COLLECTING SOLUTION

Marketplace Web Service REST API

Implementation Guide

Document version 2.4.2

Contents

1. HISTORY OF THE DOCUMENT....4

2. MARKETPLACE GLOSSARY....7

3. PRESENTATION OF THE MARKETPLACE....8

Commission principle....8
Calculation of minimum commission....8
Submission methods....9

4. PRESENTATION OF THE WEB SERVICES....11

4.1. Main payment stages....11
4.2. Web service resources....11
4.3. Prerequisites....12

5. UNDERSTANDING THE MARKETPLACE PAYMENT FLOW....13

6. IDENTIFYING YOURSELF DURING DATA EXCHANGE....16

7. UNDERSTANDING MARKETPLACE DATA....17

8. UNDERSTANDING THE RETURN CODES OF THE HTTP STATUS SENT VIA WEB SERVICE....19

9. VIEWING THE SUB-MERCHANTS REGISTERED ON THE MARKETPLACE....20

10. USING WEBHOOKS....21

10.1. Defining the webhook address....21
10.2. Accessing the webhooks defined and available on your marketplace....22
10.3. Registering, modifying or deleting a webhook....22
10.3.1. Adding a webhook....22
10.3.2. Modifying a webhook....24
10.3.3. Deleting a webhook....24

11. MAKING A PAYMENT....25

11.1. Installment payment....29
11.1.1. Regular payments: MULTI....30
11.1.2. Custom schedule: MULTI_EXT....31
11.2. Payment with capture delay....32
11.3. Payment with token creation....33
11.4. Payment by token....34
11.5. Payment initiated by the Merchant....35
11.6. Manual validation payment....36
11.7. Payment by voucher....40
11.7.1. Prerequisites....40
11.7.2. Creating and modifying the order....41
11.7.3. Selecting MIDs upon order execution....42
11.7.4. After the payment....43
11.7.5. Modification and cancellation....44
11.8. Payment using a persistent link....45
11.8.1. Generating the link....45
11.8.2. Adjusting the expiration date....45

12. ANALYZING THE PAYMENT RESULT....46

13. TOKEN MANAGEMENT....48

13.1. Creating a token.. 48
13.2. Analyzing the result of a token request.. 50
13.3. Updating an alias.. 51
13.4. Retrieving token details.. 53

14. PROCESSING THE RETURN TO THE MARKETPLACE.. 54

15. UPDATING AN ORDER.. 55
15.1. Updating an order paid in installments.. 56

16. CANCELING AN ORDER.. 58

17. REFUNDING A PAYMENT.. 59
17.1. Creating a refund request.. 59
17.2. Following up the refund request.. 61
17.3. Modifying a refund request.. 62
17.4. Canceling a refund request.. 63

18. MANUALLY TRIGGERING ITEM PAYMENT.. 64

19. VIEWING MARKETPLACE CASHOUTS.. 65
19.1. Understanding cashouts.. 65
19.2. The cashout process.. 65
19.3. Identifying the cashout and the associated orders.. 66
19.4. List of cashouts.. 66
19.5. Cashout details.. 68

20. GENERATING A CLIENT VIA OPENAPI.. 69

21. DATA DICTIONARY.. 70
21.1. Address object.. 70
21.2. Alias object.. 70
21.3. Buyer object.. 71
21.4. Item object.. 71
21.5. Marketplace object.. 73
21.6. Order object.. 74
21.7. Order voucher object.. 75
21.8. Refund object.. 75
21.9. Seller object.. 77
21.10. Shipping object.. 78
21.11. Token object.. 79
21.12. Transaction object.. 80

22. OBTAINING HELP.. 84

1. HISTORY OF THE DOCUMENT

Version	Author	Date	Comment
2.4.2	Lyra Collect	11/25/2021	Update of the *Accessing the webhooks defined and available on your marketplace* chapter: correction of the request example
2.4.1	Lyra Collect	11/17/2021	Update of the *Shipping object* chapter: correction of PACKAGE_DELIVERY_COMPANY
2.4	Lyra Collect	10/6/2021	• Addition of the *Using webhooks* chapter. • Update of the *Understanding Marketplace data* chapter.
2.3	Lyra Collect	5/27/2021	• Addition of the *Payment by vouchers* chapter and its subchapters. • Removal of the *Payment with Electronic Meal Vouchers* chapter • Addition of the *Payment using a persistent link* chapter and its sub-chapters. • Update of the *Order object* chapter in the data dictionary. • Addition of the *Order voucher object* in the data dictionary.
2.2	Lyra Collect	4/13/2021	Clarifications added to the *Submission methods* chapter.
2.1	Lyra Collect	1/28/2021	• Update of Order object description. • Update of the *Processing the return to the Marketplace* chapter. • Addition of the mobile SDK support in the *Resources of the web service* chapter.
2.0	Lyra Collect	12/21/2020	• Addition of the *Understanding Marketplace data* chapter. • Addition of the *Understanding cashouts*chapter. • Addition of the *The cashout process*chapter. • Addition of the chapter *Identifying the cashout and the associated orders*. • Update of the *List of cashouts* chapter: clarification in "capture_label".
1.9	Lyra Collect	11/23/2020	• Update of the *Analyzing the payment result* chapter.
1.8	Lyra Collect	11/17/2020	• Addition of the *Generating a client via OpenAPI* chapter. • Addition of possible values for the **auto_code** attribute of the **Transaction** object. • Update of the *Payment with manual validation* chapter.
1.7	Lyra Collect	8/11/2020	• Addition of the Transaction object description. • Correction of the minimum commission formula. • Correction of the call URL in the refund example.
1.6	Lyra Collect	6/11/2020	Addition of the *Payment with Electronic Meal Vouchers* chapter.
1.5	Lyra Collect	5/28/2020	• Addition of order statuses allowing cancellation. • Update of order statuses allowing refunds. • Addition of the minimum commission calculation. • Addition of the *Payment initiated by the Merchant* chapter.
1.4	Lyra Collect	5/6/2020	• Addition of creating payments in installments.

Version	Author	Date	Comment
			• Addition of updating an order paid in installments. • Update of commission principle. • Update of the items resource. • Update of the *Making a payment* chapter. • Correction of the POST ORDER call URL in examples.
1.3	Lyra Collect	7/16/2019	Addition of the **Cashouts** resource. Update of the **Items**, **Refunds** and **Order** resources. **Order** object: addition of the **form_token** attribute. Addition of actions: • List the cashouts of a marketplace • Obtain cashout details • Modifying a refund request • Canceling a refund request • Unlock the payment of a previously locked item Updated chapters: • Following up the request • Updating an order • Presentation of the Web Services • Main stages of a payment • Understanding marketplace payment flow • Making a payment Addition of chapters: • Manual triggering of item payment • Viewing marketplace cashouts • Modifying a refund request • Canceling a refund request • Pre-authorized payments (manual validation)
1.2	Lyra Collect	1/22/2019	Addition of field definition for "**order**". Addition of the **commission principle**. Addition of **Tokens** and **Refunds** resources. Addition of **Tokens** and **Refunds** life cycle. Addition of the **Cancel order** action. Update of the **Making a payment** chapter: • Update of code samples. • Addition of the use of Tokens resource. Addition of chapters: • Analyzing the result of a token request. • Retrieving token details • Modifying an order • Payment with capture delay • Token object • Alias object • Refund object
1.1	Lyra Collect	7/12/2018	Details added to chapters: addition of names within objects. Removal of the transfers resource. Addition of chapters:

Version	Author	Date	Comment
			• Viewing the sub-merchants registered on the marketplace • Refunding a payment
1.0	Lyra Collect	3/12/2015	Initial version

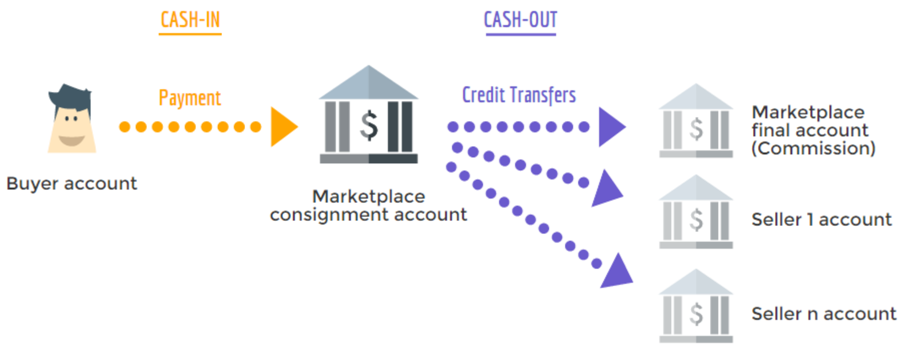
2. MARKETPLACE GLOSSARY

Vocabulary	Description
Cash-in	Payment made by the buyer to the Marketplace (CB, VISA, etc.).
Cash-out	Transfer made by the Marketplace to the Sellers.
Marketplace	A website offering items from different vendors/service providers.
Order	An order is composed of items, corresponds to an order of the marketplace. Each item is associated with a seller.
Seller	A seller on the Marketplace, also called the provider. Can be a company (holder of a business registration number) or an individual.

3. PRESENTATION OF THE MARKETPLACE

The payment gateway offers payment services for Marketplaces that sell products or services on the Internet on behalf of sellers with whom they have signed a commercial agreement to this effect.

The principle of bank flows:



Vocabulary	Description
Cash-in	Payment made by the buyer to the Marketplace (CB, VISA, etc.).
Cash-out	Transfer made by the Marketplace to the Sellers.

The buyers' payments are distributed between the payment accounts of the sellers. The payment gateway then triggers credit transfers to the bank accounts of Marketplace sellers.

Commission principle

Commission is the percentage of the order amount withdrawn by the Marketplace.

Note:

The part allocated to Lyra Collect is deducted from this commission. Therefore, the Marketplace commission should cover the costs of Lyra Collect. For this reason, the total amount of commissions included in the order is checked before the order is paid for.

Calculation of minimum commission

To avoid raising an error when checking the commission, you can first check that the commission amount sent to the marketplace API is greater than or equal to the minimum commission, the formula for which is given below:

Symbol	Definition	Example 1	Example 2
Com_{min}	Minimum commission		
a	***Coefficient*** of the minimum commission in proportion to the total amount of the order (commission included)	2%	1%
b	Fixed commission ***per transaction***	€0.50	€0.20
M	***Commission-free*** order amount	€100.00	€100.00

Symbol	Definition	Example 1	Example 2
n	Number of order transactions	1	3
vat	VAT rate	20%	20%

Note:

- *a*, *b* and *vat* are configured in the marketplace API. If you are not familiar with them, reach out to your sales contact.
- The marketplace that transmits the order to the marketplace API handles *M* and *n*.

The minimum commission is obtained using the following formula according to the order amount excluding commission (M):

$$Com_{min}(M) = \frac{aM + bn}{\frac{1}{1+tva} - a}$$

With the amounts provided in the example, expressed in cents, the minimum commission is:

Example 1:

$$Com_{min}(M) = \frac{10000 \times 0.02 + 50}{\frac{1}{1+0.2} - 0.02} = 307.37$$

Example 2:

$$Com_{min}(M) = \frac{10000 \times 0.01 + 3 \times 20}{\frac{1}{1+0.2} - 0.01} = 194.33$$

The transmitted commission amount should therefore be at least equal to €3.08 in example 1, and €1.95 in example 2.

Submission methods

There are two ways of specifying the commission amount:

1. **The commission is defined within the order**: the commission amount is indicated through a commission type item, marked with the attribute **is_commission=true**. The seller of this item must be a Marketplace seller, i.e. the one whose **is_marketplace_seller** attribute is set to "**true**". In this case, the amount of the commission is added to the amount of the other items.

