;
  if (!basket.getGiftCertificateLineItems().isEmpty()) {
    // Gift certificates always disqualifies pay later option
    disableLimepayPayLater = true;
  }
  var productLineItems = basket.getAllProductLineItems();
  if (!disableLimepayPayLater && !empty(productLineItems)) {
    // If no gift certificates, iterate product line items
    for (var i = 0; i < productLineItems.length; i++) {
      var product = productLineItems[i].product;
      if (limepayUtilsHelpers.excludeProductPayLater(product)) {
        disableLimepayPayLater = true;
        break;
      }
    }
  }
  if (disableLimepayPayLater &&
    !session.custom.disableLimepayPayLater) {
    // Set session attribute value
    session.custom.disableLimepayPayLater = true;
  } else if (!disableLimepayPayLater &&
    'disableLimepayPayLater' in session.custom) {
    // Clear session attribute value
    delete session.custom.disableLimepayPayLater;
  }
}
// Limepay changes - END
```

### 7.3. Client Side JS Changes

#### **billing.js**

**File :** *(merchant\_core)\cartridge\js\pages\checkout\billing.js*

1. Add the below code changes to the global variable set at the top of the file

```
limepay = require('.././limepay');
```

Below the following lines

```
'use strict';
```

```
/* eslint-disable */
```

```
var ajax = require('.././ajax'), formPrepare = require('./formPrepare'),giftcard = require('.././gift
```

Code changes to reflect as shown below

```
'use strict';

/* eslint-disable */

var ajax = require('../..//ajax'),
    formPrepare = require('../formPrepare'),
    giftcard = require('../..//giftcard'),
    util = require('../..//util'),
    limepay = require('../..//limepay');

/**
 * @function
 * @description Fills the Credit Card form with the passed data-
 * @param {Object} data The Credit Card data (holder, type, mask)
 */
function setCCFields(data) {
  var $creditCard = $('[data-method="CREDIT_CARD"]');
  $creditCard.find('input[name$="creditCard_owner"]').val(data
```

2. Add the below code changes to the function, `updatePaymentMethod(paymentMethodID)`

```
// Limepay changes - START
var $submitBillingFormButton = $('button[name=dwfrm_billing_save]');
$submitBillingFormButton.removeClass('hide');
if (paymentMethodID && 'Limepay' === paymentMethodID) {
$submitBillingFormButton.addClass('hide');
$('body').trigger('limePayment:handlePaymentTabEvents');
} else {
$('.limepay-submit-payment').remove();
}
// Limepay changes - END
```

Below the following lines

```
if ($selectedPaymentMethod.length === 0) {  
  
$selectedPaymentMethod = $('[data-method="Custom"]');  
}  
$selectedPaymentMethod.addClass('payment-method-expanded');  
// ensure checkbox of payment method is checked  
$('input[name$="_selectedPaymentMethodID"]').removeAttr('checked');  
$('input[value=' + paymentMethodID + ']').prop('checked', 'checked');
```

Code changes to reflect as shown below

```
|   >selectedPaymentMethod = $( [data-method= custom ] );  
| }  
| $selectedPaymentMethod.addClass('payment-method-expanded');  
  
| // ensure checkbox of payment method is checked  
| $('input[name$="_selectedPaymentMethodID"]').removeAttr('checked');  
| $('input[value=' + paymentMethodID + ']').prop('checked', 'checked');  
  
| // Limepay changes - START  
| var $submitBillingFormButton = $('button[name=dwfrm_billing_save]');  
| $submitBillingFormButton.removeClass('hide');  
| if (paymentMethodID && 'Limepay' === paymentMethodID) {  
|   $submitBillingFormButton.addClass('hide');  
|   $('body').trigger('limePayment:handlePaymentTabEvents');  
| } else {  
|   $('.limepay-submit-payment').remove();  
| }  
| // Limepay changes - END  
  
| prepare.validateForm();
```

3. Add the below code changes to `init()` function

```
// Limepay changes - STARTlimepay.init();  
// Limepay changes - END
```

Below the following lines

```
var $selectPaymentMethod = $('<div>.payment-method-options');  
  
    var selectedPaymentMethod = $selectPaymentMethod.find(':checked')  
        .val();  
        formPrepare.init({  
formSelector: 'form[id$="billing"]', continueSelector: '[name$="billing_save"]'  
        });
```

Changes to reflect as shown below

```
var $selectPaymentMethod = $('#payment-method-options');
var selectedPaymentMethod = $selectPaymentMethod.find(':checked').val();

formPrepare.init({
  formSelector: 'form[id$="billing"]',
  continueSelector: '[name$="billing_save"]'
});

// Limepay changes - START
limepay.init();
// Limepay changes - END

// default payment method to 'CREDIT_CARD'
updatePaymentMethod((selectedPaymentMethod) ? selectedPaymentMethod : 'CREDIT_CARD');
$selectPaymentMethod.on('click', 'input[type="radio"]', function () {
  updatePaymentMethod($(this).val());
});
```

4. Add the below code changes to the end of event listener  
`$addGiftCert.on('click', function (e) {`

```
// Limepay changes - START
// On gift certificate redemption, update Limepay form
$('body').trigger('limePayment:updateLimePayOrderAmount');
// Limepay changes - END
```

Below the following lines

```
if (fail) {  
  $error.html(msg);return;  
} else {  
  window.location.assign(Urls.billing);  
}
```

Code changes to reflect as shown below

```
$addGiftCert.on('click', function (e) {  
    e.preventDefault();  
    var code = $giftCertCode.val(),  
        $error = $checkoutForm.find('.giftcert-error');  
    if (code.length === 0) {  
        $error.html(Resources.GIFT_CERT_MISSING);  
        return;  
    }  
  
    var url = util.appendParamsToUrl(Urls.redeemGiftCert, {giftCertCode: code, format: 'ajax'});  
    $.getJSON(url, function (data) {  
        var fail = false;  
        var msg = '';  
        if (!data) {  
            msg = Resources.BAD_RESPONSE;  
            fail = true;  
        } else if (!data.success) {  
            msg = data.message.split('<').join('&lt;');  
            fail = true;  
        }  
        if (fail) {  
            $error.html(msg);  
            return;  
        } else {  
            window.location.assign(Urls.billing);  
        }  
  
        // Limepay changes - START  
        // On gift certificate redemption, update Limepay form  
        $('body').trigger('limePayment:updateLimePayOrderAmount');  
        // Limepay changes - END  
    });  
  
    on('click', function (e) {  
        // default //  
    });  
});
```

5. Add the below code changes to the end of event listener

```
$addCoupon.on('click', function (e) {
```

```
// Limepay changes - START
// On successful coupon addition, update Limepay form
$('body').trigger('limePayment:updateLimePayOrderAmount');
// Limepay changes - END
```

Below the following lines

```
if (data.success && data.baskettotal === 0) {
window.location.assign(Urls.billing);
}
```

Code changes to reflect as shown below

```
$addCoupon.on('click', function (e) {
    e.preventDefault();
    var $error = $checkoutForm.find('.coupon-error'),
        code = $couponCode.val();
    if (code.length === 0) {
        $error.html(Resources.COUPON_CODE_MISSING);
        return;
    }

    var url = util.appendParamsToUrl(Urls.addCoupon, {couponCode: code, format: 'ajax'});
    $.getJSON(url, function (data) {
        var fail = false;
        var msg = '';
        if (!data) {
            msg = Resources.BAD_RESPONSE;
            fail = true;
        } else if (!data.success) {
            msg = data.message.split('<').join('&lt;');
            fail = true;
        }
        if (fail) {
            $error.html(msg);
            return;
        }

        //basket check for displaying the payment section, if the adjusted total of the basket is 0 a
        //this will force a page refresh to display the coupon message based on a parameter message
        if (data.success && data.baskettotal === 0) {
            window.location.assign(Urls.billing);
        }

        // Limepay changes - START
        // On successful coupon addition, update Limepay form
        $('body').trigger('limePayment:updateLimePayOrderAmount');
        // Limepay changes - END
    });
});

// Trigger events on enter
$.on('keydown', function (e) {
    if (e.which === 13) {
```

6. Add below changes for 3DS initialization

```
// Limepay changes - START
limepay.init();
// Limepay changes - END

if ($isPaymentMethodLimePay && $isLimepay3DSEnabled) {
    // Flag to identify if order needs to be submitted or not;
    window.submitOrder = false;

    $('form.submit-order').submit(function (e) {
        if (!window.submitOrder) {
            e.preventDefault();
            var $submitOrder = $(this);
            var $orderSubmirURL = $submitOrder.attr('action');
            $orderSubmirURL = util.appendParamsToUrl($orderSubmirURL, {format: 'ajax'});
            $.ajax({
                url: $orderSubmirURL,
                type: 'POST',
                success: function (data) {
                    if (data.error || data.orderNo) {
                        window.submitOrder = true;
                        $orderSubmirURL = util.removeParamFromURL($orderSubmirURL, 'format');
                        $orderSubmirURL = util.appendParamsToUrl($orderSubmirURL, data);
                        $orderForm.attr('action', $orderSubmirURL);
                        $orderForm.find('button[name="submit"]').click();
                        // error handling
                    } else if (data.threeDSURL) {
                        var paymentResult = data.paymentResult || {};

                        if (paymentResult.paymentActionRequired) {
                            limepay.initLimePay3DS(data);
                        }
                    }
                },
                error: function (err) {
                    window.location.href = URLs.cartShow;
                }
            });
        }
    });
}
```

## productSet.js

File : (merchant\_core)\cartridge\js\pages\product\productSet.js

1. Add the below code changes to the end of event listener

```
$productSetList.on('click', '.product-set-item .swatchanchor',
                    function (e) {

// Limepay changes - START

// Sync Limepay toggle widget state for remaining product set items
if ($container.find('.limepay-option').length > 0) {
    var $currentToggle = $container.find('.limepay-
        switch input[name=limepay-selection]');
    var currentToggleSelection = $currentToggle.prop('checked') ? 'split
        ' : 'full';
    $(' .limepay-switch input[name=limepay-
        selection]').not($currentToggle).prop('checked',
        currentToggleSelection == 'split' ? true : false);
    if (currentToggleSelection == 'full') {
        $(' .limepay-pay-onetime').not($container.find(' .limepay-pay-
            onetime')).addClass('active');
        $(' .limepay-payin-four').not($container.find(' .limepay-payin-
            four')).removeClass('active');
    } else {
        $(' .limepay-pay-onetime').not($container.find(' .limepay-pay-
            onetime')).removeClass('active');
        $(' .limepay-payin-four').not($container.find(' .limepay-payin-
            four')).addClass('active');
    }
}
// Limepay changes - END
```

Below the following lines

```
callback: function () {  
  updateAddToCartButtons();tooltip.init();
```

Code changes to reflect as shown below

```
updateAddToCartButtons();
}
// click on swatch for product set
$productSetList.on('click', '.product-set-item .swatchanchor', function (e) {
    e.preventDefault();
    if ($(this).parents('li').hasClass('unselectable')) { return; }
    var url = Urls.getSetItem + this.search;
    var $container = $(this).closest('.product-set-item');
    var qty = $container.find('form input[name="Quantity"]').first().val();

    ajax.load({
        url: util.appendParamToURL(url, 'Quantity', isNaN(qty) ? '1' : qty),
        target: $container,
        callback: function () {
            updateAddToCartButtons();
            tooltip.init();

