Lyra Payment Gateway

Quick start Guide

Document version 1.0

HISTORY OF THE DOCUMENT

Version	Author	Date	Comment
1.0	Lyra Network	Jan 2020	• Intial release

TECHNICAL SUPPORT

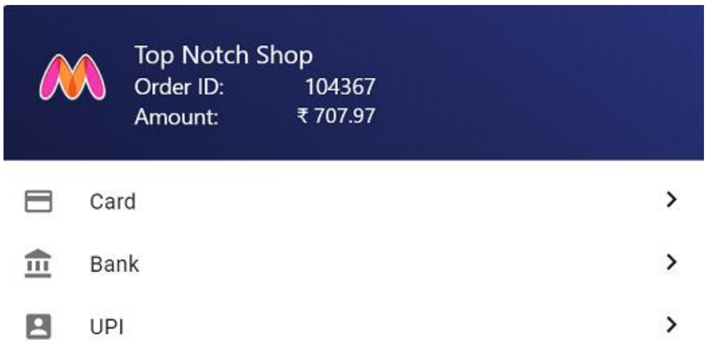
For technical inquiries or support, you can reach us from Monday to Friday, between 9am and 6pm

by phone:	+91 (022) 33864910 / 911
by email:	*support.pg.in@lyra.com*
from the Merchant Back Office:	(Menu: **Help** > **Contact support**)

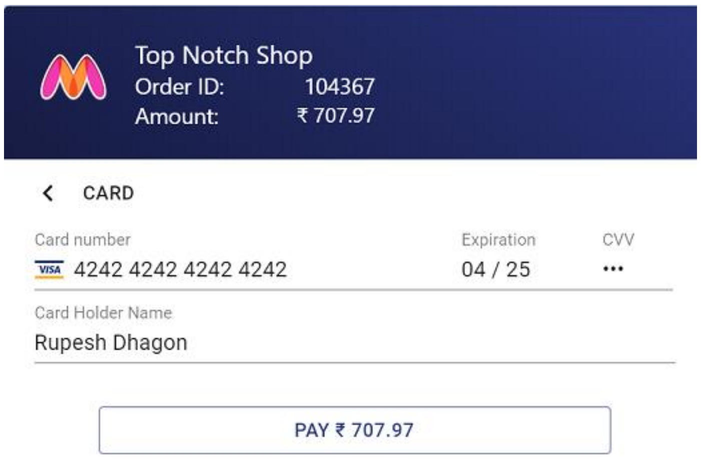
For any support request, please provide your shop ID (8-digit number).

1. Payment flow

1. At checkout time, the merchant website redirects the customer to the payment page hosted at Lyra Payment Gateway.

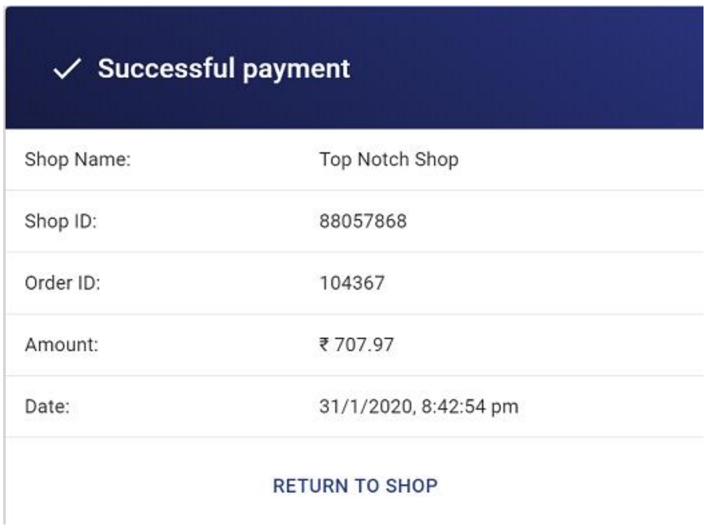


2. The customer selects a payment option and enters the payment details (e.g. card details, net banking option, UPI address, etc.)



3. The customer proceeds with the payment. Depending on the payment option selected, this may include a redirection to the bank page to enter OTP or to some external gateway.

4. The payment result is displayed



✓ Successful payment

Shop Name:	Top Notch Shop
Shop ID:	88057868
Order ID:	104367
Amount:	₹ 707.97
Date:	31/1/2020, 8:42:54 pm

5. (Optional) The customer clicks the "RETURN TO SHOP" link and is redirected back to the merchant web site. The return URL and HTTP method (POST or GET) is configured by the merchant.