   Example:

```
"items": [
{
 "seller": "4d20a9d4-0526-4474-b452-e936dc25418d",
 "reference": "sub_merchant_product",
 "description": "Product",
 "amount": 10000,
 "quantity": 1
},
{
 "seller": "72ccc2ff-b455-4653-847e-deb6fee99f8d",
 "reference": "marketplace_commission",
 "description": "Commission",
 "amount": 1000, "quantity": 1,
 "is_commission": true
}]
```

In this example, if the order currency is EUR and it contains 2 items, 1 item of €100 and 1 item of a €10 commission, then the total order amount is 100 + 10 = €110.

The seller will receive €100 and the marketplace will receive €10 (minus the part allocated to Lyra Collect).

2. **The commission is defined within an item:** the commission amount is specified by setting the **commission_amount** attribute of the item. In this case, the commission amount is deducted from the amount associated with the item and the **commission_amount** can thus be defined for the sub-merchants (and not for the marketplace operator).

   Example:

   ```
   {
    "seller": "4d20a9d4-0526-4474-b452-e936dc25418d",
    "reference": "abcdef",
    "description": "Restaurant",
    "amount": 10000,
    "quantity": 1,
    "commission_amount": 1000,
    "is_commission": false
   }
   ```

   In this example, if the order currency is EUR, the item is worth €100 and the **commission_amount** attribute is set to €10, the Marketplace will receive €10, and the item merchant will receive 100 - 10 = €90.

   This modality is useful for controlling the distribution of commissions between different items (and, therefore, between different sellers).

Notes:

- It is possible to combine the two modes, i.e. to define a **commission_amount** within one or more items, and add a commission type item. In this case, the commissions defined within an order are added to the commissions defined within an item.
- On the other hand, it does not make sense - and it is not possible - to define a **commission_amount** for a commission type item.
- The default value of the **is_commission** attribute is “**false**”. Thus, it is not necessary to indicate it for non-commission items.

An example of creating an order is provided in chapter *Making a payment* on page 25.

4. PRESENTATION OF THE WEB SERVICES

This document presents the Marketplace Web Services that allow to:

- Create an order
- Make the buyer pay on the Lyra Collect payment page *via* an embedded form
- View order details

The Marketplace Web Services have been developed in accordance with the REST protocol.

4.1. Main payment stages

1. If the merchant website has opted for and installed the embedded form, the buyer directly enters their payment details using the form. Otherwise, the **merchant website** redirects the **buyer** to the Lyra Collect payment page.

   The payment page displays the total amount to be settled.

2. Lyra Collect makes the **buyer** pay the total amount.

3. If the payment is successful, Lyra Collect creates credit transfers to the **Provider(s)** specified in the shopping cart.

4. Lyra Collect performs the capture of the transaction made by credit card and sends the transfer file to the **Bank**.

4.2. Web service resources

The Marketplace Web Services are available at the following address:

- **Test Mode** (integration phase): https://secure.lyra.com/marketplace-test/
- **Production**: https://secure.lyra.com/marketplace/

The resources of this API can be found via different HTTP methods:

Resources	Action	HTTP method	URI
Marketplaces	Retrieve the list of Marketplaces	GET	/marketplaces/
	Retrieve Marketplace details	GET	/marketplaces/{id_marketplace}
Sellers	Retrieve a provider	GET	/sellers/{id_seller}
	Retrieve the list of providers	GET	/marketplaces/{id_marketplace}
Orders	Create an order	POST	/orders?expand=items
	Modify an order	PUT	/orders/{id_order}
	Retrieve an order	GET	/orders/{id_order}
	Retrieve the list of Marketplace orders	GET	/orders
	Finalize an order and prepare the payment page	GET	/orders/{id_order}/execute
	Finalize an order and retrieve the token of the embedded form	GET	/orders/{id_order}/embedded-execute
	Finalize an order and retrieve the form token initialize the payment via the mobile SDK	GET	/orders/{id_order}/embedded-execute
	Cancel an order	DELETE	/orders/{id_order}
Items	Retrieve an element of an order	GET	/items/{id_item}

Resources	Action	HTTP method	URI
	Retrieve order details	GET	/orders/{id_order}/items
	Retrieve provider details	GET	/sellers/{id_seller}/items
	Create an element (item) inside an order	POST	/orders/{id_order}/items
	Unlock the payment of a previously locked item	POST	/items/{id_item}/activate
Refunds	Perform a refund	POST	/refunds
	Retrieve refund details	GET	/refunds/{id_refund}
	Retrieve the list of an order refunds	GET	/{id_order}/refunds
	Update a refund	PUT	/refunds/{id_refund}/
	Delete a refund	DELETE	/refunds/{id_refund}/
Tokens	Create/update a token	POST	/tokens
	Retrieve a simple token request	GET	/tokens/{id_token}
	Retrieve a token request linked to an order	GET	/tokens/{id_order}
Alias	Retrieve token details	GET	/marketplaces/{id_marketplace}/alias/{id_alias}
Cashouts	List the cashouts for the sellers of a marketplace	GET	/cashouts
	Retrieve cashout details	GET	/cashouts/{id_cashout}

4.3. Prerequisites

Contact Lyra Collect to enable access to the Marketplace and obtain the environment.

Prerequisites for the Marketplace

- Opt for the Marketplace offer.

  After you select this offer, the payment gateway will send you the data needed to access the Marketplace:

  - the **ID** of your Marketplace
  - a **login** and a **password** required for your identification

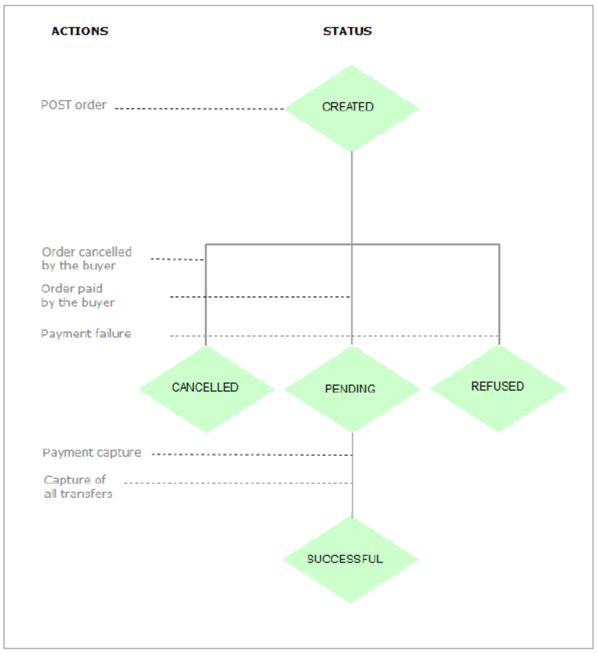
5. UNDERSTANDING THE MARKETPLACE PAYMENT FLOW

Below is the step-by-step process of a payment on the Marketplace:

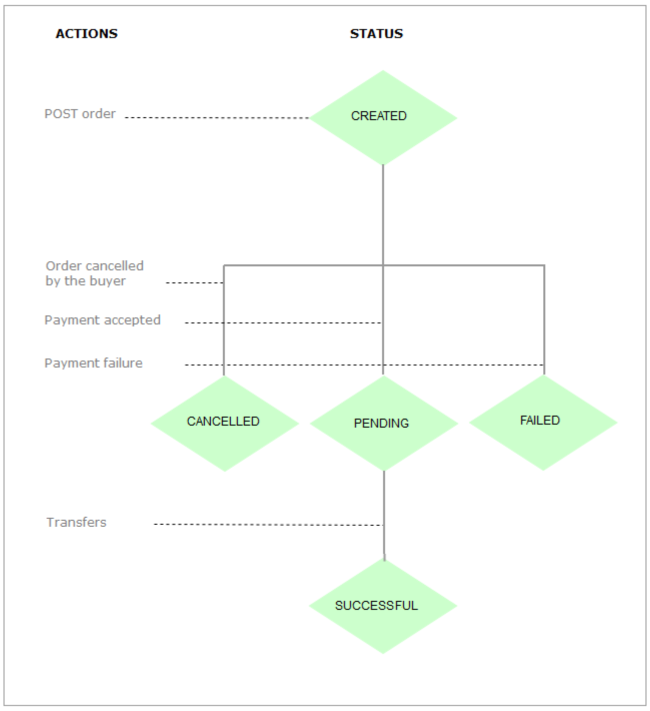
Stages	Actors	Actions
1	**Buyer**	Validates his/her cart on the Marketplace website.
2	**Marketplace**	Creates an Order via the REST API (POST ORDER) containing items.
3	**Marketplace**	Fixes the Order via the REST API (GET ORDER). Finalizes the Order and prepares the payment page or the token of the embedded form.
4	**Payment gateway**	Creates the payment context of the Order. If the embedded form is enabled, returns a token (form-token) to the Marketplace, which must be inserted in the embedded form. Otherwise, returns a URL to the Marketplace for redirecting the buyer to the payment page.
5	**Marketplace**	If the embedded form is enabled, builds the form on the merchant website. Otherwise, it redirects the buyer to the URL provided by the payment gateway and retrieved by the GET ORDER.
6	**Buyer**	Fills in the bank details (+ 3D Secure authentication on the payment pages).
7	**Payment gateway**	Validates and records the transaction.
8	**Payment gateway**	Updates the status of the Items and the Order.
9	**Payment gateway**	Sends a notification to the Marketplace (and, potentially, to the embedded form) indicating that the Order status has changed.
10	**Marketplace**	Calls the payment gateway to find out the Order status and updates the system (GET ORDER).
11	**Payment gateway**	Redirects the buyer to the Marketplace (return URL provided by the Marketplace = merchant website).
12	**Payment gateway**	Performs a capture in the bank of the credit card (or another) transaction. Creates transfers in the Seller's shop. Sends the order file of the transfer to the bank.
13	**Payment gateway**	Notifies the Marketplace about the modification of the Order and Items statuses.
14	**Marketplace**	Calls the payment gateway to find out the Order status and updates the system (GET ORDER).
15	**Marketplace**	Manually validates the order file of the transfer in the its bank's interface.
16	**Bank**	Executes the requested transfers.

The status of resources varies throughout the entire payment process.

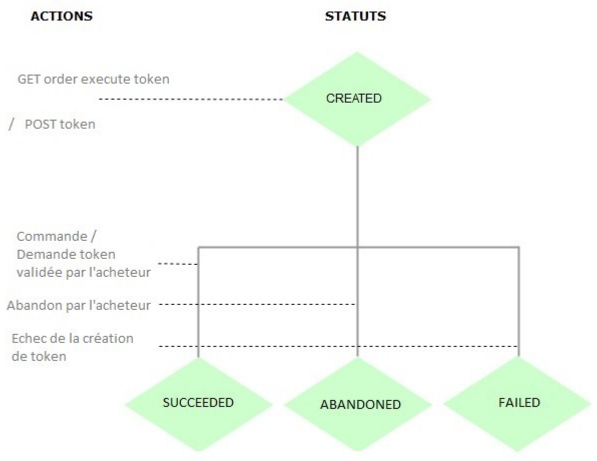
- **Order resource life cycle**



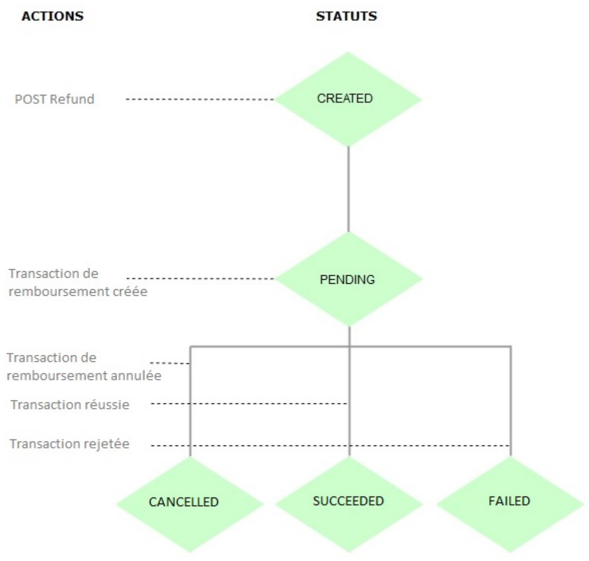
- **Item resource life cycle**



- **Tokens resource life cycle**



- **Refunds resource life cycle**



6. IDENTIFYING YOURSELF DURING DATA EXCHANGE

Identification is performed by means of an HTTP header.

