Lyra Payment Gateway

UPI Intent Integration

Document version 1.1

HISTORY OF THE DOCUMENT

Version	Author	Date	Comment
1.0	Lyra Network	18th December 2020	Initial release
1.1	Lyra Network	08th January 2021	Create charge resource – REST API Webhook signature
1.2	Lyra Network	15/01/2021	Remove null parameters from UPI intent keyId updated in webhook signature with test/prod indicator. Introduced signature content versioning: v1

TECHNICAL SUPPORT

For technical inquiries or support, you can reach us from Monday to Friday, 9am to 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. Create a charge resource

Before creating a UPI intent, the merchant has to call our REST API to create a charge resource. The charge resource has all the order details and customer details.

The charge uuid must be used as the transaction reference in the UPI intent

1.1 API

Endpoint: https://api.in.lyra.com/pg/rest/v1/charge

Method: POST

Content: JSON

The body shall be formatted in JSON as detailed below.

Parameter name	Description	Value
orderId	Order id, as provided by the merchant	e.g. TVH837dr28
orderInfo	Additional order information (free text)	e.g. Invoice #abc Shopping cart #123
currency	3-character currency code	INR
amount	Order amount, in paise	e.g. 25700 for 257.00 INR
customer.uid	Customer unique id, as provided by the merchant	e.g. v1VMfTEoZsgiBqoMtwAdfZhDX862
customer.name	Customer name	e.g. Kiran Jha
customer.emailId	Customer email id	e.g. kiran.jha23823@mailer.com
customer.phone	Customer phone no	e.g. 8291234567
customer.address	Customer address	
customer.city	Customer city	
customer.state	Customer state	
customer.zip	Customer zip code	
customer.country	Customer country	
webhook.url	URL of the merchant webhook. The webhook is called when the charge is paid or expired.	
maxAgeInHours	Charge expiry delay, in hours, between 1 and 999	e.g. 240 for 10 days

Example of charge creation request:

```
{
  "orderId": "Ha5FvH7J001",
  "orderInfo": "Bill Payment #123abc",
  "currency": "INR",
```

```
  "amount": 25700,
  "customer": {
     "uid":"v1VMfTEoZsgiBqoMtwAdfZhDX862",
     "name": "Khiran Jha",
     "emailId": "kiran.jha23823@mailer.com",
     "phone": "+918291234567"
  },
  "maxAgeInHours": 240,
   "webhook": {
     "url": "https://us-central1-sublyme-test.cloudfunctions.net/updateUPITransactionStatus"
  }
}
```

1.2 Charge resource

A charge resource object is returned as the response to the create charge request.

The same object is also returned in the notification webhook, see section 3.

The charge resource comes with a status and a unique identifier (uuid) which is generated by the platform.

The charge uuid must be passed in the UPI intent as the transaction reference.

Parameter name	Description	Value
uuid	Unique Charge ID	32 alphanumeric characters, e.g. 27019964b16b46d5b13795aaecff18ff
date	Charge creation date	e.g. 2021-01-08T10:10:21.329+00:00
expiryDate	Charge expiry date	e.g. 2021-01-09T10:10:21.329+00:00
status	Charge status	DUE / PAID / DROPPED
orderId	Order Id	as per merchant
orderInfo	Order additional info	as per merchant
currency	Currency	INR
amount	Amount raised by the merchant, in paise	e.g. 25700 for 257.00 INR
due	Amount charged to the customer, in paise	e.g. 25700 for 257.00 INR
paid	Amount paid by the customer, in paise	0 if charge status is DUE or DROPPED equal to the due amount if charge status is PAID
customer.uid	Customer unique ID	as per merchant
customer.name	Customer name	as per merchant
customer.emailId	Customer Email id	as per merchant
customer.phone	Customer Phone no	as per merchant
customer.address	Customer address	as per merchant
customer.city	Customer city	as per merchant
customer.zip	Customer zip code	as per merchant
customer.country	Customer country	as per merchant
attempts	Number of payment attempts	0 or 1
testMode	Transaction environment	true for Test transaction

		false for live transaction (production)
transactions	Array of payment transaction information	contains a single payment transaction

Example of charge resource :


```
{
   "uuid": "27019964b16b46d5b13795aaecff18ff",
   "date": "2021-01-08T10:10:21.329+00:00",
   "expiryDate": "2021-01-09T10:10:21.329+00:00",
   "status": "DUE",
   "orderId": "Ha5FvH7J001",
   "orderInfo": "Payment",
   "currency": "INR",
   "amount": 1000,
   "paid": 0,
   "due": 1000,
   "refunded": 0,
   "customer": {
      "uid": "v1VMfTEoZsgiBqoMtwAdfZhDX862",
      "name": "Rohit Tambe",
      "phone": "+919763951288",
      "email": "rohit.tambe@billcloud.in",
      "address": null,
      "city": null,
      "state": null,
      "zip": null,
      "country": null
   },
   "orderInfo": "Payment",
   "attempts": 0,
   "testMode": true,
   "dropReason": null,
   "paymentLink": "https://api.in.lyra.com/charge/27019964b16b46d5b13795aaecff18ff"
}
```


2. UPI Intent

The UPI Intent flow provides smooth checkout experience as it automatically launches a preferred UPI mobile app during payment.