            // Limepay changes - START
            // Sync Limepay toggle widget state for remaining product set items
            if ($container.find('.limepay-option').length > 0) {
                var $currentToggle = $container.find('.limepay-switch input[name=limepay-selection]');
                var currentToggleSelection = $currentToggle.prop('checked') ? 'split' : 'full';
                $('limepay-switch input[name=limepay-selection]').not($currentToggle).prop('checked', currentToggleSelection == 'split' ? true : false);
                if (currentToggleSelection == 'full') {
                    $('limepay-pay-onetime').not($container.find('.limepay-pay-onetime')).addClass('active');
                    $('limepay-payin-four').not($container.find('.limepay-payin-four')).removeClass('active');
                } else {
                    $('limepay-pay-onetime').not($container.find('.limepay-pay-onetime')).removeClass('active');
                    $('limepay-payin-four').not($container.find('.limepay-payin-four')).addClass('active');
                }
            }
            // Limepay changes - END
        }
    });
}
```

## **app.js**

**File : (merchant\_core) \ cartridge\js\app.js**

1. Add the new event listener code changes to the end of function, initializeEvents()

```
// Limepay changes - START
// Limepay toggle widget event
$('body').on('change', '.limepay-widget input#limepay-selection', function () {
var selectedToggle = $(this);
$('.limepay-pay-onetime').toggleClass('active');
$('.limepay-payin-four').toggleClass('active');
// Synchronize remaining page limepay toggles, based oncurrent one
// For product sets
$('.limepay-widget input#limepay-selection').not(selectedToggle)
.prop("checked", selectedToggle.prop("checked"));var userToggleSelection = $('input[name=limepay-
selection]').is(':checked') ? 'split' : 'full';

var saveSelectionURL = $('input[name=limepay-selection]').data('selectionurl');
saveSelectionURL = saveSelectionURL + '?limepayToggle=' +userToggleSelection;
```

```

$.ajax({
url: saveSelectionURL, success: function(data) {
}
});
});
// Limepay changes - END

```

Below the following lines

```

$('.user-account').on('click', function (e) {e.preventDefault();
$(this).parent('.user-info').toggleClass('active');
});

```

Code changes to reflect as shown below

```

>{ .user-account }.on( click , function (e) {
    e.preventDefault();
    $(this).parent('.user-info').toggleClass('active');
});

// Limepay changes - START
// Limepay toggle widget event
$('body').on('change', '.limepay-widget input#limepay-selection', function () {
    var selectedToggle = $(this);
    $('.limepay-pay-onetime').toggleClass('active');
    $('.limepay-payin-four').toggleClass('active');

    // Synchronize remaining page limepay toggles, based on current one
    // For product sets
    $('.limepay-widget input#limepay-selection').not(selectedToggle).prop("checked", selectedToggle.prop("checked"));

    var userToggleSelection = $('input[name=limepay-selection]').is(':checked') ? 'split' : 'full';
    var saveSelectionURL = $('input[name=limepay-selection]').data('selectionurl');
    saveSelectionURL = saveSelectionURL + '?limepayToggle=' + userToggleSelection;

    $.ajax({
        url: saveSelectionURL,
        success: function(data) {
        }
    });
});
// Limepay changes - END

```

## 7.4. Custom Files

Limepay.js

int\_limepay\_sg\cartridge\controllers\Limepay.js

**limepay.js**

This new file has to be added to the merchants core cartridge

Copy the file from

**limepay\_sg\_changes\cartridge\js\limepay.js**

and place it under

**(merchant\_core)\cartridge\js\**

## 7.5. ISML Changes

**billing.isml**

**File : (merchant\_core)\cartridge\templates\default\checkout\billing\billing.isml**

```
<iscomment>Limepay changes - START</iscomment>  
<isinclude template="limepay/limepaycheckoutinclude" />  
<iscomment>Limepay changes - END</iscomment>
```

1. Add the below code changes towards the end of file

Code changes to reflect as shown below

```
<div class="form-row form-row-button" >
  <button class="button-fancy-large" type="submit"
</div>
<input type="hidden" name="{dw.web.CSRFProtection.getToken

</form>
<iscript>
  importScript("util/ViewHelpers.ds");
  var addressForm = pdict.CurrentForms.billing.billingAddress.ac
  var countries = ViewHelpers.getCountriesAndRegions(addressForm
  var json = JSON.stringify(countries);
</iscript>
<script>window.Countries = <isprint value="{json}" encoding="off'
|
<iscomment>Limepay changes - START</iscomment>
<isinclude template="limepay/limepaycheckoutinclude" />
<iscomment>Limepay changes - END</iscomment>

'isdecorate>
```

**paymentmethods.isml**

**File : (*merchant\_core*)\cartridge\templates\default\checkout\billing\paymentmethods.isml**

1. Add the below code changes inside the first <isif condition

```
<iscomment>Limepay changes - START</iscomment>  
<isscript>  
var limepayUtilsHelpers = require('*/cartridge/scripts  
/helpers/limepayUtilsHelpers');  
var isLimepayCheckoutAllowed = limepayUtilsHelpers  
.limepayCheckoutAllowed();  
</isscript>  
<iscomment>Limepay changes - END</iscomment>
```

Below the following lines

```
<iscontent type="text/html" charset="UTF-8" compact="true"/>  
<iscomment> TEMPLATENAME: paymentmethods.isml </iscomment>  
<isinclude template="util/modules"/>  
<isif condition="${pdict.OrderTotal > 0}">
```

Code changes to reflect as shown below

```
:iscontent type="text/html" charset="UTF-8" compact="true"/>
:iscomment> TEMPLATENAME: paymentmethods.isml </iscomment>
:isinclude template="util/modules"/>
:isif condition="{pdict.OrderTotal > 0}">

  <iscomment>Limepay changes - START</iscomment>
  <isscript>
    var limepayUtilsHelpers = require('*/cartridge/scripts/helpers/limepayUtilsHelpers');
    var isLimepayCheckoutAllowed = limepayUtilsHelpers.limepayCheckoutAllowed();
  </isscript>
  <iscomment>Limepay changes - END</iscomment>

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

2. Add and update the existing code inside the div container as shown below

```
<div class="payment-method-options form-indent">

<iscomment>Limepay changes - START</iscomment>

<isif condition="{paymentMethodType.value.
    equals('Limepay_instalment')}"><iscontinue/></isif>
<isif condition="{paymentMethodType.value.equals('Limepay') &&
    !isLimepayCheckoutAllowed}">
    <iscontinue/>
<iselseif condition="{paymentMethodType.value.equals('Limepay')
    && isLimepayCheckoutAllowed}">
    <isinclude template="limepay/paymentmethodinput" />
<iselse/>
    <div class="form-row label-inline">
        <isset name="radioID" value="{paymentMethodType.value}"
            scope="page"/>
        <div class="field-wrapper">
            <input id="is-{$radioID}" type="radio" class="input-
radio" name="{pdict.CurrentForms.billing.paymentMethods.selectedPaymentMethod
ID.htmlName}" value="{paymentMethodType.htmlValue}" <isif condition="{paymen
tMethodType.value == pdict.CurrentForms.billing.paymentMethods.selectedPayment
MethodID.htmlValue}">checked="checked"</isif> />
            </div>
            <label for="is-{$radioID}"><isprint value="{Resource.
                msg(paymentMethodType.label, 'forms', null)}"/></label>
        </div>
    </isif>
<iscomment>Limepay changes - END</iscomment>
```

```
<div class="payment-method-options form-indent">
<isloop items="{pdict.CurrentForms.billing.paymentMethods.selectedPaymentMethodID.options}" var="payme
<iscomment>Ignore GIFT_CERTIFICATE method, GCs are handledseparately before other payment methods.</isc
<isif condition="{paymentMethodType.value.equals(dw.order. PaymentInstrument.METHOD_GIFT_CERTIFICATE)}"
</isif>
```

Below the following lines

Code changes to reflect as shown below

```
<div class="payment-method-options form-indent">
  <isloop items="{pdict.CurrentForms.billing.paymentMethods.selectedPaymentMethodID.options}" var="paymentMethodType">
    <iscomment>Ignore GIFT_CERTIFICATE method, GCs are handled separately before other payment methods.</iscomment>
    <isif condition="{paymentMethodType.value.equals(dw.order.PaymentInstrument.METHOD_GIFT_CERTIFICATE)}"><iscontinue/></isif>

    <iscomment>Limepay changes - START</iscomment>
    <isif condition="{paymentMethodType.value.equals('Limepay_instalment')}"><iscontinue/></isif>
    <isif condition="{paymentMethodType.value.equals('Limepay') && !isLimepayCheckoutAllowed}">
      <iscontinue/>
    <iselseif condition="{paymentMethodType.value.equals('Limepay') && isLimepayCheckoutAllowed}">
      <isinclude template="limepay/paymentmethodinput" />
    <iselse/>
      <div class="form-row label-inline">
        <isset name="radioID" value="{paymentMethodType.value}" scope="page"/>
        <div class="field-wrapper">
          <input id="is-{$radioID}" type="radio" class="input-radio" name="{pdict.CurrentForms.billing.paymentMethods.selectedPaymentMethodID.value}" />
        </div>
        <label for="is-{$radioID}"><isprint value="{Resource.msg(paymentMethodType.label,'forms',null)}"/></label>
      </div>
    </isif>
    <iscomment>Limepay changes - END</iscomment>
  </isloop>
</div>
```

3. Add the below code changes to the custom processor section

```
<iscomment>Limepay changes - START</iscomment>  
<isif condition="{isLimepayCheckoutAllowed}">  
<isinclude template="limepay/paymentmethod" sf-toolkit="on" />  
</isif>  
<iscomment>Limepay changes - END</iscomment>
```

Below the following lines

```
<div class="bml-terms-and-conditions form-caption">
```

```
<iscontentasset aid="bml-tc"/>
```

```
</div>
```

```
<div class="form-row form-caption">
```

```
<inputfield formfield="${pdict.CurrentForms.billing.paymentMethods.bml.termsandconditions}" type="checkbox"/>
```

```
</div>
```

```
</div>
```

Code changes to reflect as shown below

```
<inputfield formfield="{pdict.CurrentForms.billing.paymentMethods.bml.termsandc

<div class="bml-terms-and-conditions form-caption">
| <iscontentasset aid="bml-tc"/>
</div>

<div class="form-row form-caption">
| <inputfield formfield="{pdict.CurrentForms.billing.paymentMethods.bml.termsandc
</div>

</div>

<iscomment>Limepay changes - START</iscomment>
<isif condition="{isLimepayCheckoutAllowed}">
| <isinclude template="limepay/paymentmethod" sf-toolkit="on" />
</isif>
<iscomment>Limepay changes - END</iscomment>

<iscomment>
Custom processor
-----
</comment>
```

### cart.isml

File : *(merchant\_core)\cartridge\templates\default\checkout\cart\cart.isml*

1. Add the below code changes to the top include section

```
<iscomment>Limepay modules</iscomment>  
<isinclude template="limepay/util/modules" />
```

Below the following lines

```
<iscontent type="text/html" charset="UTF-8" compact="true"/>
```

```
<isdecorate template="checkout/cart/pt_cart">
```

```
<isinclude template="util/modules" />
```

```
<isinclude template="util/reporting/ReportBasket.isml" />
```