The used method is **HTTP Basic Authentication**.

In each HTTP request, the header must contain the information allowing the Marketplace to authenticate itself when connecting to Marketplace Web Services.

Description of HTTP headers:

Headers	Description	Values to use
Accept	Determines the format of the contents that will be returned by the server. REST architecture that allows to perform data exchange in JSON format.	'Accept: application/json'
Authorization: Description	Identification token according to the "Basic Authentication over HTTPS" principle. Consists of an identifier and a password of the API user separated by a ':', both encoded in Base64.	'Authorization: Basic YWRtaW46YWRtaW4='
Content-type	Determines the format of the contents sent to the server.	'content-type: application/json'
Method	See the table in the **Web Service resources** chapter to see the methods to be used according to each resource.	**GET | POST | PUT | DELETE**

The steps for building headers are:

1. Use the **Basic Authentication** method.
2. Specify the used method in the **Authorization** header: **Basic** followed by the login and the password (encoded in Base64) separated by a ':'.
3. Encode the obtained result in Base64.
4. Add the chain into "Basic".

   Note:

   *Do not forget to add a space after **Basic**.*

- Example **cURL**:

```
$ curl 'https://secure.lyra.com/marketplace-test/123456/orders/' -H
'Authorization: Basic YWRtaW46YWRtaW4=' -H
'Content-Type: application/json' -H 'Accept: application/json' --data
'{}'
-i
```

- Example of a complete request in **Python**:

```
r = requests.post(
<target url>,
data=<json data>,
auth=(<api login>, <api password>),
headers={'content-type': 'application/json'},
verify=False
)
```

- Example of a request in **.NET**:

```
var myURL = "https://secure.lyra.com/marketplace-test/orders?expand=items"
HttpWebRequest myHttpWebRequest = (HttpWebRequest)WebRequest.Create(myURL);
myHttpWebRequest.ContentType = "application/json";
myHttpWebRequest.Accept = "application/json";
myHttpWebRequest.Method = "post";
string authInfo = userName + ":" + userPassword;
authInfo = Convert.ToBase64String(Encoding.Default.GetBytes(authInfo));
myHttpWebRequest.Headers["Authorization"] = "Basic " + authInfo;
```

7. UNDERSTANDING MARKETPLACE DATA

With the help of the login details and the unique identifier of your Marketplace (`uuid`) transmitted by our services, you can access your Marketplace data by calling the resource:

```
GET /marketplaces/{uuid}
```

Here is an example of a response for a Marketplace whose **uuid** is `2434c0a2-9d46-4e96-9553-1536c898625b`:

Request

```
GET https://secure.lyra.com/marketplace/marketplaces/2434c0a2-9d46-4e96-9553-1536c898625b
```

Response

```
{
  "href":"https://secure.lyra.com/marketplace/marketplaces/2434c0a2-9d46-4e96-9553-1536c898625
  "uuid":"2434c0a2-9d46-4e96-9553-1536c898625b",
  "created_at":"2017-03-13T14:58:40.801000Z",
  "updated_at":"2020-12-07T10:41:12.184969Z",
  "reference":"MKP000001",
  "description":"La maison du cheesecake",
  "billing_method":"CASHOUT",
  "bic":"",
  "iban":"",
  "vads_key":"12345678",
  "vads_cert":"123456789012345",
  "status":"ACTIVE",
  "links": {
   "sellers": {
     "href":"https://secure.lyra.com/marketplace/marketplaces/2434c0a2-9d46-4e96-9553-1536c89
   },
   "orders": {
     "href":"https://secure.lyra.com/marketplace/marketplaces/2434c0a2-9d46-4e96-9553-1536c898
   },
   "registrations":{
     "href":"http://secure.lyra.com/marketplace/marketplaces/2434c0a2-9d46-4e96-9553-1536c8986
   },
   "webhooks":{
     "href":"http://secure.lyra.com/marketplace/marketplaces/2434c0a2-9d46-4e96-9553-1536c8986
   }
  },
  "max_capture_delay":6,
  "tva_rate":"20.00",
  "currencies":[
   {
     "currency":"EUR",
     "commission_prorata":2.0,
     "commission_fix":20,
     "is_active":true
   },
   {
     "currency":"GBP",
     "commission_prorata":1.0,
     "commission_fix":60,
     "is_active":true
   }
  ],
  "vouchers": [
   {
     "contract_type":"CONECS"
   }
  ]
}
```

What do we know from this example?

1. First of all, that the Marketplace is active. That it corresponds to the shop ID “12345678” with the key “123456789012345”.

These two pieces of information are accessible via the Expert Back Office and are specified directly by the Lyra Collect services upon the Marketplace registration.

2. Its `billing_method` indicates that it is configured for a so-called “cashout” direct debit. As opposed to a “monthly” direct debit, the marketplace must respect a minimum commission amount for each order.
3. The list of available webhooks used by the marketplace can be found at the following address: "https://secure.lyra.com/marketplace/marketplaces/2434c0a2-9d46-4e96-9553-1536c898625b/webhooks"
4. The list of sellers can be found at the following address:

   ```
   "https://secure.lyra.com/marketplace/marketplaces/2434c0a2-9d46-4e96-9553-1536c898625b/
   sellers"
   ```
5. The list of orders can be found at the following address:

   ```
   "http://secure.lyra.com/marketplace/marketplaces/2434c0a2-9d46-4e96-9553-1536c898625b/
   orders"
   ```
6. There are two currencies enabled on the Marketplace, with separate minimum commission parameters:
   - **Euro**, with 2% of the pro rata of the total order amount and 10 cents per transaction;
   - **Pound sterling**, with 1% of the total order amount and 60 pence.
7. Finally, the marketplace can accept the registration of 'CONECS' acquirer MIDs for its sub-merchants, who can then offer *payment by meal voucher*.

8. UNDERSTANDING THE RETURN CODES OF THE HTTP STATUS SENT VIA WEB SERVICE

Error code	Description
200 - 20X	Successfully processed request
401	Unauthorized access (problem in the Authorization header)
403	Forbidden access (the API user does not have the permission to perform this request)
400	Bad input data (example: the format of some parameters is not respected)
404	The requested object cannot be found (a non-existent UUID was requested)
500	Internal server error

Here are some examples of path analysis in order to help you quickly resolve the occurred issues:

Error 400:

- **"?expand=items"** is absent at the end of the POST order request
- accents are not encoded in ANSI
- the item reference includes spaces

9. VIEWING THE SUB-MERCHANTS REGISTERED ON THE MARKETPLACE

To view the sub-merchants registered in the marketplace, you must launch a call in GET mode.

```
GET /marketplaces/{marketplace}/sellers
```

Replace {marketplace} with the uuid provided by Lyra Collect.

For example:

Request

```
GET https://secure.lyra.com/marketplace/marketplaces/30805a03-11ec-4447-93a5-243f39c89009/
sellers
```

Response

```
{
  "count": 1,
  "next": null,
  "previous": null,
  "results": [
    {
      "uuid": "eb94407d-c433-40ba-a5d3-d1baa0f7ed0a",
      "href": "https://secure.lyra.com/marketplace/sellers/eb94407d-c433-40ba-a5d3-
d1baa0f7ed0a",
      "created_at": "2018-04-13T12:57:59.943702Z",
      "updated_at": "2018-04-13T12:57:59.943713Z",
      "marketplace": "30805a03-11ec-4447-93a5-243f39c89009",
      "reference": "MKPINTEG_Seller",
      "description": "",
      "bic": "",
      "iban": "",
      "status": "ACTIVE",
      "links": {
        "items": {
          "href": "https://secure.lyra.com/marketplace/sellers/eb94407d-c433-40ba-a5d3-
d1baa0f7ed0a/items"
        },
        "transfers": {
          "href": "https://secure.lyra.com/marketplace/sellers/eb94407d-c433-40ba-a5d3-
d1baa0f7ed0a/transfers"
        }
      }
    }
  ]
}
```

10. USING WEBHOOKS

Webhooks allow you to be automatically notified when an object's status changes (order, registration, refund, etc.). By following its life cycle, you are able to react by triggering new events, calls, etc.

For example: a webhook is sent to you when the payment for an order is captured on the payment account from which you can, if this has been agreed upon in your business process, send information to your sub-merchant to validate the delivery.

For security reasons, the webhook is reduced to its simplest form that is unusable without an authenticated access to the API. This is a POST request with the following body:

```
{"order":"dd5e4c4c-2c07-4af8-ae30-15f4c6d5b5e5"}
```

When you receive this webhook, it is up to you to interrogate the object in question via the corresponding resource provided to you, and find out the new status.

Example:

```
GET https://secure.lyra.com/orders/dd5e4c4c-2c07-4af8-ae30-15f4c6d5b5e5)
```

Usually, the webhook reaches the marketplace server a few seconds after the status change. If an error occurs when it is sent (e.g. network unavailable), it can be resent up to two additional times, i.e. three times in total, at 15-minute intervals.

The webhook is not resent if there is an error once it is received (for example: if your server returns a 400, 404, 500 status, etc.). To avoid the most common errors, a dispatch and receipt check is carried out when the webhook is registered.

This section explains how to:

1. Define one (or more) access points
2. Access the available webhooks, and those currently defined on your marketplace
3. Register, modify and delete your webhooks.

10.1. Defining the webhook address

A webhook implies that you provide the API with an address for receiving information.

As mentioned earlier, webhooks are defined by object type. You can therefore define as many access points as there are objects to track.

These access points must be able to receive and process POST calls with the `Content-Type: application/json` header, and the following content:

```
{"<event_type>":"<uuid>"}
```

For example:

```
{"order":"dd5e4c4c-2c07-4af8-ae30-15f4c6d5b5e5"}
```

Currently, the objects concerned are:

- **order**

- **token**
- **registrations**
- **refund**

Each time a webhook is created or modified, a notification test is performed with an empty request body to which your server must respond with a 200 status (see chapter *Registering, modifying or deleting a webhook* on page 22).

To increase your process security, you can check that the webhook originates from the IP range **194.50.38.0/24**.

10.2. Accessing the webhooks defined and available on your marketplace

All your webhooks are available via the resource:

```
GET /marketplaces/{marketplace}/webhooks
```

The resource returns:

1. The list of webhooks currently defined on the marketplace
2. And the list of available but unused webhooks

Note: this list will get longer along with the development of the Marketplace API

Example:

Request

```
GET https://secure.lyra.com/marketplace/marketplaces/6f6b04c2-0e99-4f8d-b710-8856f5654bb8/
webhooks
```

Response

```
[ { "event_type":"order", "target":"https://mymarketplace.com/mkp/webhooks/order.php" },
 { "event_type":"token", "target":"https://mymarketplace.com/mkp/webhooks/order.php" },
 { "event_type":"refund", "target":null }, { "event_type":"registration", "target":null } ]
```

This response indicates that only one webhook address is used for two objects, `order` and `token`, but that the webhooks for the `refund` and `registration` objects are not specified, and are therefore inactive.