2.1 Specification for creating a UPI intent

As per NPCI guidelines, below are the specifics to be followed by the merchant for Lyra to receive and process the notification from the bank.

Parameter name	Description	Value

pa	Payee VPA	Merchant vpa, **to be shared by Lyra**
pn	Payee Name	Merchant name
tr	Transaction ref id	**charg uuid** from the charge API - 32 alphanumeric characters
am	Transaction amount	Order amount, as per merchant
cu	Transaction currency	INR
mode	Transaction mode	01
orgid	Origination ID	000000
mid	Merchant ID	Merchant ID shared by Lyra

Example of UPI intent

upi://pay?pa=lyra.2345432@bankname&pn=billcloud&tr=27019964b16b46d5b13795aaecff18ff&am=1000&cu=INR&mode=01&orgid=000000&mid=P1700272

2.2 URL signature

For additional security the URL can be signed by the merchant as per NPCI specifications.

3. Webhook notification

The merchant registers a webhook url to receive notifications from Lyra when the UPI payment has been processed.
The webhook is also called when the charge expires.

The webhook url must be publicly accessible. It is passed on as a parameter in the charge creation request, see section 1.1.

3.1 Webhook content

The body of the webhook request contains the charge resource as detailed in section 1.2.

The charge status shall be either:

- PAID: the UPI payment was completed successfully. The **paid** amount shall be equal to the **due** amount.
- DROPPED: the charge has expired or was cancelled by the merchant. The **paid** amount shall be 0.

3.2 Webhook security

Since the webhook url is a public url it must be protected against malicious use or data tampering.

The request is signed following the *Digest Headers Drat* and the *HTTP Message Signature draft* from the IETF (Internet Engineering Task Force).

References:
HTTP Signature: https://tools.ietf.org/html/draft-ietf-httpbis-message-signatures-01
HTTP Digest Headers: https://tools.ietf.org/html/draft-ietf-httpbis-digest-headers-04
HTTP Semantics: https://tools.ietf.org/html/draft-ietf-httpbis-semantics-12

Basically a digest is computed from the request body and the digest is signed with the shop key. The digest and the signature are placed in HTTP headers.

e.g.

```
POST /
HTTP Headers
content-type: 'application/json'
digest: 'SHA-256=/U+c6wqyNUmaDzlT6MxMHDE+w1FRyiCAvAqsljnv8Jw='
signature: 'v1=:3/bA+uT86y1hQVI1beH6txZGIWrCBNTeOIwfU9aF1no=:'
signature-input: 'v1=(*created content-type digest); alg=hmac-sha256;
keyid="44247028.test"; created=1610717375'
HTTP Body
{"uuid":"7f4b634a78054183b91fb06ade5549cd","date":"Jan 15, 2021 6:59:34
PM","expiryDate":"Jan 16, 2021 6:59:34
PM","status":"PAID","orderId":"fv9sfjzw","currency":"INR","amount":836991,"paid":836991,"d
ue":836991,"refunded":0,"customer":{"uid":"customer1234","name":"Payzen
Customer","phone":"2554562523","email":"emailId@emailId.com"},"attempts":1,"testMode":true
}
```

The merchant must cross-check the digest and the signature to guarantee the integrity of the data.

Failure to check the signature will expose the merchant to data tampering and/or fraudulent use of the webhook.

3.3 Cross-check the request digest

1. Compute the SHA-256 digest from the request body
2. Convert the digest to BASE64 character string
3. Compare the result with the value of the **Digest** HTTP header. It should match.

Pseudo-code

```
val checksum = sha256(request.body)
request.headers.Digest = 'SHA-256=' + base64(checksum)
```

3.4Cross-check the request signature

1. Extract the **keyId** value from the **signature-input** HTTP Header. It indicates the shop id and the test environment which was used to sign the request, e.g. keyId="**44247028.test**" for test transaction on shop 44247028 and keyId="**44247028.prod**" for live transaction.
2. Extract the **created** value from the **signature-input** HTTP Header. It indicates the UNIX timestamp which was used to sign the request, e.g. created=**1610717375**
3. Extract the value of the **content-type** HTTP Header, e.g. **application/json**
4. Take the digest value computed in the previous step or from the **digest** HTTP header.
5. Concatenate the timestamp, the content type and the digest with the exact format detailed in the pseudo-code below
6. Sign the concatenated string with HMAC-SHA-256 algorithm and the shop key. Make sure to use the correct test or production shop key.
7. Convert the result to BASE64 character string
8. Compare the result with the value of the **signature** HTTP header. It should match.

Pseudo-code

```
val shopId = request.headers['signature-input'].keyId
val timestamp = request.headers['signature-input'].created
val contentType = request.headers['content-type']
val digest = request.headers['digest']

val toSign = '*created: ' + timestamp + '\n'
             + 'content-type: ' + contentType + '\n'
             + 'digest: ' + digest
val signature = hmacSha256(toSign, shop.key)

request.headers['Signature'] = 'v1=:' + base64(signature) + ':'
```