6. (Optional) If the merchant has configured a callback URL, Lyra Payment Gateway automatically triggers a call to this URL. This mechanism is also referred to as IPN (Instant Payment Notification).

2. Interaction with the gateway

The merchant redirects the customer to Lyra gateway by posting an HTTP form with all the information required to proceed with the payment.

checkout URL	https://form.in.lyra.com/checkout/
method	POST
parameters	see below
encoding	UTF-8

2.1 Form parameters

Make sure to encode the form data in **UTF-8**.

Name	Description	Format	Sample data
Mandatory parameters			
signature	Signature to guarantee the data integrity	string	i2le2yRmA3+Sgs10ChZJgYcKW HsniBLcG//tYCnVTvM=
vads_version	version of the checkout form	enum	V2
vads_action_mode	Type of web integration for the payment page	enum	INTERACTIVE
vads_site_id	Identifier of the merchant online shop or website. Provided by Lyra.	string 8 characters	84373659
vads_ctx_mode	Test mode or live mode	enum	TEST or PRODUCTION
vads_order_id	Order id, provided by the merchant. Should be unique	string max 64 characters	allowed characters: alphabetic, numeric, space, dot (.), hyphen(-), underscore(_)
vads_amount	Amount to pay, in the smallest currency unit (e.g. in paisa for India).	numeric	154000 for 1,540.00 INR
vads_currency	Currency ISO 4217 alpha code	string 3 characters	INR
vads_cust_name	Customer full name	string	Rupesh Diwan
vads_cust_email	Customer email id	string	rupesh.diwan@bizbee.com
vads_cust_phone	Customer phone (landline or mobile)	string	9123465656
vads_cust_address	Customer address	string	Satguru Sachkand, apt 605, 3rd street
vads_cust_city	Customer city	string	Mumbai
vads_cust_state	Customer region or state	string	Maharashtra
vads_cust_zip	Customer Zip Code	string	400601
vads_cust_country	Customer country	string	India
vads_return_mode	HTTP method for the return URL	enum	NONE, GET, POST
Optional parameters			
vads_url_return	Return URL to the merchant web site at the end of the payment flow. If provided, overrides the return url configured in the back-office	string	htpps://myshop.co.in/checkout /return?pg=lyra
vads_redirect_success_timeout	Delay (in seconds) before automatic redirection is triggered from the result page to the merchant web site	numeric	60 for automatic redirection after 1 minute. 0 for immediate redirection.
vads_order_info	Additional free-text information related to	string	

	the order	max 255 characters	
vads_payment_option_code	Restrict the available payment options	enum list separated by semicolon	CARD, NET_BANKING, UPI, WALLET, EMI
vads_cust_id	Customer identifier. Required to propose **saved cards** option. **Must be unique**	string max 64 characters	

2.2 Signature

To secure the data exchange and to prevent the customer from tampering with the data, the data is signed with a secret key. Two different secret keys are available, one for TEST mode, one for PRODUCTION mode.

The secret keys shall remain on the merchant server and never be accessible from the client side (e.g. javascript in the client browser)

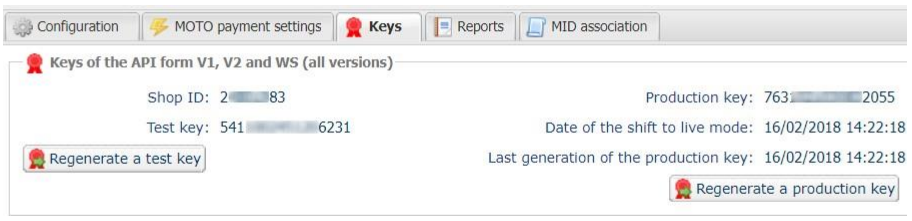
Make sure to use the right key to sign the form, depending on the vads_ctx_mode value.

The test and production keys are accessible from the Merchant back-office:

https://secure.payzen.co.in/vads-merchant/

Go to menu **Settings > Shop**

Select the **Keys** tab.