10.3. Registering, modifying or deleting a webhook

10.3.1. Adding a webhook

In order to add a webhook, first check that the webhook address is public and returns a 200 status when an empty request is received, then execute a request

```
POST /marketplaces/{marketplace}/webhooks
```

with the following body:

```
{
 "event_type":"<event_type">,
 "target":"<webhook_url>"
}
```

For example:

Request

```
POST https://secure.lyra.com/marketplaces/6f6b04c2-0e99-4f8d-b710-8856f5654bb8/webhooks
```

Body

```
{
 "event_type":"registration",
 "target":"https://mymarketplace.com/mkp/webhooks/sellers.php"
}
```

In case of an error during the test, the creation (or modification) request will return a 400 error.

Example if you have declared a non-existent page as webhook:

```
{"error":"Url https://mymarketplace.com/mkp/webhooks/sellers.php returned a status 404 instead
 of 200}
```

In case of success, the server returns a 200 code and the corresponding object.

10.3.2. Modifying a webhook

The same principles apply to the modification of the webhook, except that a `PUT` request must be made instead of `POST`.

```
PUT /marketplaces/{marketplace}/webhooks
```

Example for assigning a new address to the existing webhook on the `token` object:

Request

```
PUT https://secure.lyra.com/marketplaces/6f6b04c2-0e99-4f8d-b710-8856f5654bb8/webhooks
```

Body

```
{
 "event_type":"token",
 "target":"https://mymarketplace.com/mkp/webhooks/token.php"
}
```

10.3.3. Deleting a webhook

To delete a webhook, simply send a request

```
DELETE /marketplaces/{marketplace}/webhooks/{event_type}
```

For example to delete the webhook of the `registration` object:

```
DELETE https://secure.lyra.com/marketplaces/6f6b04c2-0e99-4f8d-b710-8856f5654bb8/webhooks/
registration
```

In case of success, the server returns a 204 status.

11. MAKING A PAYMENT

In order to make a payment, follow the steps below:

1. Create an order by using the POST method on the ORDER resource.
2. You can update the order, for example to update the shopping cart or the commission fee amounts.
3. Fix the order to make the payment using the GET method:
   - Either on the URL of the **execute** attribute provided in the ORDER links, in this case the response contains the redirection URL (**payment_url** attribute).
   - Or on the URL of the **execute-embedded** attribute for initializing payment via embedded form, in this case the response contains a formToken (**form_token** attribute).
4. Redirect the Buyer to the payment page URL or display the embedded form using the form token.

Step 1

In order to declare items in your order, you **must** mention the **?expand=items** argument in the URL (/marketplace/orders?expand=items).

Otherwise, your items will not be taken into account (the list of items returned by the server is empty) and the order will be created without items. When the order is executed (step 3), you encounter the error 400 {"amount":"The sum of item amounts cannot be zero or negative."}.

Example of a POST ORDER call:

```
POST https://secure.lyra.com/marketplace/orders?expand=items
```

```
{
  "marketplace": "ebfb36ab-2d30-4326-adb9-e16b0c9a89f3",
  "reference": "MKP BURO example",
  "description": "Office supplies order",
  "currency": "EUR",
  "url_return": "https://URLreturn.com",
"items": [
    {
      "seller": "d0f80202-0676-4d8d-9247-f455f30aec1b",
      "reference": "commburo",
      "description": "commission BURO",
      "amount": 150,
      "is_commission": true
    },
    {
      "seller": "5d0ef88c-3345-4b33-948e-80e23d553b73",
      "reference": "STB1",
      "description": "blue pens
      "amount": 1000,
      "is_commission": false
    }
  ],
  "buyer": {
    "type": "PRIVATE",
    "email": "john.smith@buyer.com",
    "phone_number": "33 (0)1 11 22 33 44",
    "reference": "ZhIsYM"
  },
  "shipping": {
    "shipping_method": "RELAY_POINT"
  }
}
```

Example of a POST ORDER response:

```
{
  "uuid": "1a2b51f8-9d62-408f-ab39-aa8e28ab15f0",
  "href": "https://secure.lyra.com/marketplace/orders/1a2b51f8-9d62-408f-ab39-aa8e28ab15f0",
"created_at": "2019-01-21T16:46:46.517000Z",
  "updated_at": "2019-01-21T16:46:46.517000Z",
  "marketplace": "ebfb36ab-2d30-4326-adb9-e16b0c9a89f3",
  "reference": "MKP BURO example",
  "description": "Office supplies order",
  "alias": null,
  "buyer": [
    {
      "reference": "ZhIsYM",
      "title": null,
      "type": "PRIVATE",
      "first_name": null,
      "last_name": null,
      "legal_name": null,
      "phone_number": "33 (0)1 11 22 33 44",
      "email": "john.smith@buyer.com",
      "address": null
    }
  ],
  "shipping": [
    {
      "shipping_method": "RELAY_POINT",
      "delivery_company_name": null,
      "shipping_speed": null,
      "shipping_delay": null,
      "type": null,
      "first_name": null,
      "last_name": null,
      "legal_name": null,
      "phone_number": null,
      "address": null
    }
  ],
  "amount": null,
  "initial_amount": null,
  "currency": "EUR",
  "status": "CREATED",
  "webhook_result": null,
  "url_return": "https://URLreturn.com",
"items": [
    {
      "uuid": "d33ba1ab-793e-4d3d-943f-72b14a1e885b",
      "href": "https://secure.lyra.com/marketplace/items/d33ba1ab-793e-4d3d-943f-72b14a1e885b",
"created_at": "2019-01-21T16:46:46.510000Z",
      "updated_at": "2019-01-21T16:46:46.573000Z",
      "seller": "5d0ef88c-3345-4b33-948e-80e23d553b73",
      "order": "1a2b51f8-9d62-408f-ab39-aa8e28ab15f0",
      "reference": "STB1",
      "description": "Blue pens”,
      "type": "ENTERTAINMENT",
      "amount": 1000,
      "quantity": 1,
      "transfer": null,
      "status": "CREATED",
      "links": null,
      "is_commission": false
    },
    {
      "uuid": "20034374-b425-4beb-91a8-a660ca5a3f9b",
      "href": "https://secure.lyra.com/marketplace/items/20034374-b425-4beb-91a8-a660ca5a3f9b",
"created_at": "2019-01-21T16:46:46.502000Z",
      "updated_at": "2019-01-21T16:46:46.551000Z",
      "seller": "d0f80202-0676-4d8d-9247-f455f30aec1b",
      "order": "1a2b51f8-9d62-408f-ab39-aa8e28ab15f0",
      "reference": "commburo",
      "description": "commission BURO",
      "type": "ENTERTAINMENT",
      "amount": 150,
      "quantity": 1,
      "transfer": null,
      "status": "CREATED",
      "links": null,
      "is_commission": true
    }
  ],
  "links": {
    "items": {
    "href": "https://secure.lyra.com/marketplace/orders/1a2b51f8-9d62-408f-ab39-aa8e28ab15f0/
items"
    },
    "refunds": {
```

```
    "href": "https://secure.lyra.com/marketplace/orders/1a2b51f8-9d62-408f-ab39-aa8e28ab15f0/
refunds"
    },
    "execute": {
    "href": "https://secure.lyra.com/marketplace/orders/1a2b51f8-9d62-408f-ab39-aa8e28ab15f0/
execute"
    },
    "execute-embedded": {
    "href": "https://secure.lyra.com/marketplace/orders/1a2b51f8-9d62-408f-ab39-aa8e28ab15f0/
execute-embedded"
    }
  },
  "vads_transaction_id": 0,
  "vads_transaction_date": null,
  "expected_capture_date": null,
  "capture_delay": null
}
```

Step 2

An order can be modified multiple times by a **PUT ORDER** query (/marketplace/orders/<order_uuid>?expand=items) using the **order_uuid** returned in step 1.

The **buyer** and **shipping** attributes must be specified.

The item list is updated with each PUT. If you wish to add another item, you must transmit it with the already created items. Otherwise, the list will only contain one item.

On the other hand, in order to remove an item, you need to return a complete list of items without the item that you wish to remove.

Example of a PUT ORDER call for adding an item to the cart:

```
PUT https://secure.lyra.com/marketplace/orders/515abac9-6cb2-4e21-8a25-b08d7e41e43c?
expand=items
```

```
{
  "marketplace": "ebfb36ab-2d30-4326-adb9-e16b0c9a89f3",
  "reference": "MKP BURO example",
  "description": "Office supplies order",
  "currency": "EUR",
  "url_return": "https://URLreturn.com",
"items": [
    {
      "seller": "d0f80202-0676-4d8d-9247-f455f30aec1b",
      "reference": "commburo",
      "description": "commission BURO",
      "amount": 270,
      "is_commission": true
    },
    {
      "seller": "5d0ef88c-3345-4b33-948e-80e23d553b73",
      "reference": "STB1",
      "description": "blue pens
      "amount": 1000,
      "is_commission": false
    },
    {
    "seller": "5d0ef88c-3345-4b33-948e-80e23d553b73",
    "reference": "RP3",
    "description": "Red pens",
    "amount": 1000,
    "is_commission": false
    },
  ],
  "buyer": {
    "type": "PRIVATE",
    "email": "john.smith@buyer.com",
    "phone_number": "+44 (0)1 11 22 33 44",
    "reference": "ZhIsYM"
  },
  "shipping": {
    "shipping_method": "RELAY_POINT"
  }
}
```

Step 3

With the exception of manually validated payments (see chapter *Manual validation payment* on page 36), *before* making the execution request and freezing the order, you must make sure that the commission amount is higher than the required minimum. If this is not the case, the execution request will return a validation error 400 with the message {"commission": "Insufficient amount of commission."}.

You can find information on the minimum commission in the Commission principle section of the chapter *Presentation of the Marketplace* on page 8.

Example of a GET call via redirection for freezing the order:

```
GET https://secure.lyra.com/marketplace/orders/515abac9-6cb2-4e21-8a25-b08d7e41e43c/execute
```

Example of a GET response:

```
{
 'payment_url': 'https://secure.lyra.com/vads-payment/
exec.refresh.a;jsessionid=HqKAcpvcgZQA29qCZDjXw4kS.marketplacevad01?
cacheId=450272311503195000050'
}
```

Example of a GET call via the embedded form for freezing the order:

```
GET https://secure.lyra.com/marketplace/orders/515abac9-6cb2-4e21-8a25b08d7e41e43c/execute-
embedded
```

Example of a GET response:

```
{"form_token":"fa0yfV2FQuR3aak1SwsgcuZg195eyJhbW91bnQiOjEwOTAwLCJjdXJyZW5jeSI6IkVVUiIsIm
       1vZGUiOiJURVNUIiwidmVyc2lvbiI6Mywib3JkZXJJZCI6IjVjNDk1N2E5LWNkOGEtNGJhZC1hMGNiLTgxMT
       ZiMWE3ZDdmNiIsInNob3BOYW1lIjoiTHlyYSBTTVMiLCJicmFuZFByaW9yaXR5IjpbIkJBTkNPTlRBQ1QiL
       CJDQiIsIkUtQ0FSVEVCTEVVRSIsIk1BU1RFUkNBUkQiLCVGQVVMVCJ9fX0a702"}
```

Note: the size of the form_token can be up to 8KB.

Step 4

In case of payment with redirection, simply redirect the buyer to the provided link. The same address can be used for an iframe display.

In case of an embedded payment form, the **form_token** must be inserted in the form (**kr-form-token** attribute).