Code changes to reflect as shown below

```
:iscontent type="text/html" charset="UTF-8" compact="true"/>
:isdecorate template="checkout/cart/pt_cart">
  <isinclude template="util/modules" />
  <isinclude template="util/reporting/ReportBasket.isml" />

  <iscomment>Limepay modules</iscomment>
  <isinclude template="limepay/util/modules" />

  <isset name="enableCheckout" value="${pdict.EnableCheckout}" scope="page" />
  <isif condition="${dw.system.Site.getCurrent().getCustomPreferenceValue('enab
```

2. Add the below code changes above the last 'cart-actions' div container

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

```
  <iscomment>Limepay payment options widget - START</iscomment>
```

```
    <isscript>
```

```
      var limepayUtilsHelpers = require('*/cartridge/scripts/helpers/limepayUtilsHelpers');
```

```
      // Preferred cart widget display mode
```

```
      var cartWidgetDisplayMode = limepayUtilsHelpers.getLimepayWidgetDisplayMode('cart');
```

```
      var excludePayLaterForCustomer = limepayUtilsHelpers.isPayLaterExcludedForCustomer();
```

```
      var bnpCartThresholdExceeds = limepayUtilsHelpers.bnplContextThresholdExceeds('cart');
```

```
      // Rendering widget display mode based on current basket products & threshold
```

```
      var userWidgetDisplayMode = excludePayLaterForCustomer || bnpCartThresholdExceeds ? 'hidden' : cartWidgetDisplayMode;
```

```
    </isscript>
```

```
    <islimepaypaymentwidget context="${'cart'}" amount="${orderTotalValue.decimalValue}" currency="${session.currency.currencyCode}" widgetmode="${userWidgetDisplayMode}" />
```

```
  <iscomment>Limepay payment options widget - END</iscomment>
```

Below the following lines

```
<elseif condition="${pdict.CouponStatus != null &&pdict.CouponStatus.error}">
<div class="error">
${Resource.msgf('cart.' + pdict.CouponStatus.code, 'checkout', null, pdict.CurrentForms.cart.couponCode
</div>
</isif>
</div>
</div>
<input type="hidden" name="${dw.web.CSRFProtection.getTokenName()}" value="${dw.web.CSRFProtection.gen
</fieldset>
</form>
```

Code changes to reflect as shown below

```
        </isif>
      </div>
    </div>
    <input type="hidden" name="{dw.web.CSRFProtection.getTokenName()}" value="{dw.web.CSRFProtection.generateToken()}" />
  </fieldset>
</form>

<iscomment>Limepay payment options widget - START</iscomment>
<isscript>
  var limepayUtilsHelpers = require('*/cartridge/scripts/helpers/limepayUtilsHelpers');
  // Preferred cart widget display mode
  var cartWidgetDisplayMode = limepayUtilsHelpers.getLimepayWidgetDisplayMode('cart');
  var excludePayLaterForCustomer = limepayUtilsHelpers.isPayLaterExcludedForCustomer();
  var bnpCartThresholdExceeds = limepayUtilsHelpers.bnplContextThresholdExceeds('cart');
  // Rendering widget display mode based on current basket products & threshold
  var userWidgetDisplayMode = excludePayLaterForCustomer || bnpCartThresholdExceeds ? 'hidden' : cartWidgetDisplayMode;
</isscript>
<islimepaypaymentwidget context="{ 'cart' }" amount="{orderTotalValue.decimalValue}" currency="{session.currency.currencyCode}" widgetmode="{u
<iscomment>Limepay payment options widget - END</iscomment>

<div class="cart-actions">

  <iscomment>continue shop url is a non-secure but checkout needs a secure and that is why separate forms!</iscomment>
```

## htmlhead.isml

File : (merchant\_core) \cartridge\templates\default\components\header\htmlhead.isml

1. Add the below code changes to the end of file

```
<!-- Limepay UI -->  
<link rel="stylesheet" href="${URLUtils.staticURL('/css/limepay.css')}" />
```

Code changes to reflect as shown below

```
<isif condition="{GoogleVerificationTag" in dw.system.Site.current.preferences
  <meta name="google-site-verification" content="<isprint value="{dw.system
</isif>

<iscomment>Gather device-aware scripts</iscomment>
<isinclude url="{URLUtils.url('Home-SetLayout')}" />

<!-- Limepay UI -->
<link rel="stylesheet" href="{URLUtils.staticURL('/css/limepay.css')}" />
```

### **productsetproduct.isml**

File : (merchant\_core)\cartridge\templates\default\product\components\productsetproduct.isml

1. Add the below code changes beneath the following line

```
<isinclude template="product/components/pricing"/>
```

```
<iscomment> Limepay Product Widget - START </iscomment>
<isif condition="{pdict.isSet}">
<isset name="limepayPartOfProductSet" value="{true}"scope="pdict" />
<isinclude template="limepay/limepayproductwidget" />
</isif>
<iscomment> Limepay Product Widget - END </iscomment>
```

Code changes to reflect as shown below

```
<iscomment>
  product pricing
  -----
</iscomment>

<isinclude template="product/components/pricing"/>

<iscomment> Limepay Product Widget - START </iscomment>
<isif condition="{pdict.isSet}">
  <isset name="limepayPartOfProductSet" value="{true}" scope="pdict" />
  <isinclude template="limepay/limepayproductwidget" />
</isif>
<iscomment> Limepay Product Widget - END </iscomment>

<isif condition="{pdict.isSet}">
  <isinclude template="product/components/promotions"/>
</isif>
```

## productcontent.isml

File : (merchant\_core) \cartridge\templates\default\product\productcontent.isml

1. Add the below code changes beneath the following line  
`<isinclude template="product/components/pricing"/>`

```
<iscomment> Limepay Product Widget - START </iscomment>
```

```
<isinclude template="limepay/limepayproductwidget" />
```

```
<iscomment> Limepay Product Widget - END </iscomment>
```

Code changes to reflect as shown below

```
<iscomment>
  product pricing
  =====
</iscomment>

<isset name="showTieredPrice" value="{true}" scope="page"/>
<isinclude template="product/components/pricing"/>

<iscomment> Limepay Product Widget - START </iscomment>
<isinclude template="limepay/limepayproductwidget" />
<iscomment> Limepay Product Widget - END </iscomment>

<isset name="pam" value="{pdict.Product.getAttributeModel()}" scope="page"/>
<isset name="group" value="{pam.getAttributeGroup('mainAttributes')}" scope="page"/>
```

### **producttopcontentPS.isml**

**File : (merchant\_core)\cartridge\templates\default\product\producttopcontentPS.isml**

1. Add the below code changes beneath the following line  
`<isinclude template="product/components/pricing"/>`

```
<iscomment> Limepay Product Widget - START </iscomment>  
<isinclude template="limepay/limepayproductwidget" />  
<iscomment> Limepay Product Widget - END </iscomment>
```

Code changes to reflect as shown below

```
<iscomment>
  product set price
  =====
</iscomment>

<label>${Resource.msg('product.setpricelabel','product',null)}</label>
<isinclude template="product/components/pricing"/>
<iscomment> Limepay Product Widget - START </iscomment>
<isinclude template="limepay/limepayproductwidget" />
<iscomment> Limepay Product Widget - END </iscomment>
<isif condition="${pdict.isSet}">
  <button id="add-all-to-cart" type="submit" value="${Resource.msg(
    ${Resource.msg('global.addalltocart','locale',null)}
  )}>
</button>
</isif/>
```

## 8. User Guide

### 8.1. Custom CSS

Limepay allows merchant to redesign the checkout iFrame content through custom css file hosted by cartridge and rendered by the Limepay checkout API

File Location : *int\_limepay\_sg\cartridge\static\default\css\limepay.css*

CSS added to the custom file will override the Limepay default CSS.

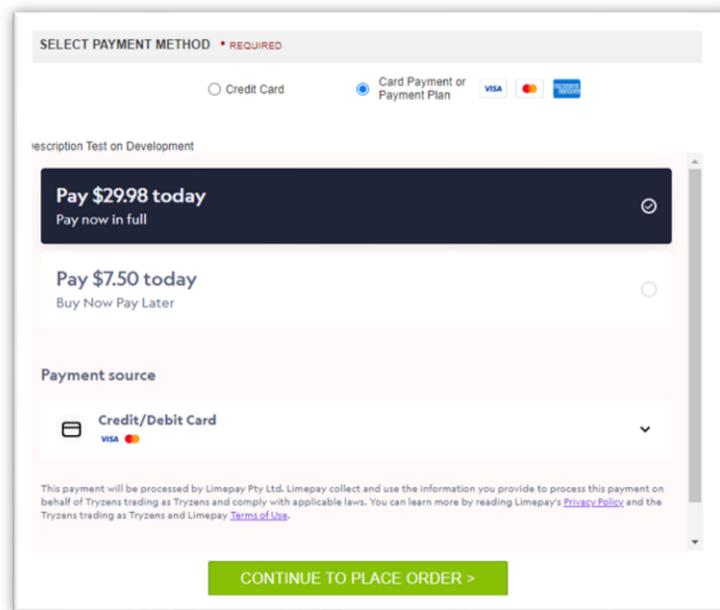
File : *(merchant\_core)\cartridge\js\limepay.js*

```
    }
  });
  LimepayCheckout.render({
    elementId: 'limepay-cont',
    currency: order.orderInfo.currency,
    amount: order.orderInfo.amount,
    paymentType: context.preselectedPaymentType == 'full' ? 'paycard' : 'payplan',
    showPayNow: false,
    showPayPlanSubmit: false,
    checkoutCSSOverride: context.checkoutCSSOverride
  });
  LimepayCheckout.errorHandler(function (err) {
    // Handle errors
```

For more detail [see](#)

## 8.2. Limepay Widget

Limepay provides widgets on product pages / quick views and cart page allowing the customer to know a breakdown of price for each payment modes i.e, one time and pay later. There are 2 types of widget the first being a toggle widget allowing customer interaction whereas the second a plain textual one and requires no customer interaction. Widget enhances customer experience on the store letting him/her know what mode of payment will be allowed for checkout. The toggle widget is identical to the toggle button observed within the checkout iFrame as shown below, with the widget and checkout toggle states being synced throughout the site on customer choice basis



**NB:** Widgets are visible on product/cart pages only when the followings conditions are satisfied

- '*Limepay Enabled*' preference is set as true
- Payment method '*Limepay*' is active for site
- User session currency matches the configured currency here AUD for the '*Limepay*' payment method

### 8.2.1. Toggle Widget

Limepay`s toggle widget has two states, with one time payment on the left and pay later payment on the right. Each side also shows the price breakdown when used for checkout by the customer. One time price split up on the left matches with the full price as seen on product or cart level. Whereas pay later price split up on the right matches to an instalment amount of one fourth the full product or cart total that the customer would pay equally in 4 different slots over a duration of 2 weeks each. Toggle widgets are allowed for a site only when the preference '*Limepay Payment Options*' is mapped

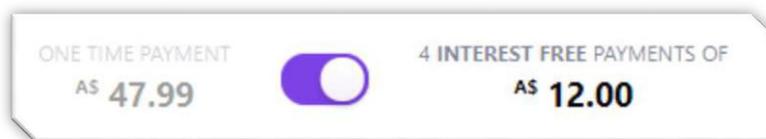
to *'multiple'*. Followingly a merchant can configure to have the toggle button be preselected for all customers on a particular side through preference *'Limepay Multi Option Default'*. The customer can

change the state by clicking on widget and have a particular mode of payment preselected for his/her checkout. Once a customer views the toggle widget, and changes the state either from product, cart or checkout pages it is then recorded against user session and the same state would be retained throughout the storefront session for product/cart/checkout pages.