Keys of the API form V1, V2 and WS (all versions)

Shop ID: 2[redacted]83

Test key: 541[redacted]6231

Production key: 763[redacted]2055

Date of the shift to live mode: 16/02/2018 14:22:18

Last generation of the production key: 16/02/2018 14:22:18

Figure 1: Keys tab

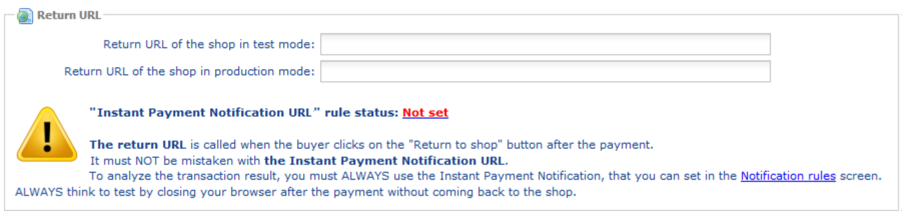
For security reason, the production key is hidden as soon as a successful transaction has been done. For signature computation, please refer to section 3.

2.3 Return URL

At the end of the payment sequence, the customer is redirected to the merchant web site on the return URL provided by the merchant.

return URL	configured by the merchant
method	POST or GET
parameters	optional, see below
encoding	UTF-8

The merchant can configure a static return URL in the Merchant Back Office via the menu **Settings** > **Shop** > **Configuration** tab:



Return URL

Return URL of the shop in test mode:

Return URL of the shop in production mode:

"Instant Payment Notification URL" rule status: Not set

The return URL is called when the buyer clicks on the "Return to shop" button after the payment.
It must NOT be mistaken with **the Instant Payment Notification URL.**
To analyze the transaction result, you must ALWAYS use the Instant Payment Notification, that you can set in the Notification rules screen.
ALWAYS think to test by closing your browser after the payment without coming back to the shop.

Figure 2: Setting up return URLs

If no return URL is set, the main shop URL is used, as defined in the **Details** section of the shop.

For a dynamic return URL, the merchant can use the form parameter **vads_return_url**. If this parameter is posted it overrides the default URL configured in the portal.

2.4 Return parameters

Return parameters are optional. If the form parameter vads_return_mode is empty or is set to NONE, no return parameter is posted. If the form parameter vads_return_mode=GET or POST, the parameters below are returned.

The signature in the return parameters shall always be checked against the other returned parameters. See section 3. for signature check.

Not checking the signature exposes the merchant to potential data tampering and fraud. E.g. a fraudster may intercept the return data from the client browser and change the charge status from DROPPED to PAID, or change the charge amount, etc.

Parameter	Description	Format	Sample data
signature	Signature to guarantee the data integrity	string	CmAVv4wujOnzEtBmovS4bf8Lg Mao0lAkaLz9/CQuwWY=
vads_version	version of the checkout form	enum	same as input
vads_site_id	shop identifier	string 8 characters	same as input
vads_ctx_mode	Triggers test mode or live mode	enum	same as input
vads_order_id	Order id. Should be unique	string max 64 characters	same as input
vads_amount	Payment amount in the smallest currency unit (e.g. in paisa for India).	numeric	same as in put
vads_currency	Currency, ISO 4217 alpha code	string 3 characters	same as input
vads_order_info	Additional free-text information related to the order	string max 255 characters	same as input
vads_charge_uuid	Unique id of charge resource	string 32 characters	c34747bf82044a70ab661cbe01aff 6a2
vads_charge_status	Charge status	enum	PAID, DROPPED
vads_drop_reason	Charge drop reason: expired, too many failed attempts, etc.	string	Too many failed attempts

Any additional parameter compared to the above list can be safely ignored. Those are legacy parameters which will be removed after some time.

2.5 Callback URL (IPN)

The merchant can configure a callback URL (also referred to as IPN URL) that is called automatically by the gateway each time a charge is either PAID or DROPPED.