Example:

```
<body>
 <!-- payment form -->
 <div class="kr-embedded"
 kr-form-token="fa0yfV2FQuR3aak1SwsgcuZg195eyJhbW91bnQiOjEwOTAwLCJjdXJyZW5jeSI6IkVVUiIsIm
   1vZGUiOiJURVNUIiwidmVyc2lvbiI6Mywib3JkZXJJZCI6IjVjNDk1N2E5LWNkOGEtNGJhZC1hMGNiLTgxMT
   ZiMWE3ZDdmNiIsInNob3BOYW1lIjoiTHlyYSBTTVMiLCJicmFuZFByaW9yaXR5IjpbIkJBTkNPTlRBQ1QiL
   CJDQiIsIkUtQ0FSVEVCTEVVRSIsIk1BU1RFUkNBUkQiLCVGQVVMVCJ9fX0a702">
  <!-- payment form fields -->
  <div class="kr-pan"></div>
  <div class="kr-expiry"></div>
  <div class="kr-security-code"></div>

  <!-- payment form submit button -->
  <button class="kr-payment-button"></button>
  <!-- error zone -->
  <div class="kr-form-error"></div>
 </div>
</body>
```

11.1. Installment payment

If the Buyer wishes to extend the payment over a period of time, he or she can make a payment in installments.

During a payment in installments, all the transactions are created on the days of payment.

Only the first installment can be guaranteed to the Merchant on the condition that their requested capture date is set before the authorization expiry date, depending on the payment method.

At the time of the payment, the validity of the payment method is checked throughout the payment schedule.

This payment mode is not supported by the embedded form.

The list of transactions can be seen at the following address:

```
GET /orders/{order}/transactions
```

It also appears on the list of links in response to a GET ORDER, POST ORDER, or PUT ORDER.

You should note that transactions are generated after the order has been paid. If the link is visited before, the list of transactions is empty.

In order to initialize a payment in installments, simply add the **payment_config** attribute in the POST ORDER resource when creating the order.

Here is an example of how to use the **payment_config** attribute:

```
{
 "marketplace": "9537e049-8862-400a-ae8d-da2ec9ca6051",
 "payment_config" : "MULTI:first=30000;count=3;period=30"
 "reference": "order00053",
 "description": Order",
 "currency": "EUR",
 "url_return": "http://www.my-website.com",
 "language":"en"
}
```

Notes:

- By default, the payment is mono-transactional. If the **payment_config** attribute is not specified when the order is created, it will take the **SINGLE** value;
- Once the order is fixed, it is not possible to change the **payment_config** value. However, some parameters of pending transactions can still be changed (see infra);
- Payment in installments is currently not compatible with cascading payment (and is therefore not compatible with the Electronic Meal Voucher system).

Two different payments in installments can be defined, depending on whether the Marketplace wishes to automatically create a regular schedule, or define the dates and amounts at its discretion.

11.1.1. Regular payments: MULTI

In case of a regular payment schedule, the first payment is made on the day of order creation, and its amount must be specified. The amounts and dates of the following payments are calculated according to the remaining amount and the number and frequency of the installments.

Syntax: the value of **payment_config** must be prefixed with the "**MULTI:**" string followed by key=value pairs separated by ";":

- **first=**: amount of the first installment (expressed in the smallest currency unit)
- **count=**: total number of installments
- **period=**: interval in days between 2 installments

Example:

The amount of the last transfer is calculated according to the amount remaining to be paid.

If a round-off is necessary, it is applied.

For example, in case of an order of €1,000 placed on January 15, the following **payment_config**:

```
{
 ...
 "amount": "100000",
 "payment_config": "MULTI:first=30000;count=4;period=30",
 ...
}
```

corresponds to a payment in 4 installments, including:

- a first transfer of €300 on January 15
- a second transfer of (1000 - 300) / 3 = €233.33 on February 14
- a third transfer of €233.33 on March 16
- and the last transfer of €233.34 on April 15

11.1.2. Custom schedule: MULTI_EXT

In case of a custom schedule, all transaction amounts and dates are specified by the marketplace. This makes it possible to define variable amounts, and the first transaction can be subsequent to the day of the order.

However, certain constraints must be respected:

- The date of the first installment payment shall not be earlier than the day of payment.
- It is only possible to record one transaction per day.
- The delay between the payment date and the last transaction may not exceed 365 days.

Syntax: Syntax: the value of **payment_config** must be prefixed with the "**MULTI_EXT:**" string, followed by the list of installments in date=amount format separated by ";".

Dates must be expressed in the YYYYMMDD format.

The amounts must be expressed in the smallest currency unit.

To use the previous example, but this time for settling order installments on the 15th of each month, the **payment_config** value should look like this:

```
{
...
"amount": "100000",
"payment_config": "MULTI_EXT:20200115=30000;20200215=23333;20200315=23333;20200415=23334",
...
}
```

Note: Using the MULTI_EXT value requires a subscription to the **Advanced installment payment** option.

11.2. Payment with capture delay

A delay for capturing the transaction can be defined upon order creation. To do this, populate the **capture_delay** attribute with the desired number of days for the delay.

This value must be between 0 and 6:

Example:

Request

```
POST https://secure.lyra.com/marketplace/orders?expand=items
```

Body

```
{
  "marketplace": "ebfb36ab-2d30-4326-adb9-e16b0c9a89f3",
  "reference": "MKP BURO example",
  "description": "Office supplies order",
  "currency": "EUR",
  "capture_delay": "3",
  ...
}
```

The **capture_delay** attribute cannot be used for changing the capture date of the transaction associated with an already executed order (PENDING status). You must pass “**expected_capture_date**” with the desired capture date, between the current date (D) and D+6. The format of the date is “YYYYMMDDHHMMSS”.

Example:

Request

```
PUT https://secure.lyra.com/marketplace/orders/515abac9-6cb2-4e21-8a25-b08d7e41e43c
```

Body

```
{
  "marketplace": "ebfb36ab-2d30-4326-adb9-e16b0c9a89f3",
  "reference": "Exemple MKP BURO",
  "description": "Commande fournitures",
  "currency": "EUR",
  "expected_capture_date": "20190425193000",
  ...
}
```

11.3. Payment with token creation

When creating an order, you can request to record the payment details as a token that can be reused.

For this, use the "**execute/token**" (or "**execute-embedded/token**") command instead of "execute" when validating the order.

Example of a GET call to fix the order:

```
GET https://secure.lyra.com/marketplace/orders/515abac9-6cb2-4e21-8a25-b08d7e41e43c/execute/
token
```

When the order is executed, a "token" object is created. **Its UUID is the same as for the Order**.

It is made up of two elements:

- A “buyer” object, as defined in the order.
- A “token” attribute, where the payment method data will be stored.

  See chapter Token management on page 48 for more information on tokens and aliases.

Then, redirect the Buyer to the URL transmitted in **payment_url** to proceed to payment with payment method recording.

After the payment, a GET Order will not return the created token.

In order to obtain the token, make a GET token call with the UUID of the Order.

Go to chapter *Analyzing the result of a token request* on page 50 to see the result of Get token.

11.4. Payment by token

To use a token referenced in an order, add the **alias** attribute when creating the order.

Example:

Request

```
POST https://secure.lyra.com/marketplace/orders?expand=items
```

Body

```
{
  "marketplace": "ebfb36ab-2d30-4326-adb9-e16b0c9a89f3",
  "reference": "Exemple MKP BURO",
  "description": "Commande fournitures",
  "currency": "EUR",
  "alias": "1144951ea7ab42989c97159b3dfc0382",
  ...
}
```

By calling “**execute**” on this order, the payment will be made via the payment page with the token that has “1144951ea7ab42989c97159b3dfc0382” as an alias.

By calling “**execute-embedded**” on this order, the payment will be made via the embedded form with the token that has “1144951ea7ab42989c97159b3dfc0382” as an alias.

By calling “**execute/token**”, the token will also be updated via the payment page with the information contained in the order.

By calling “**execute-embedded/token**”, the token will also be updated via the embedded form with the information contained in the order.

11.5. Payment initiated by the Merchant

The use of an alias when creating an order allows to execute payments initiated by the Merchant ("Merchant Initiated Transaction"): the payment is made *without* interaction with the cardholder.

This functionality can be useful in the case of recurring payments.

For this, create a payment by token (see chapter *Payment by token* on page 34), then use the **execute-mit** link returned by the server.

For example:

Request

```
POST https://secure.lyra.com/marketplace/orders?expand=items
```

Body

```
{
  "marketplace": "ebfb36ab-2d30-4326-adb9-e16b0c9a89f3",
  "reference": "MKP BURO example",
  "description": "Office supplies order",
  "currency": "EUR",
  "alias": "1144951ea7ab42989c97159b3dfc0382",
  ...
}
```

In the sever response, the **links** attribute contains the **execute-mit** link to be called.

```
{...
 "links": {
  "execute-mit": {
  "href": "http://mymarketplace.com/marketplace/orders/5a439e70-4ccc-4d86-bf30-223552e2c74f/
execute-mit"
 },
…
}
```

11.6. Manual validation payment

By default, orders are validated automatically during step 2 of the payment (during GET <url>/execute).

It is also possible to dissociate the steps of payment authorization from those of a transaction capture.

This allows to make authorizations while the final transfer of cart items between the seller amount and the Marketplace fee amount is not yet known.

Step 1

In order to enable this feature, all you need to do is add the **awaiting_validation** attribute when creating the order and assigning it the **true** value.

Example of a POST ORDER creation call in manual validation mode:

```
POST https://secure.lyra.com/marketplace/orders?expand=items
```

```
{
  "marketplace": "ebfb36ab-2d30-4326-adb9-e16b0c9a89f3",
  "reference": "MKP BURO example",
  "description": "Office supplies order",
  "currency": "EUR",
  "url_return": "https://URLreturn.com",
  "awaiting_validation" : true,
  "items": [
    {
      "seller": "5d0ef88c-3345-4b33-948e-80e23d553b73",
      "reference": "STB1",
      "description": "blue pens
      "amount": 1150,
      "is_commission": false
    }
  ],
  "buyer":
    {
      "type": "PRIVATE",
      "email": "john.smith@buyer.com",
      "phone_number": "33 (0)1 11 22 33 44",
      "reference": "ZhIsYM"
    },
  "shipping":
    {
      "shipping_method": "RELAY_POINT"
    }
}
```

In this example, the amount of the item (buyer amount) is known, but not the percentage of the sub-merchant and the operator fee (Marketplace).

Note: Please note that operator fees can already be specified in this step. If needed, they can be updated in step 4.

Step 2

Step 2 is executed in the same way as in case of standard payment and the redirection URL for proceeding to the payment is returned.

Example of a GET call to fix the order:

```
GET https://secure.lyra.com/marketplaces/marketplace/
orders/7fac13b0-7ab9-4382-9073-11ddb38d4427/execute
```

Since the order is waiting for validation, the verification steps of the minimum fee amount are not executed.

Step 3

Once the payment has been successfully made, the order status changes to PENDING and the status of the corresponding transactions changes to **TO_VALIDATE**.

Step 4

This optional step allows to the amount of the operator fees if they were not provided in step 1.

It is also possible to add:

- a fee item
- and/or fees for items (see corresponding paragraph).

One or several items can also be deleted (standard function).

The total amount of the order can be decreased but not increased.

On the other hand, the **awaiting_validation** attribute cannot be modified.

Step 5

The process is finalized with the validation of the order. During this step one can make sure that the amount of the fee is sufficient.

In order to validate an order, make a POST to the URL of the order, **validate** attribute.