A. Toggle widget selected for one time payment

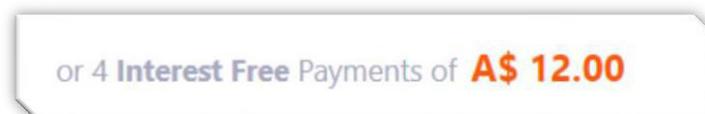


B. Toggle widget selected for pay later payment



### 8.2.2. Textual Widget

Limepay's textual widget calls out only the pay later price breakdown with a plain text message. This widget requires no customer interaction.



### 8.2.3. Product Widget

Limepay toggle and textual widgets are supported in product pages and quick views for all type of products, and positioned beneath the product price section. Merchant can configure to show either toggle or textual widget through preference 'Limepay PDP Widget Mode'. However for preference 'Limepay Payment Options' selected as 'splitOnly' product widget would always fallback to textual mode irrespective of the selected preference value for 'Limepay PDP Widget Mode'. Certain products and or categories can be configured to disallow pay later option through custom attributes 'disableLimepayPayLater'. Products viewed would not show any form of the widget if the above attribute is set as 'true'. The attribute look up for product viewed follows the hierarchy as product >

product's primary category. No other factors have control over the display of product widgets unlike in the cart page related to customer scenarios like basket including pay later disabled products and or total exceeding pay later threshold range. This will be discussed below in the cart widget section.

Let us see how product widgets are shown for different scenarios as follows

1. 'Limepay Payment Options' set as 'multiple' and 'Limepay PDP Widget Mode' set as 'toggle', product attribute 'disableLimepayPayLater' is set as 'no'. Product widget would be as shown below



2. 'Limepay Payment Options' set as 'multiple' and 'Limepay PDP Widget Mode' set as 'textual', product attribute 'disableLimepayPayLater' is set as 'no'. Product widget would be as shown below



Classic Bermuda Short  
Item No. 23564588  
★★★★☆  
A\$ 49,00  
or 4 Interest Free Payments of **A\$ 12.25**

SELECT COLOR  
 SWISS NAVY & WHITE

SELECT SIZE  
4 6 8 10 12 14 16  
Size Chart

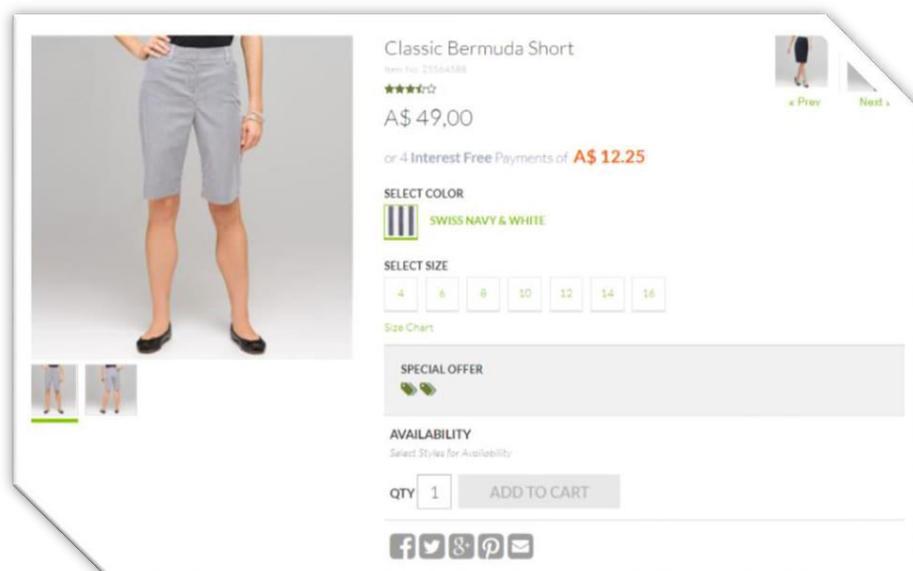
SPECIAL OFFER  


AVAILABILITY  
Select Styles for Availability

QTY



3. 'Limepay Payment Options' set as 'splitOnly' and 'Limepay PDP Widget Mode' set as 'toggle' or 'textual', product attribute 'disableLimepayPayLater' is set as 'no'. Product widget would be as shown below



Classic Bermuda Short  
Item No. 23564588  
★★★★☆  
A\$ 49,00  
or 4 Interest Free Payments of **A\$ 12.25**

SELECT COLOR  
 SWISS NAVY & WHITE

SELECT SIZE  
4 6 8 10 12 14 16  
Size Chart

SPECIAL OFFER  


AVAILABILITY  
Select Styles for Availability

QTY



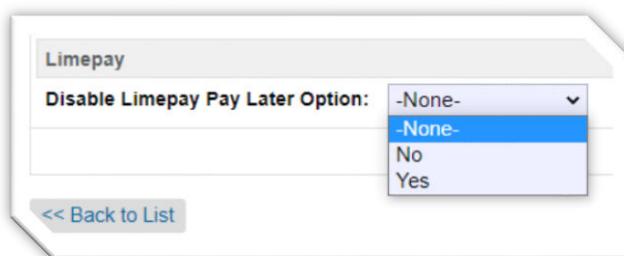
- When product attribute *'disableLimepayPayLater'* is set as *'yes'*, Limepay preferences matches either of the above configurations 1,2 or 3 OR when *'Limepay Enabled'* is set as *'no'*, product widget would not be shown



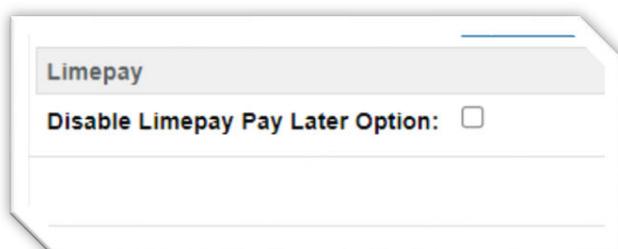
**NB** : Merchant can set the *'disableLimepayPayLater'* at category level also, products viewed whose primary category attribute have *'disableLimepayPayLater'* value as checked (true) do hide the product widget. Product's *'disableLimepayPayLater'* attribute is looked upon for a true/checked value in the following order

- Product
- Product's primary category

Product level disable pay later attribute :



Category Level disable pay later attribute :



**NB:** Product page widgets are placed within the cached product container, while the textual widgets are cached toggle widgets are not, in order to render customer dynamic toggle state. So changing the widget behavior through preference 'Limepay PDP Widget Mode' requires cache clear for effective display of widget behaviour

When widget is configured for Toggle mode, they are rendered through remote include. This is to synchronize the toggle widget with the customer dynamic selection throughout the storefront session. While textual widget are always cached requires no dynamic content other than product price.

#### 8.2.4. Cart Widget

Limepay cart widgets supports toggle and textual modes and is positioned below cart total in summary section. Merchant can configure to show either toggle or textual widget through preference 'Limepay Cart Widget Mode'. However, for preference 'Limepay Payment Options' selected as 'splitOnly' cart widget mode would always fallback to textual mode irrespective of the selected preference value for 'Limepay Cart Widget Mode' This is same as the product widget behavior. Configured mode of cart widget would not be visible if basket contains at least one pay later disabled product described in above section AND/OR cart total exceeds the pay later threshold range configured for the dummy payment method 'Limepay\_instalment'.

1. 'Limepay Payment Options' set as 'multiple' and 'Limepay Cart Widget Mode' set as 'toggle'. Basket do not contain pay later disabled product. Cart widget would be as shown below

The screenshot displays a shopping cart widget with the following details:

PRODUCT	DELIVERY OPTIONS	QTY	PRICE	TO.
 <b>Classic Bermuda Short</b> Item No.: 701643382095 Color: swiss navy & white Size 8 <a href="#">Edit Details</a>	Home Delivery	1	In Stock <a href="#">Remove</a> <a href="#">Add to Wishlist</a>	AS 49,00

Below the product table, there is a coupon code field with 'Apply' and 'Update Cart' buttons.

Summary section:

Subtotal	AS 49,00
Shipping Ground	AS 5,00
Shipping Discount	- AS 2,50
<b>Estimated Total</b>	<b>AS 51,50</b>

Payment options:

ONE TIME PAYMENT AS 51.50  4 INTEREST FREE PAYMENTS OF AS 12.88

Navigation: « Continue Shopping **CHECKOUT**

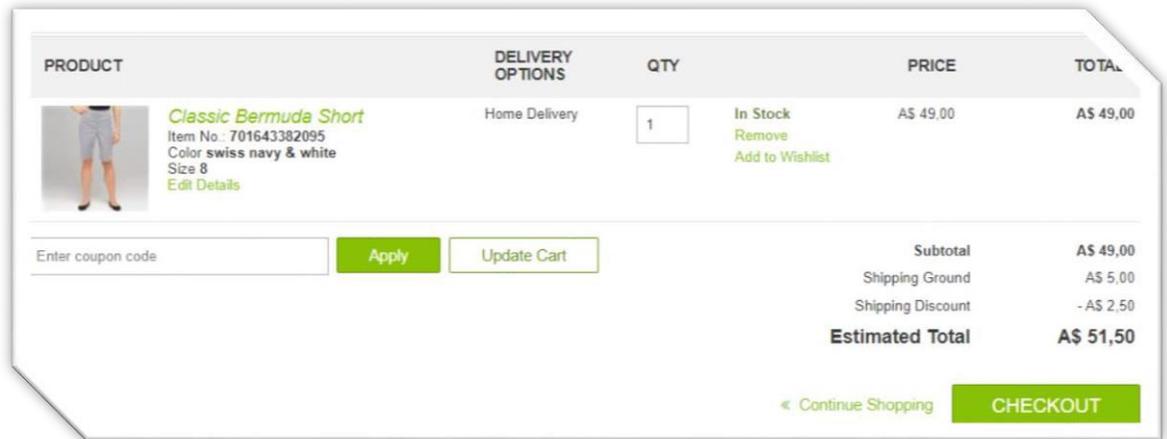
2. 'Limepay Payment Options' set as 'multiple' and 'Limepay Cart Widget Mode' set as 'textual'. Basket do not contain pay later disabled product. Cart widget would be as shown below

PRODUCT	DELIVERY OPTIONS	QTY	PRICE	TO
 <p><b>Classic Bermuda Short</b> Item No.: 701643382095 Color swiss navy &amp; white Size 8 <a href="#">Edit Details</a></p>	Home Delivery	1	In Stock <a href="#">Remove</a> <a href="#">Add to Wishlist</a>	A\$ 49,00
<input type="text" value="Enter coupon code"/> <input type="button" value="Apply"/> <input type="button" value="Update Cart"/>				<p>Subtotal A\$ 49,00</p> <p>Shipping Ground A\$ 5,00</p> <p>Shipping Discount -A\$ 2,50</p> <p><b>Estimated Total A\$ 51,50</b></p>
<p>or 4 Interest Free Payments of <b>A\$ 12.88</b></p>				
<p><a href="#">« Continue Shopping</a></p>				<input type="button" value="CHECKOUT"/>

3. 'Limepay Payment Options' set as 'splitOnly' and 'Limepay Cart Widget Mode' set as 'toggle' or 'textual'. Basket do not contain pay later disabled product. Cart widget would be as shown below

PRODUCT	DELIVERY OPTIONS	QTY	PRICE	TO
 <p><b>Classic Bermuda Short</b> Item No.: 701643382095 Color swiss navy &amp; white Size 8 <a href="#">Edit Details</a></p>	Home Delivery	1	In Stock <a href="#">Remove</a> <a href="#">Add to Wishlist</a>	A\$ 49,00
<input type="text" value="Enter coupon code"/> <input type="button" value="Apply"/> <input type="button" value="Update Cart"/>				<p>Subtotal A\$ 49,00</p> <p>Shipping Ground A\$ 5,00</p> <p>Shipping Discount -A\$ 2,50</p> <p><b>Estimated Total A\$ 51,50</b></p>
<p>or 4 Interest Free Payments of <b>A\$ 12.88</b></p>				
<p><a href="#">« Continue Shopping</a></p>				<input type="button" value="CHECKOUT"/>

- When basket contains at least one pay later disabled product AND/OR Limepay pay later threshold exceeded for cart total, Limepay preferences matches either of the above configurations 1,2 or 3 OR when 'Limepay Enabled' is set as 'no', cart widget would not be shown



**NB:** Threshold range configured for pay later payment 'Limepay\_instalment' has to be in sync with range provided by Limepay for the merchant to have unified checkout payment and widget experience among product, cart and checkout pages.