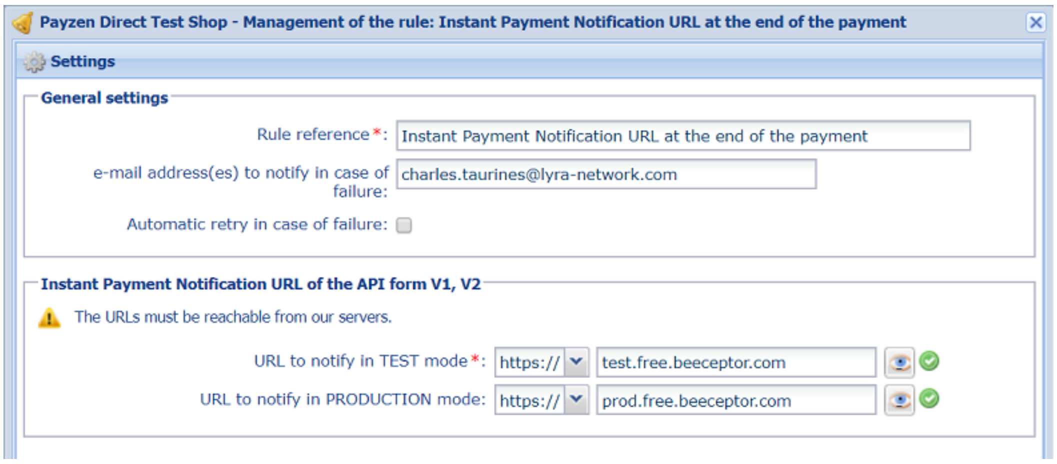
callback URL	configured by the merchant
method	POST
parameters	see below
encoding	UTF-8
gateway source IP	**194.50.38.0/24** (in case white listing is required on merchant backend)

The callback URL is configured from the Merchant back-office:

https://secure.payzen.co.in/vads-merchant/

Go to menu **Settings > Notification rules**

1. Right-click **Instant Payment Notification URL at the end of payment**.
2. Select **Manage the rule**.


Payzen Direct Test Shop - Management of the rule: Instant Payment Notification URL at the end of the payment

Settings

General settings

Rule reference*: Instant Payment Notification URL at the end of the payment

e-mail address(es) to notify in case of failure: charles.taurines@lyra-network.com

Automatic retry in case of failure: ☐

Instant Payment Notification URL of the API form V1, V2

The URLs must be reachable from our servers.

URL to notify in TEST mode*: https:// test.free.beeceptor.com

URL to notify in PRODUCTION mode: https:// prod.free.beeceptor.com

3. Enter the callback URL for **TEST** mod and for **PRODUCTION** mode
4. Enter the **E-mail id** to notify in case of failure. You can set multiple email ids separated with a semi-colon.
5. (Optional) Check the parameter for **Automatic retry in case of failure**. If checked, the gateway will try to call the callback URL up to 4 times.
6. Save the modifications.

2.6 Callback parameters

The parameters posted on the callback URL are similar to the parameters posted on the return URL. return parameters. See section 2.4.

The signature in the callback parameters shall be checked against the other returned parameters. See section 3. for signature check

Not checking the signature exposes the merchant to potential data tampering and fraud. E.g. a "man-in-the-middle" attacker may intercept the callback data and change the charge status from DROPPED to PAID, or change the charge amount, etc.

3. Signature computation

3.1 Digest algorithm

The same signature algorithm is used to sign the form parameters and to check the return parameters and the callback parameters.

Algorithm

1. Make sure that all parameters are encoded in **UTF-8**
2. Sort all the parameters with prefix "**vads_**" in **alphabetic** order
3. Concatenate the parameter **values** using the character "**+**" as a separator
4. Concatenate the result string with the **test** or **production secret key**, separated with "**+**"
5. Apply the **HMAC-SHA-256** hash function to the result string
6. Encode the hash result in **Base64**
7. The result is the signature.

⇨ Form posting: set the result in the **signature** parameter.

⇨ Return URL or callback URL: check the result with the signature parameter received. It should match!

Example of parameters sent to the payment gateway:

```
<form method="POST" action="https://form.in.lyra.com/checkout/">
<input type="hidden" name="vads_version" value="V2" />
<input type="hidden" name="vads_action_mode" value="INTERACTIVE" />
<input type="hidden" name="vads_site_id" value="12345678" />
<input type="hidden" name="vads_ctx_mode" value="TEST" />
<input type="hidden" name="vads_order_id" value="INV_ABC123" />
<input type="hidden" name="vads_order_info" value="Ref product #GHJFK48"/>
<input type="hidden" name="vads_amount" value="151200" />
<input type="hidden" name="vads_currency" value="INR" />
<input type="hidden" name="vads_cust_name" value="Rupesh Abhishek Diwan" />
<input type="hidden" name="vads_cust_phone" value="9892452635" />
<input type="hidden" name="vads_cust_email" value="rupesh.diwan@bizbee.com" />
<input type="hidden" name="vads_cust_address" value="Satguru compound B, apt. 305, S.V. Road"/>
<input type="hidden" name="vads_cust_city" value="Mumbai" />
<input type="hidden" name="vads_cust_state" value="Maharashtra" />
<input type="hidden" name="vads_cust_zip" value="400601" />
<input type="hidden" name="vads_cust_country" value="India" />
<input type="hidden" name="vads_return_mode" value="POST" />
<input type="hidden" name="signature" value="aeab3116f867d05680635ca6926b7a8d89a0ce34" />
<input type="submit" name="pay" value="Pay"/>
</form>
```