<u>Example of a POST ORDER call for manually validating it:</u>

```
POST https://secure.lyra.com/marketplace/orders/7fac13b0-7ab9-4382-9073-11ddb38d4427/validate
```

Example of the validate POST ORDER response:

The order details are returned. The **awaiting_validation** attribute of the order then takes the **false** value and the status of the corresponding transactions changes from **TO_VALIDATE** to **PENDING**.

```
{
  "uuid": "7fac13b0-7ab9-4382-9073-11ddb38d4427",
  "href": "https://secure.lyra.com/marketplace/orders/7fac13b0-7ab9-4382-9073-11ddb38d4427",
  "created_at": "2019-05-27T09:30:24.434556Z",
  "updated_at": "2019-05-27T09:36:06.500936Z",
  "marketplace": "ebfb36ab-2d30-4326-adb9-e16b0c9a89f3",
  "reference": "MKP BURO 2 Example",
  "description": "Order of supplies",
  "alias": null,
  "awaiting_validation": false,
  "buyer": [
    {
      "reference": "ZhIsYM",
      "title": null,
      "type": "PRIVATE",
      "first_name": null,
      "last_name": null,
      "legal_name": null,
      "phone_number": "33 (0)1 11 22 33 44",
      "email": "john.mith@buyer.com",
      "address": null
    }
  ],
  "shipping": [
    {
      "shipping_method": "RELAY_POINT",
      "delivery_company_name": null,
      "shipping_speed": null,
      "shipping_delay": null,
      "type": null,
      "first_name": null,
      "last_name": null,
      "legal_name": null,
      "phone_number": null,
      "address":
        {
          "street_number": "37",
          "street": "rue Marcel Philippe",
          "district": "Wallis-et-Futuna",
          "zipcode": "59259",
          "city": "Dupont",
          "state": "Meurthe-et-Moselle",
          "country": "FR"
        }
    }
  ],
  "payment_config": "SINGLE",
  "amount": 1150,
  "initial_amount": 1150,
  "currency": "EUR",
  "status": "PENDING",
  "webhook_result": 200,
  "url_return": "https://URLreturn.com",
  "links":
    {
      "items":
        {
          "href": "https://secure.lyra.com/marketplace/
orders/7fac13b0-7ab9-4382-9073-11ddb38d4427/items"
        },
      "transactions":
        {
          "href": "https://secure.lyra.com/marketplace/
orders/7fac13b0-7ab9-4382-9073-11ddb38d4427/transactions"
        },
      "refunds":
        {
          "href": "https://secure.lyra.com/marketplace/
orders/7fac13b0-7ab9-4382-9073-11ddb38d4427/refunds"
        }
     },
  "vads_transaction_id": 600001,
  "vads_transaction_date": "20190527093234",
  "expected_capture_date": "20190530093235",
  "capture_delay": 3
}
```

Important: The order can be validated as long as the expiration date of the authorization request has not passed. If this date has passed, the transaction takes the final **EXPIRED** status, and the order status changes to **FAILED**.

11.7. Payment by voucher

The Marketplace API allows partial or full payment of the shopping cart items using vouchers such as Electronic Meal Vouchers (TRD), holiday vouchers (chèques vacances), etc.

The currently enabled payment methods are:

Customary name	Acquirer contract
Titres-Restaurant	CONECS
Chèque-Vacances Connect	CVCONNECT

11.7.1. Prerequisites

1. **At the marketplace level**

   The marketplace must be eligible for the specified voucher MID(s).

   To make sure, make a **GET** request on the marketplace resource with the "*vouchers*" attribute:

   ```
   GET /marketplaces/{uuid}
   ```

   For example, for a marketplace eligible for **CONECS** and **CVCONNECT** MIDs, the response will contain the following object list:

   ```
   ...
     "vouchers":[
    {
      "contract_type":"CONECS"
    },
    {
      "contract_type":"CVCONNECT"
    }
     ]
   ...
   ```

   These parameters are specified by the Lyra Collect services. You can contact tech support if you notice a deviation from the expected configuration.

2. **At the sub-merchants' level**

   The MID must be specified at the sub-merchant's level.

   For example, for a sub-merchant whose only active MID is **CONECS**:

   ```
   GET /sellers/{uuid}
   ```

   Example:

   ```
   ...
     "vouchers":[
    {
      "contract_type":"CONECS"
      "contract_number":"1999011"
    }
     ]
   ...
   ```

   To create or activate a MID at the sub-merchant level via the onboarding API, see the *corresponding documentation*.

11.7.2. Creating and modifying the order

When an item is payable by voucher, you must indicate it by sending the list of MIDs concerned at the level of the “items” object, as follows:

```
  "vouchers":[
 {
   "contract_type":"CVCONNECT"
 }
  ]
```

For **CONECS** MIDs, you must also specify the eligible amount (always in the smallest currency unit).

Example:

```
  "vouchers":[
 {
   "contract_type":"CONECS",
   "eligible_amount":1900
 }
  ]
```

You have the possibility to mention several MIDs for the same item.

Example:

```
...
  "items":[
 {
   "seller":"6f167596-07f2-4256-9210-7b4ab54fc3b9",
   "reference":"Buffet1",
   "description":"Buffet 1",
   "amount":8000,
   "type":"FOOD",
   "vouchers":[
  {
    "contract_type":"CVCONNECT"
  },
  {
    "contract_type":"CONECS",
    "eligible_amount":1400
  }
   ]
 },
...
  ]
...
```

In this example, *“Buffet 1”* is payable:

- by Titre-Restaurant (CONECS) up to €14,
- by Chèque-Vacances (CVCONNECT),
- or by both.

In this case, depending on the payment method chosen by the end buyer, the order may eventually result in creating one, two or three transactions.

- 1 transaction:
  - By classic card (CB, VISA, MasterCard, etc.) for the whole amount of €80
  - Or by Chèque-Vacances for €80
- 2 transactions:
  - By CONECS + classic card (CB, VISA, MasterCard, etc.)
  - By CONECS + CVCONNECT
  - Or by CVCONNECT + classic card (CB, VISA, MasterCard, etc.)

- 3 transactions:
  - By CONECS + CVCONNECT + classic card (CB, VISA, MasterCard, etc.)

11.7.3. Selecting MIDs upon order execution

For each order, only one merchant can be paid by one of the MID types. Thus, if you indicate a **CONECS** MID for several sub-merchants, only one of them can be paid by this payment method.

How does it work? When the order is executed, the API calculates the total amount per MID of the amounts payable by voucher and then, for each MID, selects the merchant with the highest amount.

In the following (simplified) example:

```
...
  "items":[
 {
   "external_ref":"seller1",
   "ref":"Seller 1 Item 1",
   "amount":2000,
   "voucher":[
     {
    "contract_type":"CONECS",
    "eligible_amount":900
     }
   ]
 },
 {
   "external_ref":"seller1",
   "ref":"Seller 1 Item 2",
   "amount":4000,
   "voucher":[
  {
    "contract_type":"CONECS",
    "eligible_amount":500
  }
   ]
 },
 {
   "external_ref":"seller2",
   "ref":"Seller 2 Item 1",
   "amount":2000,
   "voucher":[
  {
    "contract_type":"CONECS",
    "eligible_amount":1900
  }
   ]
 }
  ]
```

Seller 2 will be selected, as their eligible amount is €19 versus €14 (= 9 + 5) for **Seller 1**.

The principle is the same for **CVCONNECT** MIDs.

The selection is independent from one MID to another, i.e. one sub-merchant can be selected for a **CONECS** MID and another can be selected for the **CVCONNECT** MID, while both offer the two MIDs.

NOTE
If the amounts to be split are identical, the decision is made according to the MID number.

At the end of the execution, each "*voucher*" object of the concerned articles receives the `is_selected` attribute that is populated depending on the selection.

For example:

```
...
  "status":"CREATED",
  "amount":8000,
  "items":[
 {
   "uuid":"fc300ee2-fd43-46e0-9314-770f05a5b338",
   ...
   "reference":"Buffet1",
   "description":"Buffet 1",
   "type":"FOOD",
   "amount":8000,
   ...
   "vouchers":[
  {
    "contract_type":"CVCONNECT",
    "is_selected":true
  },
  {
    "contract_type":"CONECS",
    "eligible_amount":1400,
    "is_selected":true
  }
   ]
 }
  ]
```

11.7.4. After the payment

The choice of payment method and the amount actually paid per MID is only visible at the end of the payment cycle.

From the marketplace point of view, this information is therefore only available from the moment the order status changes to **PENDING**.

If a voucher was used, the `actual_amount` attribute provides information about the amount that was affected to the item.

If we take the previous example, we can have the following "*item*" object:

```
...
"status":"PENDING",
"amount":8000,
"items":[{
  "uuid":"fc300ee2-fd43-46e0-9314-770f05a5b338",
  ...
  "reference":"Buffet1",
  "description":"Buffet 1",
  "type":"FOOD",
  "amount":8000,
  ...
  "vouchers":[{
    "contract_type":"CVCONNECT",
    "is_selected":true,
    "actual_amount":3000
    },
    {
    "contract_type":"CONECS",
    "eligible_amount":1400,
    "is_selected":true,
    "actual_amount":1100
    }]
  }]
```

In this example, we can see that the buyer paid €11 with a CONECS card and €30 with a holiday voucher (Chèque-Vacances). Since the total amount is €80, there will be a third transaction for the remaining amount (i.e. 80 - 30 - 11 = €39).

NOTE
Since an order can contain multiple items payable by voucher, the API automatically allocates the `actual_amount` in random order. For example,

```
...
"items":[{
  "reference":"Buffet1",
  "description":"Buffet 1",
  "type":"FOOD",
  "amount":8000,
  ...
  "vouchers":[{
    "contract_type":"CONECS",
    "eligible_amount":1400,
    "is_selected":true,
    "actual_amount":1400
    }]
  },
  {
  "reference":"Buffet2",
  "description":"Buffet 2",
  "type":"FOOD",
  "amount":5500,
  ...
  "vouchers":[{
    "contract_type":"CONECS",
    "eligible_amount":1100,
    "is_selected":true,
    "actual_amount":500
    }]
  },
  {
  "reference":"Buffet3",
  "description":"Buffet 3",
  "type":"FOOD",
  "amount":5500,
  ...
  "vouchers":[{
    "contract_type":"CONECS",
    "eligible_amount":1100,
    "is_selected":true,
    "actual_amount":0
    }]
  }]
...
```

In this case, the buyer seems to have used their daily credit amount of €19, which was allocated for the entire first item (€14), for a part of the second item (€5), and for none of the third item.
The credit of €19 could also be split as follows:

- **€11 for Buffet 2;**
- **€8 for Buffet 3;**
- **€0 for Buffet 1.**

11.7.5. Modification and cancellation

It is possible to **cancel** an order containing transactions that were paid by vouchers.

However, this type of transaction cannot be **modified**.

11.8. Payment using a persistent link

You can generate a persistent payment link and send it to the end buyer.

By default, they will then have 10 days to display the payment form and complete their order.

11.8.1. Generating the link

As soon as the Order is created, a “persist” link is available in the *Order* object at the associated “**links**” list level.

For example:

```
  ...
  "links": {
  ...
 "persist": {
   "href": "https://secure.lyra.com/marketplace/orders/c4e214db-3ba7-4646-84dc-4e247b1e4b5f/
persist"
    }
  ...
  },
  ...
```

In order to generate the payment link, simply call the corresponding resource:

```
GET /orders/{uuid}/persist
```

NOTE: This resource performs a redirection and returns the *Order* resource

Once generated, the `persist_url` link and the `expiry_date` will now be accessible every time you call the *Order* object (via a **POST**, **PUT** or **GET**).