Synchronising the range between systems is to keep the behaviour of the widget and checkout iFrame identical (product pages / quick views widget don't rely on threshold check). For example when both payment modes are configured for site, and cart widget is set for toggle, if cart total exceeds pay later threshold no widget will be shown. Likewise iFrame at checkout won't allow pay later payment. If cart widget is set for textual widget and cart total exceeds pay later threshold no widget will be shown and checkout iFrame will be rendered with a message signalling pay later threshold breach.

## Pay later threshold range configuration

Configure range provided by Limepay for merchant under payment method

Limepay	Card Payment or Payment Plan	Yes	11
<b>Limepay_instalment</b>	<b>Limepay Pay Later configuration</b>	<b>Yes</b>	<b>12</b>
PayFlex	Payflex	Yes	10
PayPal	Pay Pal	No	6
QuadPay	QuadPay	Yes	9
Zip	Zip	Yes	8

**Limepay\_instalment Details**

Image:  [Select](#)

Payment Processor:

Countries: All [Edit](#)

Currencies: (1) AUD [Edit](#)

Customer Groups: All [Edit](#)

Min/Max Payment Ranges: **Min/Max Payment Ranges**

A\$	<input type="text" value="5"/>	to	<input type="text" value="1200"/>
¥	<input type="text"/>	to	<input type="text"/>
€	<input type="text"/>	to	<input type="text"/>

Since 'Limepay' is the main payment method is used for checkout no range shall be set here

GIFT_CERTIFICATE	Gift Certificate	Yes	4
<b>Limepay</b>	<b>Card Payment or Payment Plan</b>	<b>Yes</b>	<b>11</b>
Limepay_instalment	Limepay Pay Later configuration	Yes	12
PayFlex	Payflex	Yes	10
PayPal	Pay Pal	No	6
QuadPay	QuadPay	Yes	9
Zip	Zip	Yes	8

**Limepay Details**

Description:

Image:  [Select](#)

Payment Processor:

Countries: All [Edit](#)

Currencies: (1) AUD [Edit](#)

Customer Groups: All [Edit](#)

Min/Max Payment Ranges: **Min/Max Payment Ranges**

A\$	<input type="text"/>	to	<input type="text"/>
-----	----------------------	----	----------------------

## 8.2.5. Checkout Content & Stages

Limepay payment section for checkout is rendered within an iFrame appearing when the customer selects Limepay payment method for billing. The customer must enter the card details in the iFrame and then proceeds. Upon progressing the customer might encounter subsequent stages if applicable within the same iFrame, that has to be filled in before completing the order. The subsequent stages for one time / pay later checkout could be as follows :

### One time settlement stages

- 3D Secure page for 3DS supported cards

### Pay later payment stages

- email/phone OTP verification
- Identity Details – Name/address/DOB details for KYC (for new customer who have not KYCed before)
- Identity Details – Passport/DL details for KYC (for new customer who have not KYCed before)
- Review payment plan, accepting terms and conditions
- 3D Secure page for 3DS supported cards

As the customer advances with each stage within the iFrame and click on the place order button, a payment token is generated by the Limepay API and saved against the order to be used for the create order service. Customer then lands on order summary page, and upon clicking the place order button a service call is triggered to Limepay to complete the order. On successful service response customer gets the order placed and lands on the order confirmation page.

Limepay payment iFrame in checkout billing :

The screenshot displays the Limepay payment interface within a checkout iFrame. At the top, it says "SELECT PAYMENT METHOD" with a red asterisk and "REQUIRED". Below this, there are two radio button options: "Credit Card" (unselected) and "Card Payment or Payment Plan" (selected). To the right of the second option are logos for VISA, Mastercard, and American Express. Below the radio buttons, there is a small text "Description Test on Development". The main content area is divided into two sections. The first section has a dark blue header "Pay \$2998 today" with a checkmark icon and "Pay now in full" below it. The second section has a light blue header "Pay \$7.50 today" with a radio button icon and "Buy Now Pay Later" below it. Below these sections is a "Payment source" section with a dropdown menu currently showing "Credit/Debit Card" with VISA and Mastercard logos. At the bottom of the iFrame, there is a green button that says "CONTINUE TO PLACE ORDER >".

iFrame content when toggle switched to pay later option

<b>\$7.50</b> 17 JUN	<b>\$7.49</b> 1 JUL	<b>\$7.49</b> 15 JUL
-------------------------	------------------------	-------------------------

**Contact Details**

Phone  
+61 9895292329

Email  
abduravoof@gmail.com

By continuing, you consent to your personal information being disclosed and used to help check and verify your identity. Limepay collect and use the information you provide to administer your payment plan on behalf of the merchant. It is a condition of the payment plan that you provide this information. You can check the information we hold about you at any time. You can learn more by reading our [Privacy Policy](#) and [Terms of Use](#)

**Verify contact details**

## 3D Secure page (test page for sandboxes)

Credit Card  Card Payment or Payment Plan

pay upfront or in instalments, using your credit or debit card

To protect your card against unauthorised use, enter your bank issued code below.  
They may reference our payment provider, Limepay, and pre-authorise a zero amount.

# 3D Secure Test Page

Complete a required action for this test.

**What is this page?**

If you unexpectedly reached this page while checking out on a website, **you won't be charged**. Please reach out to us with the URL of this page in your message.

[Contact support →](#)

This is a **3D Secure non-payment authentication test page**.  
In live mode, customers will be asked to verify their identity with a push notification, a text message, or another method chosen by their bank.

[COMPLETE AUTHENTICATION](#)

[FAIL AUTHENTICATION](#)

[CONTINUE TO PLACE ORDER >](#)

**NB:** 3DS page is shown within the iFrame in subsequent stages of customer billing form

## Pay later email & phone OTP verification

### DID YOU RECEIVE THE CODE(S)?

We've sent codes to +917907163740 and fiaz.raffi@tryzens.com

Mobile code  Email code

[Resend Code](#) [Resend Code](#)

[Confirm →](#)

[Go back](#)

## Identity Details

### IDENTITY VERIFICATION

We need to collect some personal information to verify your identity

**Name:**

**Date of Birth:**

**Australian Residential Address:**

**Identification Method:**

[Verify identity →](#)

[Go back](#)

Pay later review payment plan stage

← Back

### YOUR PAYMENT PLAN

**Due Today: 25.14** [Change todays payment amount](#) →

3 fortnightly payments of 25.12, starting 3 Jul 2021 [Change](#) →

---

I've read and agree to the [Payment Plan Terms](#) and consent to the collection, use and disclosure of my personal information in accordance with the Limepay [Privacy Policy](#)

Pay later change today's payment

### TODAY'S PAYMENT

How much would you like to pay today? (Minimum 25.14)  
We may add a couple of cents so your future payments are equal.

\$ 25.14

---

← Cancel [Update](#) →

Pay later change payment dates (+/- 2days from the proposed dates)

← Back

### YOUR PAYMENT PLAN

FIRST PAYMENT	SECOND PAYMENT
<b>19 JUN 2021</b>	<b>3 JUL 2021</b>
\$30.00	\$23.50
<a href="#">Change</a>	<a href="#">Change</a>
THIRD PAYMENT	FOURTH PAYMENT
<b>17 JUL 2021</b>	<b>31 JUL 2021</b>
\$23.50	\$23.50
<a href="#">Change</a>	<a href="#">Change</a>

---

I've read and agree to the [Payment Plan Terms](#) and consent to the collection, use and disclosure of my personal information in accordance with the Limepay [Privacy Policy](#)

Once customer has successfully advanced through all stages applicable for one time or pay later payments, customer can finally proceed for order completion.

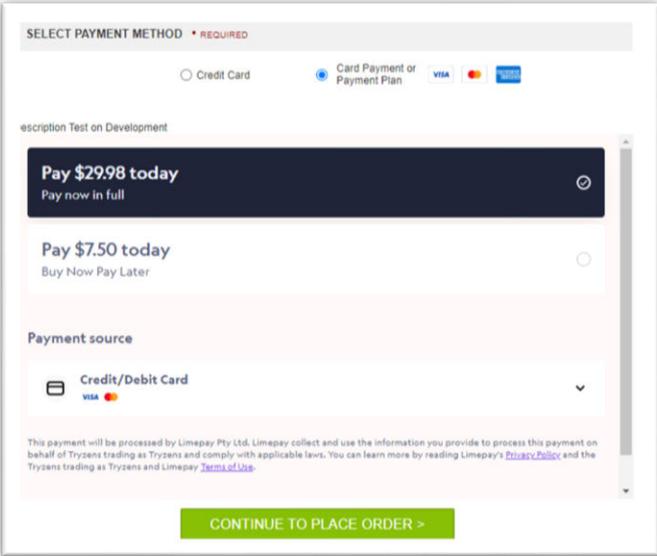
### 8.2.6. Customer Applicable Checkout View

The iFrame rendered for Limepay payment section differs based on configuration and customer basket state. Let us see different possible payment iFrame checkout options offered to customer based on following rules :

#### A. Limepay one time and pay later allowed

When both mode of payments are configured for the site (preference '*Limepay Payment Options*' setas '*multiple*')

1. And storefront customer has no pay later disabled product in cart nor pay later threshold exceeded for cart total, iFrame renders as shown below. Customer can choose between the 2 modes for payment for checkout by toggling to the desired state



The screenshot displays the 'SELECT PAYMENT METHOD' section of a checkout page. At the top, it indicates 'REQUIRED' and offers two radio button options: 'Credit Card' (unselected) and 'Card Payment or Payment Plan' (selected). Below these are logos for VISA, Mastercard, and AMEX. A note reads 'Description Test on Development'. Two payment options are listed: 'Pay \$29.98 today' (Pay now in full) with a checked radio button, and 'Pay \$7.50 today' (Buy Now Pay Later) with an unchecked radio button. Under 'Payment source', a dropdown menu shows 'Credit/Debit Card' with VISA and Mastercard logos. A disclaimer at the bottom states: 'This payment will be processed by Limepay Pty Ltd. Limepay collect and use the information you provide to process this payment on behalf of Trzyens trading as Trzyens and comply with applicable laws. You can learn more by reading Limepay's [Privacy Policy](#) and the Trzyens trading as Trzyens and Limepay [Terms of Use](#).' A green button at the bottom says 'CONTINUE TO PLACE ORDER >'.