1. Parameter list, prefixed with "vads_", sorted alphabetically:
   - vads_action_mode

- vads_amount
- vads_ctx_mode
- vads_currency
- vads_cust_address
- vads_cust_city
- vads_cust_country
- vads_cust_email
- vads_cust_name
- vads_cust_phone
- vads_cust_state
- vads_cust_zip
- vads_order_id
- vads_order_info
- vads_return_mode
- vads_site_id
- vads_version

2. The values of these fields are concatenated using the "**+**" character:

```
INTERACTIVE+151200+TEST+INR+Satguru compound B, apt. 305, S.V. Road+Mumbai+India+rupesh.diwan@biz
bee.com+Rupesh Abhishek Diwan+9892452635+Maharashtra+400601+INV ABC123+Ref product #GHJFK48+POST
+12345678+V2
```

3. The value of the secret key is added at the end of the chain and separated with the "**+**" character. In this example, the test secret key is **1122334455667788**

```
INTERACTIVE+151200+TEST+INR+Satguru compound B, apt. 305, S.V. Road+Mumbai+India+rupesh.diwan@biz
bee.com+Rupesh Abhishek Diwan+9892452635+Maharashtra+400601+INV ABC123+Ref product #GHJFK48+POST
+12345678+V2+1122334455667788
```

4. The string above is hashed with algorithm HMAC-SHA-256 and encoded in Base64 format. Note: the hashing algorithm requires also the secret key.

Finally the signature is:

4SOiU5sf1xvglULlcuQkC6kLtBAinY7dpiNbd9/gaps=

3.2 Java implementation

```java
import javax.crypto.Mac;
import javax.crypto.spec.SecretKeySpec;
import java.io.UnsupportedEncodingException;
import java.security.InvalidKeyException;
import java.security.NoSuchAlgorithmException;
import java.util.Base64;
import java.util.TreeMap;

public class VadsSignatureExample {
    /**
     * Build signature (HMAC SHA-256 version) from provided parameters and secret key.
     * Parameters are provided as a TreeMap (with sorted keys).
     */
    public static String buildSignature(TreeMap<String, String> formParameters, String secretKey)
         throws NoSuchAlgorithmException, InvalidKeyException, UnsupportedEncodingException
    {
        // Build string to sign from parameters
        String message = String.join("+", formParameters.values());
        message += "+" + secretKey;
        // Sign
        return hmacSha256Base64(message, secretKey);
    }

    /**
     * Signature computation
     */
    public static String hmacSha256Base64(String message, String secretKey) throws
        NoSuchAlgorithmException, InvalidKeyException, UnsupportedEncodingException
   {
        // Prepare hmac sha256 cipher algorithm with provided secretKey
        Mac hmacSha256;
        try {
            hmacSha256 = Mac.getInstance("HmacSHA256");
        } catch (NoSuchAlgorithmException nsae) {
            hmacSha256 = Mac.getInstance("HMAC-SHA-256");
        }
        SecretKeySpec secretKeySpec = new SecretKeySpec(secretKey.getBytes("UTF-8"), "HmacSHA256");
        hmacSha256.init(secretKeySpec);
        // Build and return signature
        return Base64.getEncoder().encodeToString(hmacSha256.doFinal(message.getBytes("UTF-8")));
        }
    }
}
```

3.3 PHP implementation

```
/**
 * Function that computes the signature
 * $params : table containing the fields to send in the payment form.
 * $key : TEST or PRODUCTION key
 */

function getSignature ($params,$key)
{
    // Initialization of the variable that contains the string to encrypt
    $to sign = "";

    // sort fields alphabetically
    ksort($params);
    foreach($params as $name=>$value) {
        // Filter fields with vads  prefix
        if (substr($nom,0,5)=='vads ') {
            // String concatenation with separator "+"
            $to sign .= $value."+";
        }
    }

    // Concatenate the secret key
    $to sign .= $key;

    // HSH-MAC-SHA256 + Base64 encoding
    $signature = base64 encode(hash hmac('sha256',$to sign, $key, true));

    return $signature;
}
```