For example:

```
  ...
 "persist_url": "https://secure.lyra.com/t/ioeD1uRP",
 "expiry_date": "2021-02-15T15:37:47.632009Z"
  ...
```

NOTE: In case of continuous direct debits, the commission amount is checked from the first call of the `GET /orders/{order}/persist` resource. The amount will be checked again when the payment form is called by the end buyer.

It is only possible to generate a persistent link on an *Order* with the “**CREATED**” status.

The links cannot be modified or deleted.

11.8.2. Adjusting the expiration date

By default, the expiration delay of the link is 10 days.

To adjust this delay, add the “*delay*” parameter during the link generation request.

For example:

```
GET https://secure.lyra.com/marketplace/orders/3c58e343-abbd-4c2b-9cac-42db158987b0/persist?
delay=5
```

The expiration delay cannot be modified after it has been created.

12. ANALYZING THE PAYMENT RESULT

The Marketplace server is notified via a *webhook* about each change in the statuses of the Order and Token objects.

The *webhook* is sent to the address specified by the marketplace during the integration phase.

This URL can be found in the "**webhook_url**" attribute of the Marketplace object.

The *webhook* is sent in POST with the "**Content-Type: application/json**" header.

Example of a notification (webhook)

```
{'order':'515abac9-6cb2-4e21-8a25-b08d7e41e43c'}
```

In PHP, this JSON can be retrieved using the following code:

```
$json = file_get_contents('php://input');
$data = json_decode($json);
$orderUuid = $data->order ?? false;
```

After that, the Marketplace has to call the GET ORDER (or TOKEN) resource with the transmitted UUID in order to know the new status of the object in question.

```
GET /orders/{uuid}
```

Example of a GET ORDER call

```
GET https://secure.lyra.com/marketplace/orders/515abac9-6cb2-4e21-8a25-b08d7e41e43c
```

Example of a GET ORDER response

```
{
  "uuid": "515abac9-6cb2-4e21-8a25-b08d7e41e43c",
  "href": "https://secure.lyra.com/marketplace/orders/515abac9-6cb2-4e21-8a25-b08d7e41e43c",
  "created_at": "2015-03-19T16:30:14.434Z",
  "updated_at": "2015-03-19T16:30:14.434Z",
  "marketplace": "9537e049-8862-400a-ae8d-da2ec9ca6051",
  "reference": "order00052",
  "description": "Order",
  "buyer":[
     {
       "reference": "nope775",
       "title": "Mrs",
       "type": "PRIVATE",
       "first_name": "Nathalie",
       "last_name": "Durand",
       "phone_number": "02 13 06 95 27",
       "email": "ndurand@tiscali.fr",
       "address": {
          "street_number": "29",
          "street": "rue Besnard",
          "district": "Île-de-France",
          "zipcode": "83819",
          "city": "Roux",
          "state": "Hautes-Pyrénées",
          "country": "FR"
       }
     }
  ],
  "shipping":[
     {
       "delivery_company_name": "DHL",
       "address": {
          "street_number": "493",
          "street": "avenue Duhamel",
          "district": "Saint-Martin",
          "zipcode": "33980",
          "city": "Hamon",
          "state": "Charente-Maritime",
          "country": "FR"
       },
       "shipping_speed": "EXPRESS",
       "shipping_method": "ETICKET",
       "type": "PRIVATE",
       "first_name": "Luc",
       "last_name": "Leveque",
       "phone_number": "+33 (0)1 46 05 15 89"
     }
  ],
  "amount": null,
  "currency": "EUR",
  "status": "CREATED",
  "webhook_result": 200,
  "url_return": "http://www.lyra-sms.com/",
  "links": {
     "items": {
       "href": "https://secure.lyra.com/marketplace/orders/515abac9-6cb2-4e21-8a25-
b08d7e41e43c/items"
     },
     "transactions": {
       "href": "https://secure.lyra.com/marketplace/orders/515abac9-6cb2-4e21-8a25-
b08d7e41e43c/transactions"
     },
     "refunds": {
       "href": "https://secure.lyra.com/marketplace/orders/515abac9-6cb2-4e21-8a25-
b08d7e41e43c/refunds"
     },
     "execute":{
       "href": "https://secure.lyra.com/marketplace/orders/515abac9-6cb2-4e21-8a25-
b08d7e41e43c/execute"
     },
     "execute-embedded": {
       "href": "https://secure.lyra.com/marketplace/orders/515abac9-6cb2-4e21-8a25-
b08d7e41e43c/execute-embedded"
     },
     ...
  },
  "vads_transaction_id": 500003,
  "vads_transaction_date": ""
}
```

In case of any doubts concerning the transmission or the reception of the *webhook*, you can check the "**webhook_result**" attribute of the resource in question. This attribute contains the "status code" of the HTTP query made by our server. For example,

- If it is null, the webhook has not (yet) been sent.
- If it shows 200, the webhook should have reached you.
- If it shows 404, we invite you to check the webhook that you declared.
- etc.

Note:

The transaction authorization code is recorded in the **auto_code** attribute returned by the transaction result (accessible via the GET Transaction resource (see *Transaction object* on page 80).

13. TOKEN MANAGEMENT

The Marketplace API uses two notions:

- **alias**
- **token**

An **alias** is an object that allows to store payment method data.

A **token** is an object associated with an alias. It also contains buyer details.

There are 2 ways of creating an alias (and a token):

1. During the payment (see chapter *Payment with token creation* on page 33). The alias and token are therefore associated with an Order.
2. Without an Order being created (see next chapter).

13.1. Creating a token

It is possible to request the recording of the payment method (and, therefore, the token creation) without it being linked to an order.

For this, you need to call the TOKEN API to create a token, with a query similar to the one used for creating a payment.

Create a token using the payment page

```
POST /tokens/
```

Example:

Request

```
POST https://secure.lyra.com/marketplace/tokens/
```

Body

```
{
 "marketplace": "9537e049-8862-400a-ae8d-da2ec9ca6051",
 "url_return": "http://www.my-website.com/",
 "buyer": {
  "type": "PRIVATE",
  "first_name": "Jean",
  "last_name": "Dupond",
  "email": "jean.dupond@lyra.fr",
  "phone_number": "1234",
  "reference": "Acheteur_1",
  "address": {
   "zipcode": "59259",
   "street_number": "37",
   "country": "FR",
   "street": "rue Marcel Philippe",
   "city": "Vignoux"
  }
 }
}
```

Response

```
{
 "token": "d3329266-c8d6-421c-8d2a-10a8ffbcaef6",
 "payment_url": "https://secure.lyra.com/vads-payment/
exec.refresh.a;jsessionid=335D2aDb5eF8356Aed2cf3dF.vadpayment02inte01lbg?
cacheId=91335531181127600040"
}
```

Once the token has been created, the Buyer must be redirected to the URL transmitted in **payment_url** in order to proceed to payment method recording (and, therefore, to creating an alias).

Create a token using the embedded form

```
POST /tokens/embedded/
```

Example:

Request

```
POST https://secure.lyra.com/marketplace/tokens/embedded/
```

Body

```
{
 "marketplace": "9537e049-8862-400a-ae8d-da2ec9ca6051",
 "url_return": "http://www.my-website.com/",
 "buyer": {
  "type": "PRIVATE",
  "first_name": "Jean",
  "last_name": "Dupond",
  "email": "jean.dupond@lyra.fr",
  "phone_number": "1234",
  "reference": "Acheteur_1",
  "address": {
   "zipcode": "59259",
   "street_number": "37",
   "country": "FR",
   "street": "rue Marcel Philippe",
   "city": "Vignoux"
  }
 }
}
```

Response

```
{
 "token": "d3329266-c8d6-421c-8d2a-10a8ffbcaef6",
 "form_token":"fa0yfV2FQuR3aak1SwsgcuZg195eyJhbW91bnQiOjEwOTAwLCJjdXJyZW5jeSI6IkVVUiIsIm
    1vZGUiOiJURVNUIiwidmVyc2lvbiI6Mywib3JkZXJJZCI6IjVjNDk1N2E5LWNkOGEtNGJhZC1hMGNiLTgxMT
    ZiMWE3ZDdmNiIsInNob3BOYW1lIjoiTHlyYSBTTVMiLCJicmFuZFByaW9yaXR5IjpbIkJBTkNPTlRBQ1QiL
    CJDQiIsIkUtQ0FSVEVCTEVVRSIsIk1BU1RFUkNBUkQiLCVGQVVMVCJ9fX0a702"
}
```

Once the token is created, the **form_token** must be inserted in the form (**kr-form-token)** attribute) to proceed to payment method recording (and, therefore, to creating an alias).

13.2. Analyzing the result of a token request

As with a payment, the Marketplace will be notified via the *webhook* about the progress of a token request.

Example of a notification (webhook):

```
{"token":"d3329266-c8d6-421c-8d2a-10a8ffbcaef6"}
```

The transmitted identifier will enable the Marketplace to execute a GET TOKEN to see if the token was successfully created.

```
GET /tokens/{uuid}
```

Note:

During a payment with token creation, you will receive two notifications: one upon creating the Order and one upon creating the token (use the ID of the token returned in the notification to find the corresponding token).

Example of a GET TOKEN call:

```
GET https://secure.lyra.com/marketplace/tokens/d3329266-c8d6-421c-8d2a-10a8ffbcaef6
```

Example of a GET TOKEN response:

```
{
 "uuid": "d3329266-c8d6-421c-8d2a-10a8ffbcaef6",
 "created_at": "2018-11-20T12:53:51.547541Z",
 "updated_at": "2018-11-20T12:53:52.513541Z",
 "marketplace": "9537e049-8862-400a-ae8d-da2ec9ca6051",
 "buyer": {
  "reference": "Acheteur_1",
  "title": "MR",
  "type": "PRIVATE",
  "first_name": "Jean",
  "last_name": "Dupond",
  "phone_number": "012345678",
  "email": "jean.dupond@lyra.fr",
  "address": {
   "street_number": "37",
   "street": "rue Marcel Philippe",
   "district": null,
   "zipcode": "59259",
   "city": "Martin",
   "state": null,
   "country": "FR"
  }
 },
 "language": "fr",
 "url_return": "http://www.my-website.com/",
 "status": "SUCCEEDED",
 "alias": "1144951ea7ab42989c97159b3dfc0382",
 "alias_to_update": null,
 "payment_url": "https://secure.lyra.com/vads-payment/
exec.refresh.a;jsessionid=CDb37C1CcfC5eA2BE82bDCA6?cacheId=91335531181120600004O"
}
```

13.3. Updating an alias

In order to update a payment method, you must create a new token and add the existing alias to the **alias_to_update** attribute of the POST TOKEN request.

Updating an alias using the payment page

Example:

Request

```
POST https://secure.lyra.com/marketplace/tokens/
```

Body

```
{
 "marketplace": "9537e049-8862-400a-ae8d-da2ec9ca6051",
 "alias_to_update": "1144951ea7ab42989c97159b3dfc0382",
 "url_return": "http://www.my-website.com/",
 "buyer": {
  "type": "PRIVATE",
  "first_name": "Jean",
  "last_name": "Dupond",
  "email": "jean.dupond@lyra.fr",
  "phone_number": "1234",
  "reference": "Acheteur_1",
  "address": {
   "zipcode": "59259",
   "street_number": "37",
   "country": "FR",
   "street": "rue Marcel Philippe",
   "city": "Vignoux"
  }
 }
}
```

Response

```
{
 "token": "62a52e50-ce29-409a-9cec-9ea6ee36ab41",
 "payment_url": "https://secure.lyra.com/vads-payment/
exec.refresh.a;jsessionid=335D2aDb5eF8356Aed2cf3dF.vadpayment02inte01lbg?
cacheId=9133553118112760000040"
}
```

In order to proceed to payment update, you must redirect the Buyer to the URL transmitted in **payment_url**.