2. Pay later disabled product added to cart by customer, iFrame renders as shown below. Limepay payment falls back to only allowing one time settlement for placing order and customer cannot toggle nor check out with pay later option

SELECT PAYMENT METHOD • REQUIRED

Credit Card  Card Payment or Payment Plan   

escription Test on Development

**Pay \$29.98 today**  
Pay now in full

**Pay \$7.50 today**  
Buy Now Pay Later

Payment source

 Credit/Debit Card  

1234 1234 1234 1234 

MM / YY CVC

This payment will be processed by Limepay Pty Ltd. Limepay collect and use the information you provide to process this payment on behalf of Tryzens trading as Tryzens and comply with applicable laws. You can learn more by reading Limepay's [Privacy Policy](#) and the Tryzens trading as Tryzens and Limepay [Terms of Use](#).

**CONTINUE TO PLACE ORDER >**

3. Pay later threshold exceed for cart total (no pay later disabled product in cart), iFrame renders as shown below. Limepay payment falls back to only allowing one time settlement for placing order and customer cannot toggle nor check out with pay later option

SELECT PAYMENT METHOD • REQUIRED

Credit Card  Card Payment or Payment Plan   

escription Test on Development

**Pay \$29.98 today**  
Pay now in full

**Pay \$7.50 today**  
Buy Now Pay Later

Payment source

 Credit/Debit Card  

This payment will be processed by Limepay Pty Ltd. Limepay collect and use the information you provide to process this payment on behalf of Tryzens trading as Tryzens and comply with applicable laws. You can learn more by reading Limepay's [Privacy Policy](#) and the Tryzens trading as Tryzens and Limepay [Terms of Use](#).

**CONTINUE TO PLACE ORDER >**

4. Basket contains pay later disqualifying products (rule A2 above) and basket total also exceeds pay later threshold (rule A3 above), rule A2 takes precedence over A3. Here Limepay payment falls back to only allowing one time settlement for placing order

SELECT PAYMENT METHOD \* REQUIRED

Credit Card  Card Payment or Payment Plan

Description Test on Development

**Pay \$29.98 today**  
Pay now in full

**Pay \$7.50 today**  
Buy Now Pay Later

Payment source

Credit/Debit Card

1234 1234 1234 1234

MM / YY CVC

This payment will be processed by Limepay Pty Ltd. Limepay collect and use the information you provide to process this payment on behalf of Tryzens trading as Tryzens and comply with applicable laws. You can learn more by reading Limepay's [Privacy Policy](#) and the Tryzens trading as Tryzens and Limepay [Terms of Use](#).

CONTINUE TO PLACE ORDER >

## B. Limepay pay later only allowed

When only pay later payment is configured for the site (preference 'Limepay Payment Options' set as 'splitOnly')

1. And storefront customer has no pay later disabled product in cart nor pay later threshold exceeds, iFrame renders as shown below. Customer can only checkout with pay later advancing through different stages, one time settlements not allowed for site

SELECT PAYMENT METHOD \* REQUIRED

Credit Card  Card Payment or Payment Plan

Description Test on Development

**Pay \$29.98 today**  
Pay now in full

**Pay \$7.50 today**  
Buy Now Pay Later

Payment source

Credit/Debit Card

This payment will be processed by Limepay Pty Ltd. Limepay collect and use the information you provide to process this payment on behalf of Tryzens trading as Tryzens and comply with applicable laws. You can learn more by reading Limepay's [Privacy Policy](#) and the Tryzens trading as Tryzens and Limepay [Terms of Use](#).

CONTINUE TO PLACE ORDER >

2. Pay later disqualifying product added to cart, no Limepay payment method option is shown for checkout rather only shows other applicable payment methods. Limepay payment stays disabled entirely for customer until basket is cleared form pay later disabled products.

A screenshot of a credit card payment form. At the top, there is a radio button labeled "Credit Card" which is selected. Below this is a dropdown menu labeled "Select Credit Card" with the text "Select from saved cards". The form contains several input fields: "Name on Card", "Type" (with "Visa" selected), "Number" (with an example "4111111111111111" below it), and "Expiration Date" (with "January" and "2016" selected). There is also a "Security Code" field with a "What is this?" link. At the bottom, there is a checked checkbox labeled "Save this card" and a button labeled "CONTINUE TO PLACE ORDER >".

3. Pay later threshold exceed for cart total and no pay later disabled product in cart, iFrame renders as shown below. A message is displayed to customer within iFrame signalling breach of pay later threshold. Customer cannot checkout using Limepay in this scenario unless cart total is adjusted within the stipulated range

A screenshot of a payment plan offer. At the top, there are two radio buttons: "Credit Card" (unselected) and "Card Payment or Payment Plan" (selected). To the right of the second radio button are logos for VISA, Mastercard, and American Express. Below the radio buttons is a text box containing the message: "Pay upfront or in instalments, using your credit or debit card" and "Split the cost over 4 payments on purchases of up to \$1200.00.". At the bottom, there is a button labeled "CONTINUE TO PLACE ORDER >".

- Basket contains pay later disabled product (rule B2 above) and basket total exceeds pay later threshold (rule B3 above), rule B.2. takes precedence over B. 3. and customer iFrame behavior would be the same as of rule B3. Where no Limepay payment methods are offered for checkout. Customer must proceed with other payment methods for placing order

The image shows a credit card payment form within a rounded rectangular frame. At the top, there is a radio button labeled "Credit Card" which is selected. Below this is a dropdown menu labeled "Select Credit Card" with the text "Select from saved cards" and a downward arrow. The form contains several required fields, each with a red bullet point: "Name on Card" (text input), "Type" (dropdown menu showing "Visa"), "Number" (text input with an example "4111111111111111" below it), and "Expiration Date" (two dropdown menus showing "January" and "2016"). There is also a "Security Code" field (text input) with a "What is this?" link in green text. At the bottom, there is a checked checkbox labeled "Save this card" and a grey button labeled "CONTINUE TO PLACE ORDER >".

#### 8.2.7. Payment Method Description

Limepay display's a configurable description text *'Pay upfront or in instalments, using your credit or debit card'* to the top of the iFrame content once a customer has selected the Limepay payment method. See below

Merchant may decide to remove or amend the description text through payment methods configuration from Business Manager

### 8.2.8. Changing Billing Email / Phone

Limepay accepts customer`s billing address details for rendering the checkout iFrame. Any change made to billing form fields customer email (for guest user) or billing phone number after checkout iFrame has been loaded and customer focusses out of the field, iFrame auto refreshes. It does so for Limepay checkout API to accept the updated email/phone value. Note OTP is sent to phone and mailbox, and this is the reason why iFrame has to refresh to send codes to the last entered customer details.

For checking out with pay late payment, customer email address will be auto populated within the iFrame, whereas customer will have to reenter their billing phone number in E164 format (prefixed by + [country code]) onto the phone field within the iFrame. Limepay has a future enhancement in pipeline to auto populate the phone number in E164 onto the iFrame field rather than customer entering it manually as mentioned above.

#### 8.2.9. Limepay Control Switch

Limepay provides a control switch to turn off Limepay payments for site at checkout and hiding product/cart widgets. This is based on the preference '*Limepay Enabled*', setting it as '*No*' hides all instances of Limepay widgets and billing payment method on checkout for the site.

Note, for effective display of Limepay widgets and billing payment methods make sure the following configurations are correct with respect to the storefront currency allowed for the site :

- Preference '*Limepay Enabled*' is set as '*Yes*'
- Main payment method '*Limepay*' enabled status is '*Yes*' i.e, Active
- Main payment method '*Limepay*' is configured with currency "*AUD*" and storefront currency matches with the same

## 9. Refunds & Backend Operations

### 9.1. Refunds

The cartridge provides the ability for merchants to trigger refunds through code. As SFCC does not offer refund functionality OOB, merchant to implement the following code wherever applicable to trigger refund

Use the following code hooks to trigger a refund

```
if (HookMgr.hasHook('limepay.payment.refund')) {  
  // Invokes Limepay refund hooks  
  refundResult = HookMgr.callHook('limepay.payment.refund', 'ProcessRefund', order, refundAmount);  
}
```

Function Argument	Data Type	Value
order	dw.order.Order	SFCC Order object
refundAmount	String	Refunding amount Please refer <a href="#">link</a> for more details

## 9.2. Payment Flow Processes

It is important to know what backend processes takes place while placing a storefront order through Limepay. This knowledge gives merchant an idea of adding new customisations wherever required. The process begins all with rendering the billing payment form in iFrame. The iFrame is loaded when Limepay payment method at billing section is selected by the customer. At this moment all necessary billing details that the customer has entered is parsed by the Limepay Checkout API through `init()` and `render()` functions

File : {merchant\_core}\cartridge\js\limepay.js

```
LimepayCheckout.init({
  publicKey: context.publicKey,
  email: limepayEmail,
  phone: formattedPhoneNo,
  customerFirstName: billingDetails.firstName,
  customerLastName: billingDetails.lastName,
  customerResidentialAddress: billingDetails.customerResidentialAddress,
  customerToken: limepayCustomerToken,
  hidePayLaterOption: context.hidePayLaterOption,
  hideFullPayOption: context.hideFullPayOption,
  paymentType: context.preselectedPaymentType == 'full' ?
    'paycard' : 'payplan',
  paymentToken: function (token) {
    savePaymentToken(token);
  }
});
LimepayCheckout.render({
  elementId: 'limepay-cont',
  currency: order.orderInfo.currency,
  amount: order.orderInfo.amount,
  paymentType: context.preselectedPaymentType == 'full' ?
    'paycard' : 'payplan',
  showPayNow: false,
  showPayPlanSubmit: false,
  checkoutCSSOverride: context.checkoutCSSOverride,
  primaryColor: limepayPrimaryColor
});
LimepayCheckout.errorHandler(function (err) {
  // Handle errors
});
LimepayCheckout.eventHandler(function (event) {
  $('body').trigger('limePayment:handleLimePayEvents',
    { event: event });
});
```

Once customer has filled in all the payment details within the iFrame and advances through all subsequent stages and clicks on the place order button, then the Limepay submit() API is invoked

```
LimepayCheckout.submit();
```

Upon invoking `submit()`, Limepay generates a payment token for the amount initialized in the `init()` call and on success returns to the callback function `paymentToken()` initialized under the `init()`. Here `savePaymentToken(token)`. The purpose of the callback is to save the generated token that is used later for placing order

```
LimepayCheckout.init({  
  .  
  .  
  .  
  paymentToken: function (token) {savePaymentToken(token);  
  }  
});
```

**NB:** Changes to order total, billing phone and email needs after iFrame has been rendered and is in view to customer, the same changes has to be parsed again by the checkout API through `init()` and then iFrame has to be reloaded through `render()` function. This is because payment tokens are generated based on the order amount that is being declared inside the `init()`, and OTP has to be sent to the latest email/phone entered by the customer for pay later payment option. OOTB cart update scenarios do currently invoke the `init()` and `render()` methods, however any merchant specific requirement has to follow the same.

Inside the callback function the generated payment token and E164 formatted phone number from customer billing is saved to basket custom attributes. Later on upon clicking the placing order button from order summary page the same data is copied to order object and then used for the Pay Order service API.

Next, during place order Limepay executes two service calls one after the other to complete a transaction, they are Create Order followed by Pay Order API (Refer section 2.3). Make sure during integration the quota limit related to HTTPClient services are not exceeded that would lead to an error during checkout.

## 10. Testing

Important, test with cache *'enabled'* for the site as product page and quick views are cached. If product pages are configured for toggle widget, widget rendering is not at all cached in order to reflect the customer specific toggle state. But when configured for textual widget, widget renders from the usual product content cache. Cart and checkout pages have no caching so widget or iFrame would always be respective to current configuration and customer basket state. When testing with cache enabled, and switching preference values for product widget behavior *'Limepay Cart Widget Mode'*, requires cache to be cleared each time for rendering the expected widget behavior.

## 10.1. Checkout with One Time Payment

Add suitable products to cart and proceed by entering shipping details

### Customer Billing page

The screenshot displays the 'Customer Billing page' with the following sections:

- ENTER GIFT CERTIFICATE OR COUPON CODES**
  - Enter coupon code:** A text input field with an 'Apply' button and a 'Help' link.
  - Redeem Gift Certificate:** A text input field with 'Apply', 'Check Balance', and 'Help' buttons.
- SELECT PAYMENT METHOD • REQUIRED**
  - Credit Card
  - Card Payment or Payment Plan (with VISA, Mastercard, and American Express logos)
- Description Test on Development** (shown in a scrollable box)
  - Pay \$29.98 today** (Pay now in full) - Selected with a radio button.
  - Pay \$7.50 today** (Buy Now Pay Later) - Unselected with a radio button.
  - Payment source:** A dropdown menu currently showing 'Credit/Debit Card' with VISA and Mastercard logos.
  - This payment will be processed by Limepay Pty Ltd. Limepay collect and use the information you provide to process this payment on behalf of Tryzens trading as Tryzens and comply with applicable laws. You can learn more by reading Limepay's [Privacy Policy](#) and the Tryzens trading as Tryzens and Limepay [Terms of Use](#).
- CONTINUE TO PLACE ORDER >** (Green button)

### Payment Form

Customer could use the one time payment for checkout as outlined in above section by entering details onto the respective form. The payment form viewed could be either one according to following scenarios (based on the group of configurations and customer basket state)

1. 'Limepay Payment Options' set as 'multiple', basket has no pay later disabled products and cart total threshold is not exceeded. Customer has set the toggle towards one time payment option

The screenshot shows the 'SELECT PAYMENT METHOD' screen. At the top, there are two radio buttons: 'Credit Card' (unselected) and 'Card Payment or Payment Plan' (selected). Below this, there are two payment options: 'Pay \$2998 today' (selected) and 'Pay \$7.50 today' (unselected). The 'Pay \$2998 today' option is highlighted with a dark blue background. Below the payment options, there is a 'Payment source' section with a dropdown menu set to 'Credit/Debit Card'. At the bottom, there is a green button labeled 'CONTINUE TO PLACE ORDER >'. A small disclaimer at the bottom of the form reads: 'This payment will be processed by Limepay Pty Ltd. Limepay collect and use the information you provide to process this payment on behalf of Tryzens trading as Tryzens and comply with applicable laws. You can learn more by reading Limepay's [Privacy Policy](#) and the Tryzens trading as Tryzens and Limepay [Terms of Use](#).'

OR

2. 'Limepay Payment Options' set as 'multiple', basket has pay later disabled products and cart total threshold is not exceeded

The screenshot shows the 'SELECT PAYMENT METHOD' screen. At the top, there are two radio buttons: 'Credit Card' (unselected) and 'Card Payment or Payment Plan' (selected). Below this, there are two payment options: 'Pay \$2998 today' (unselected) and 'Pay \$7.50 today' (selected). The 'Pay \$7.50 today' option is highlighted with a light blue background. Below the payment options, there is a 'Payment source' section with a dropdown menu set to 'Credit/Debit Card'. Below the dropdown, there are input fields for the card number (1234 1234 1234 1234), the expiry date (MM / YY), and the CVC. At the bottom, there is a green button labeled 'CONTINUE TO PLACE ORDER >'. A small disclaimer at the bottom of the form reads: 'This payment will be processed by Limepay Pty Ltd. Limepay collect and use the information you provide to process this payment on behalf of Tryzens trading as Tryzens and comply with applicable laws. You can learn more by reading Limepay's [Privacy Policy](#) and the Tryzens trading as Tryzens and Limepay [Terms of Use](#).'

OR

3. 'Limepay Payment Options' set as 'multiple', basket has no pay later disabled products and cart total threshold is exceeded

SELECT PAYMENT METHOD • REQUIRED

Credit Card  Card Payment or Payment Plan   

Description Test on Development

**Pay \$29.98 today**   
Pay now in full

Pay \$7.50 today   
Buy Now Pay Later

Payment source

 Credit/Debit Card 

This payment will be processed by Limepay Pty Ltd. Limepay collect and use the information you provide to process this payment on behalf of Tryzens trading as Tryzens and comply with applicable laws. You can learn more by reading Limepay's [Privacy Policy](#) and the Tryzens trading as Tryzens and Limepay [Terms of Use](#).

**CONTINUE TO PLACE ORDER >**

OR

4. 'Limepay Payment Options' set as 'multiple', basket has pay later disabled products and cart total threshold is exceeded

SELECT PAYMENT METHOD • REQUIRED

Credit Card  Card Payment or Payment Plan   

Description Test on Development

**Pay \$29.98 today**   
Pay now in full

Pay \$7.50 today   
Buy Now Pay Later

Payment source

 Credit/Debit Card 

1234 1234 1234 1234 

MM / YY  CVC

This payment will be processed by Limepay Pty Ltd. Limepay collect and use the information you provide to process this payment on behalf of Tryzens trading as Tryzens and comply with applicable laws. You can learn more by reading Limepay's [Privacy Policy](#) and the Tryzens trading as Tryzens and Limepay [Terms of Use](#).

**CONTINUE TO PLACE ORDER >**

Enter the one time settlement card details. Proceed through 3DS page if applicable and click on place order button to proceed to the order summary page

## Order Summary Page

STEP 1: Shipping > STEP 2: Billing > **STEP 3: Place Order** Help? 800-555-015

PRODUCT	QTY	TOTAL
 <b>Classic Bermuda Short</b> Item No.: 701643382088 Color swiss navy & white Size 6	1 In Stock	A\$ 49,00
<b>Subtotal</b>		<b>A\$ 49,00</b>
Shipping Ground		A\$ 5,00
Shipping Discount		- A\$ 2,50
<b>Order Total</b>		<b>A\$ 51,50</b>

[Edit Cart](#) **PLACE ORDER**

**ORDER SUMMARY** [Edit](#)

**Subtotal** A\$ 49,00  
[Edit](#) Shipping Ground A\$ 5,00  
Shipping Discount - A\$ 2,50  
**Order Total: A\$ 51,50**

**SHIPPING ADDRESS** [Edit](#)

Ella Ella S Hirschfeld  
41 Blairgowrie Avenue  
SPRINGFIELD, NSW 2630  
country.AU  
Method: Ground

**BILLING ADDRESS** [Edit](#)

Ella Ella S Hirschfeld  
41 Blairgowrie Avenue  
SPRINGFIELD, NSW 2630  
country.AU

**PAYMENT METHOD** [Edit](#)

Card Payment or Payment Plan  
Amount: A\$ 51,50

Click the place order button on summary page, to complete the order

## Order Confirmation Page

Thank you for your order.

If you have questions about your order, we're happy to take your call (800-555-0199) Monday - Friday, 8AM - 8PM

 Order Number: **00012106**

Order Placed: **June 25, 2021**

#### PAYMENT METHOD

Card Payment or Payment Plan  
Amount: A\$ 51,50

#### BILLING ADDRESS

Ella Ella S Hirschfeld  
41 Blairgowrie Avenue  
SPRINGFIELD, NSW 2630  
country:au  
Phone: (02) 6190 4180

#### SHIPMENT NO. 1

##### SHIPPING STATUS:

Not Shipped

##### METHOD:

Ground

##### SHIPPING TO

Ella Ella S Hirschfeld  
41 Blairgowrie Avenue  
SPRINGFIELD, NSW 2630  
country:AU  
(02) 6190 4180

#### PAYMENT TOTAL

**Subtotal** A\$ 49,00

Shipping Ground A\$ 5,00

Shipping Discount - A\$ 2,50

**Order Total:** A\$ 51,50

#### ITEM

Classic Bermuda Short

SKU: 701643382088

Swiss navy & white

#### QTY

1

#### PRICE

A\$ 49,00

## 10.2. Checkout with Pay Later Payment

Add suitable products to cart and proceed by entering shipping details

### Customer Billing page

ENTER GIFT CERTIFICATE OR COUPON CODES

Enter coupon code   [Help](#)

Redeem Gift Certificate    [Help](#)

SELECT PAYMENT METHOD • REQUIRED

Credit Card  Card Payment or Payment Plan   

Description Test on Development

**Pay \$29.98 today**  
Pay now in full

**Pay \$7.50 today**  
Buy Now Pay Later

Payment source

 **Credit/Debit Card**  

This payment will be processed by Limepay Pty Ltd. Limepay collect and use the information you provide to process this payment on behalf of Tryzens trading as Tryzens and comply with applicable laws. You can learn more by reading Limepay's [Privacy Policy](#) and the Tryzens trading as Tryzens and Limepay [Terms of Use](#).

## Payment Form

Customer could use the pay later payment for checkout as outlined in above section by entering details onto the respective form. The payment form viewed could be either one according to following scenarios (based on the group of configurations and customer basket state)

1. 'Limepay Payment Options' set as 'multiple', basket has no pay later disabled products and cart total threshold is not exceeded. Customer has set the toggle towards pay later payment option

**SELECT PAYMENT METHOD** • REQUIRED

Credit Card  Card Payment or Payment Plan   

Description Test on Development

**Pay \$29.98 today**  
Pay now in full

**Pay \$7.50 today**  
Buy Now Pay Later

**Payment source**

 **Credit/Debit Card**   

This payment will be processed by Limepay Pty Ltd. Limepay collect and use the information you provide to process this payment on behalf of Tryzens trading as Tryzens and comply with applicable laws. You can learn more by reading Limepay's [Privacy Policy](#) and the Tryzens trading as Tryzens and Limepay [Terms of Use](#).

**CONTINUE TO PLACE ORDER >**

OR

2. 'Limepay Payment Options' set as 'splitOnly', basket has no pay later disabled products and cart total threshold is not exceeded

SELECT PAYMENT METHOD • REQUIRED

Credit Card  Card Payment or Payment Plan   

Description Test on Development

**Pay \$29.98 today**   
Pay now in full

**Pay \$7.50 today**   
Buy Now Pay Later

Payment source

 **Credit/Debit Card**   
VISA

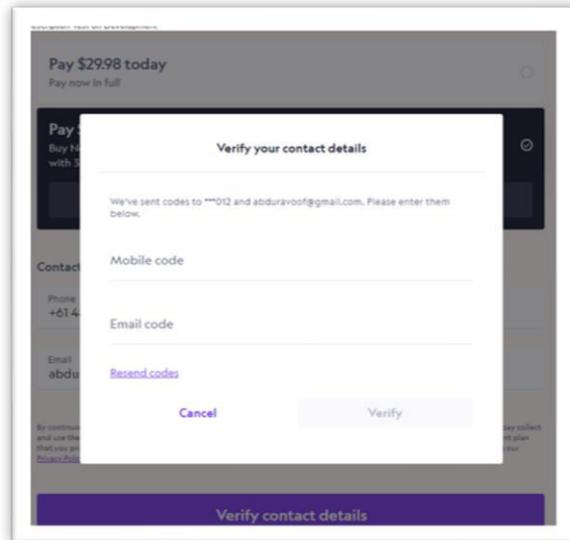
1234 1234 1234 1234 

MM / YY  CVC

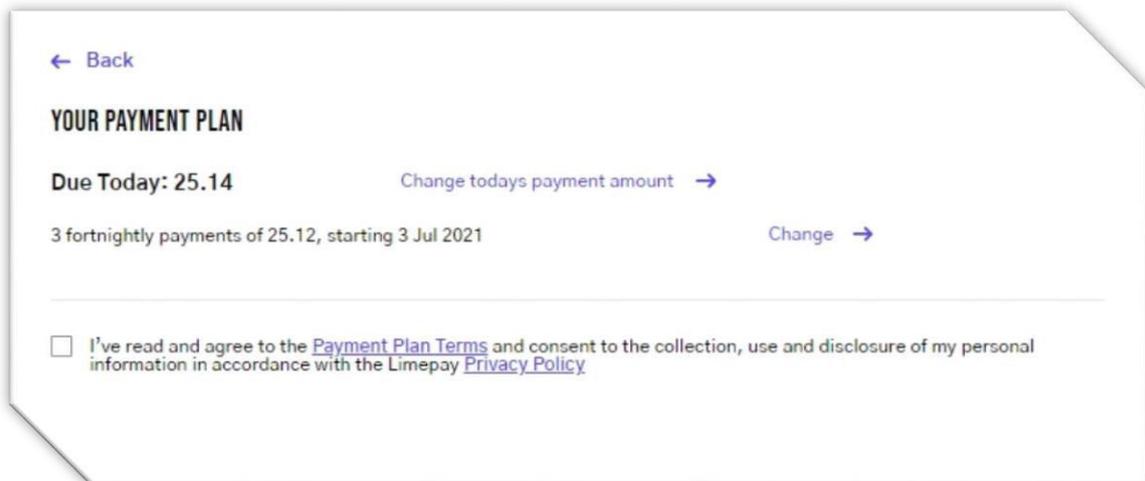
This payment will be processed by Limepay Pty Ltd. Limepay collect and use the information you provide to process this payment on behalf of Tryzens trading as Tryzens and comply with applicable laws. You can learn more by reading Limepay's [Privacy Policy](#) and the Tryzens trading as Tryzens and Limepay [Terms of Use](#).

**CONTINUE TO PLACE ORDER >**

Enter the pay later payment details. Proceed to subsequent OTP verification stage by clicking on the 'Review payment plan' button within the iFrame.



Enter OTP received on billing email & phone number inbox. Then click on 'Confirm' button on iFrame and proceed to next stage



Customer can adjust the payment plan in this stage, and finally check the terms and condition checkbox before clicking on the place order button to proceed to the order summary page

## Order Summary Page

PRODUCT	QTY	TOTAL
 <b>Classic Bermuda Short</b> Item No.: 701643382088 Color swiss navy & white Size 6	1 In Stock	AS 49,00
Subtotal		AS 49,00
Shipping Ground		AS 5,00
Shipping Discount		- AS 2,50
<b>Order Total</b>		<b>AS 51,50</b>

[Edit Cart](#) [PLACE ORDER](#)

**ORDER SUMMARY** [Edit](#)

Subtotal AS 49,00  
[Edit Shipping Ground](#) AS 5,00  
Shipping Discount - AS 2,50  
**Order Total: AS 51,50**

**SHIPPING ADDRESS** [Edit](#)

Ella S Hirschfeld  
41 Blairgowrie Avenue  
SPRINGFIELD, NSW 2630  
country.AU  
Method: Ground

**BILLING ADDRESS** [Edit](#)

Ella S Hirschfeld  
41 Blairgowrie Avenue  
SPRINGFIELD, NSW 2630  
country.AU

**PAYMENT METHOD** [Edit](#)

Card Payment or Payment Plan  
Amount: AS 51,50

Click the place order button on summary page, to complete the order

## 3DS Verification during Placing Order

Shipping Ground AS 49,00  
Order Total AS 1.240,98

[Edit Cart](#) [PLACE ORDER](#)

**SHIPPING ADDRESS** [Edit](#)

Test Test Test  
1 Elgin Parade  
Centre, NSW 4870  
country.AU  
Method: Ground

**BILLING ADDRESS** [Edit](#)

Test Test Test  
1 Elgin Parade  
Centre, NSW 4870  
country.AU

**PAYMENT METHOD** [Edit](#)

Card Payment or Payment Plan  
Amount: AS 1.240,98

### Test Payment Page

This is a test payment of \$1,240.98 AUD using 3D Secure.  
In live mode, customers will be asked to verify their identity with a push notification, a text message, or another method chosen by their bank.

[COMPLETE AUTHENTICATION](#) [FAIL AUTHENTICATION](#)

**FOLLOW US**

GET SPECIAL OFFERS & NEWS ON THE LATEST STYLES

Check Order Gift Certificates Privacy  
Wish List Help Terms  
Gift Registry Site Map Jobs

Enter Your Email

© 2016 hannah.com.au. All Rights Reserved

### 3DS Verification error during Placing Order

salesforce commerce cloud    

NEW ARRIVALS WOMENS MENS ELECTRONICS TOP SELLERS

STEP 1: Shipping > **STEP 2: Billing** > STEP 3: Place Order Help? 800-555-0199

Invalid payment confirmation request

**SELECT OR ENTER BILLING ADDRESS** • REQUIRED

• First Name

• Last Name

• Address 1

Address 2  APO/FPO

• City

• ZIP Code

• Country

• State

• Phone  Why is this required?  
Example: 333-333-3333

• Email

Please add me to Salesforce Commerce Cloud's email list. Salesforce Commerce Cloud does not share or sell personal info.  
[See Privacy Policy](#)

**ENTER GIFT CERTIFICATE OR COUPON CODES**

Enter coupon code   [Help](#)

Redeem Gift Certificate

[Help](#)

**SELECT PAYMENT METHOD** • REQUIRED

Credit Card  Card Payment or Payment Plan   

• Name on Card

• Type

• Number   
Example: 4111111111111111

**ORDER SUMMARY** [Edit](#)

Sony Erava® W-Series 40" LCD High Definition Television  
Qty: 1 AS 1,199.99

Subtotal	AS 1,199.99
Edi Shipping Ground	AS 40.99
<b>Order Total:</b>	<b>AS 1,240.98</b>

**SHIPPING ADDRESS** [Edit](#)

Test Test Test Test  
1 Marlin Parade  
Cairns, NSW 4870  
country:AU  
Method: Ground

**BILLING ADDRESS** [Edit](#)

Test Test Test Test  
1 Marlin Parade  
Cairns, NSW 4870  
country:AU

**PAYMENT METHOD** [Edit](#)

Card Payment or Payment Plan  
Amount: AS 1,240.98

### Order Confirmation Page

Thank you for your order.  
If you have questions about your order, we're happy to take your call (800-555-0199) Monday - Friday, 8AM - 8PM

**Order Number: 00012108**

Order Placed: June 25, 2021

**PAYMENT METHOD**  
Card Payment or Payment Plan  
Amount: AS 51.50

**BILLING ADDRESS**  
Ela S Hironaka  
42 Berginra Avenue  
SPRINGFIELD, NSW 2430  
Australia  
Phone: (61) 9430 0280

**SHIPMENT NO. 1**

**SHIPPING STATUS:**  
Not Shipped  
**METHOD:**  
Ground

**SHIPPING TO**  
Ela S Hironaka  
42 Berginra Avenue  
SPRINGFIELD, NSW 2430  
country:AU  
PO: APO FPO US0

	<b>QTY</b> 1	<b>PRICE</b> AS 49.00
-------------------------------------------------------------------------------------	-----------------	--------------------------

<b>PAYMENT TOTAL:</b>	
Subtotal	AS 49.00
Shipping Ground	AS 5.00
Shipping Discount	-AS 2.50
<b>Order Total:</b>	<b>AS 51.50</b>

### 10.3. Test Cards

Refer [link](#), for Limepay test card details

Refer [link](#), for Limepay pay later plan test values

Please check with Limepay for 3DS and other test details

## 11. Operation, Maintenance

### 11.1. Availability

In case of problems with the connection to Limepay Payments, please contact Limepay support

Please supply as much information as possible (Merchant account, time, order number,). Also, check the log files.

### 11.2. Failover / Recovery Process

In case the Limepay service is unavailable, the user will not be able to checkout using Limepay payment methods and an error message would be displayed on the checkout pages.

The service availability can be tracked in SFCC using the Service Status.

### 11.3.Support

In case of problems with the integration, missing features, etc. please contact the Limepay support

## 12. Appendix

### 12.1. System Extensions

In addition to the changes mentioned above, the below system extensions were done as part of the cartridge.

### 12.1.1. Order

ID	Purpose
limepayOrderId	Stores the Limepay order id generated during payment authorization in Create Order API
limepayPaymentToken	Stores the payment token generated at billing used for completing an order with Pay Order API. Value copied from equivalent basket attribute during place order.
limepayFormattedPhoneNo	Stores the E164 format phone number obtained from billing details. This is sent in with Pay Order API to complete order and value is copied from equivalent basket attribute during place prder.
limepayRefundResponse	Stores array of response objects containing refund id and amount obtained after Create Refund API

### 12.1.2. PaymentTransaction

ID	Purpose
limepayPaymentResponse	Stores the response of Pay Order API

### 12.1.3. Basket

ID	Purpose
limepayPaymentToken	Stores the payment token generated at billing before proceeding to order summary page
limepayFormattedPhoneNo	Stores the E164 format phone number obtained from billing details before proceeding to the order summary page

### 12.1.4. Site Preference

ID
limepayEnabled
limepayClientPublicKey
limepayClientSecretKey
limepayPaymentOptions
limepayMultiOptionDefault
limepayPDPWidgetMode
limepayCartWidgetMode

limepayScriptAPIUrl
limepayCustomerServiceEmail
3DSEnabled
limePayMin3DSAmount
limepayPrimaryColor
applePayDomainAssociation

Refer section for 6.1. for their purpose

#### 12.1.5. Category

ID	Purpose
disableLimepayPayLater	Stores pay later payment enable/disable status for product category

#### 12.1.6. Product

ID	Purpose
disableLimepayPayLater	Stores pay later payment enable/disable status for product

#### 12.1.7 Profile

ID	Purpose
limepayCustomerId	Stores limepay customer ID

### 12.2. Locating Code Changes

Developers can locate code changes done for the Limepay integration by searching for the following comments pattern

**JS**

```
// Limepay changes - START
```

Limepay code changes

```
// Limepay changes - END
```

**ISML**

<!-- Limepay changes - START -->

Limepay code changes

<!-- Limepay changes - END -->

## 13. Known Issues

No known issues

## 14. Release History

Version	Date	Changes
21.1.0	22 June 2021	Initial Release
22.1.0	22 March 2022	2 <sup>nd</sup> Release
22.2.0	26 June 2022	Logged in users - upserting the Limepay customer details. Primary color & JS changes. Enabling and verification of Apple Pay. Change of Message and Detail field for error messages.
22.2.1	04 July 2022	Update documentation