You will receive a webhook notification at the end of the operation.

Updating an alias using embedded form

Example:

Request

```
POST https://secure.lyra.com/marketplace/tokens/embedded/
```

Body

```
{
 "marketplace": "9537e049-8862-400a-ae8d-da2ec9ca6051",
 "alias_to_update": "1144951ea7ab42989c97159b3dfc0382",
 "url_return": "http://www.my-website.com/",
 "buyer": {
  "type": "PRIVATE",
  "first_name": "Jean",
  "last_name": "Dupond",
  "email": "jean.dupond@lyra.fr",
  "phone_number": "1234",
  "reference": "Acheteur_1",
  "address": {
   "zipcode": "59259",
   "street_number": "37",
   "country": "FR",
   "street": "rue Marcel Philippe",
   "city": "Vignoux"
  }
 }
}
```

Response

```
{
 "token": "62a52e50-ce29-409a-9cec-9ea6ee36ab41",
 "form_token":"fa0yfV2FQuR3aak1SwsgcuZg195eyJhbW91bnQiOjEwOTAwLCJjdXJyZW5jeSI6IkVVUiIsIm
    1vZGUiOiJURVNUIiwidmVyc2lvbiI6Mywib3JkZXJJZCI6IjVjNDk1N2E5LWNkOGEtNGJhZC1hMGNiLTgxMT
    ZiMWE3ZDdmNiIsInNob3BOYW1lIjoiTHlyYSBTTVMiLCJicmFuZFByaW9yaXR5IjpbIkJBTkNPTlRBQ1QiL
    CJDQiIsIkUtQ0FSVEVCTEVVRSIsIk1BU1RFUkNBUkQiLCVGQVVMVCJ9fX0a702"
}
```

In order to proceed to payment method update, the **form_token** must be inserted in the form (**kr-form-token)** attribute).

13.4. Retrieving token details

One a token is created, it is possible to obtain its details by calling the token summary service.

```
GET /marketplaces/{marketplace}/alias/{alias}
```

Example:

Request

```
GET https://secure.lyra.com/marketplace/marketplaces/9537e049-8862-400a-ae8d-da2ec9ca6051/
alias/1144951ea7ab42989c97159b3dfc0382
```

Response

```
 {
 "brand": "CB",
 "expiry_month": "6",
 "expiry_year": "2023",
 "number": "497010XXXXXX0000",
 "first_name": "Jean",
 "last_name": "Dupond",
 "email": "jean.dupond@lyra.fr",
 "creation_date": "2019-01-21T16:01:19Z"
}
```

Another example response file with an IBAN type token:

```
{
    "brand": "SDD",
    "expiry_month": "5",
    "expiry_year": "2023",
    "number": "FR7630002005701234567890158_CRLYFRPPXXX",
    "first_name": "Jean",
    "last_name": "Dupond",
    "email": "jean.dupond@lyra.fr",
    "creation_date": "2020-05-12T14:24:13Z"
 }
```

14. PROCESSING THE RETURN TO THE MARKETPLACE

In order to redirect the buyer to the Marketplace, several attributes of the ORDER resource can be specified:

- "**url_return**" is the default return URL. You can use only this attribute if you do not want to differentiate return cases. The following attributes override its value on a case-by-case basis.
- "**url_success**" is the URL called if the payment is successfully completed.
- "**url_refused**" is the URL called if the payment is refused.
- "**url_cancel**" is the URL called if the payment is canceled.
- "**url_error**" is the URL called if the payment results in an error.

By default, the payment data is sent to the return URL in an HTTP GET form (in the “query string”).

Example:

```
https://mymarketplace.com/return_to_shop?ref=1234&customer=ABCD
```

This behavior can be overridden via the "**return_mode**" attribute that can take the following values:

- '**NONE**': no parameters are sent to the return URL.
- '**POST**': the parameters are sent to the return URL in an HTTP POST form (if the return to the shop is done in a non-https environment, the browser will display a security pop-up message to the buyer).
- '**GET**' (by default): The return fields are transmitted to the return URL in an HTTP GET form (in the “query string”).

<u>**Note**</u>: the return to the Marketplace should only allow you to show visual context to the buyer. Do not use the received data for processing in the database or for checking the payment status.

15. UPDATING AN ORDER

An order can be updated as long as its status is transient (**CREATED** or **PENDING**).

Therefore, the initial transaction must not be captured in the bank.

In case of an Order with the **CREATED** status, all values are editable, without limitations (See step 2 of chapter *Making a payment* on page 25).

In case of an Order with the **PENDING** status, the PUT request is rejected if the two following values are identical to the ones in the previous record:

- **amount**, determined by the total amount of the items.

  The update request cancels and replaces all items of the initial Order.
- **expected_capture_date** in UTC in the YYYYMMDDHHMMSS format.

  The comparison of expected_capture_date takes only the date into account.

  For example, 20200101100000 (1 January 2020 at 10 a.m. UTC) equals to 20200101180000 (1 January 2020 at 18 p.m. UTC).

Furthermore:

- The amount cannot be higher than the initial amount of the order (i.e. value recorded in **initial_amount**).
- The items must imperatively be transmitted in the PUT query (?expand=items).

Example of a PUT ORDER call:

```
PUT https://secure.lyra.com/marketplace/orders/515abac9-6cb2-4e21-8a25-b08d7e41e43c?
expand=items
```

```
{
 "marketplace": "ebfb36ab-2d30-4326-adb9-e16b0c9a89f3",
 "reference": "MKP BURO example",
 "description": "Office supplies order",
 "currency": "EUR",
 "url_return": "https://URLreturn.com",
 "expected_capture_date" : "202006205352",
 "items": [{
  "seller": "d0f80202-0676-4d8d-9247-f455f30aec1b",
  "reference": "commburo",
  "description": "commission BURO",
  "amount": 150,
  "is_commission": true
 },
 {
  "seller": "5d0ef88c-3345-4b33-948e-80e23d553b73",
  "reference": "STV1",
  "description": "Green pens",
  "amount": 1000,
  "is_commission": false
 },
 {
  "seller": "d0f80202-0676-4d8d-9247-f455f30aec1b",
  "reference": "del",
  "description": "Delivery",
  "amount": 500,
  "is_commission": false
 }
 ],
 "buyer": {
  "type": "PRIVATE",
  "email": "john.smith@buyer.com",
  "phone_number": "+33 (0)1 11 22 33 44",
  "reference": "ZhIsYM"
 },
 "shipping": {
  "address": {
   "zipcode": "59123",
   "street_number": "37",
   "country": "FR",
   "street": "rue Marcel Philippe",
   "city": "Nantes"
  },
  "shipping_method": "RELAY_POINT"
 }
}
```

15.1. Updating an order paid in installments

It is possible to edit transactions after the payment as long as they are pending (with the PENDING status).

Only two values can be changed:

- The transaction amount (**amount** attribute),
- The capture date of the transaction (**expected_capture_date** attribute).

Notes:

- Similarly to a mono-transactional order, it is not possible to increase the transaction amount beyond its initial amount, as recorded in the read-only **initial_amount** attribute.
- Items must imperatively be transmitted in the PUT query (?expand=items).
- It is possible to delete one (or several) transaction(s) (see below), but currently it is not possible to add transactions.

Procedure:

Transaction update uses the same update command as for an order update (PUT ORDER), with two distinct aspects:

1. The complete list of order transactions must be added in the PUT ORDER request body in a **transactions** list, including each of the potentially edited **trans_uuid**, **expected_capture_date** and **amount** attributes.

   For example:

   ```
   {
    "marketplace": "2434c0a2-9d46-4e96-9553-1536c898625b",
    "reference": "MyMultitransactionOrder01",
    "description": "Order update.",
    "currency": "EUR",
    "url_return": "https://www.my-website.com/",
    "language": "en",
    "items": [
      {
        "seller": "4d20a9d4-0526-4474-b452-e936dc25418d",
        "reference": "0000001",
        "description": "Flat screen TV",
        "amount": 390000,
        "quantity": 1,
        "is_commission": false
      },
      {
        "seller": "72ccc2ff-b455-4653-847e-deb6fee99f8d",
        "reference": "0000002",
        "description": "Commission",
        "amount": 14755,
        "quantity": 1,
        "is_commission": true
      }
    ],
    "transactions": [
      {
        "trans_uuid": "c383739d12a6489badc3bb6847db84cc",
        "payment_scheme": "CB",
        "amount": 14755,
        "expected_capture_date": "201909185743"
      },
      {
        "trans_uuid": "d0c4d34249d540af87ff8df3a2fa314a",
   ```

```
      "payment_scheme": "CB",
      "amount": 190000,
      "expected_capture_date": "201905175352"
    },
    {
      "trans_uuid": "8c5a6788b3334d368185b0a567dd7bcd",
      "payment_scheme": "CB",
      "amount": 200000,
      "expected_capture_date": "201904205352"
    }
  ],
  "buyer": {
  …
  }
}
```

2. “**transactions**” must be added in the “**expand**” URL attribute.

   Therefore, you must submit a PUT ORDER resource to

   https://secure.lyra.com/marketplace/orders/{order_uuid}?expand=items,transactions.

   If the "**transactions**" value is not indicated, transaction updates cannot be taken into account.

During the update, the application checks if the item and transaction amounts match: normally, the total amount of the item must be identical to the total amount of the transactions. Thus, a decrease in the item amount should be followed by a manual decrease of the transaction amount.

Special case of transaction deletion

The decrease of the item amount may exceed the amount of one of the transactions. If you choose to delete one of the transactions instead of decreasing the amount across all transactions, simply exclude it from the list of transactions submitted in the update request. It will then be interpreted as canceled and its status will change to CANCELLED.

Other remarks:

- The application returns an error if the update does not include any changes (concerning transactions, items or order values).
- The new value of **expected_capture_date** must not be before the date and time of the update.

16. CANCELING AN ORDER

An order can be canceled before and after the payment, as long as the latter has not been captured at the bank, i.e. its status is:

- **CREATED** and *before the order execution*,
- **PENDING**.

If you try to cancel the payment between the time of order execution (while the status is still CREATED) and the end of the payment process (i.e. the transition from CREATED to PENDING), you will receive the following error:

```
{'Order': "This order has been executed and cannot be canceled until the end of the payment
 process"}
```

To cancel an order you must use the DELETE method on the ORDER resource.

```
DELETE /orders/{uuid}
```

Example:

```
DELETE https://secure.lyra.com/marketplace/orders/60c9dbf5-ff99-40fb-9fb6-a709005359f8
```

In case of success, the server responds by a **HTTP 204** code (NO CONTENT).

17. REFUNDING A PAYMENT

To refund a payment, you must indicate which order it is linked to and which seller will take on the cost of the refund.

To request a payment refund, follow the 2 following steps:

1. **Create a refund request**
2. **Follow up the request**

17.1. Creating a refund request

This step allows to create the refund request by using a POST request on the REFUND resource:

```
POST /refunds/
```

When a refund request is created, the following elements are controlled by the API and are subject to a 400 returned error code if they are not verified:

- The refund request must have the SUCCEEDED status.
- The seller that performs the refund must be one of the sellers of the refunded order.
- The refund must concern only one seller. If several sellers must refund a part of the same order, several separate refund requests must be created: one for each seller. The only exception is if the second seller of the request is the marketplace administrator.

***Note**: the refunded amount is not verified at this stage. The verification occurs later on, in scheduled process chains, that is reflected in the change of the processing request status.*

Example of a call POST REFUND

In this example of the JSON content to post, a refund request of €130 is created, where €120 are refunded by the seller and €10 are refunded by the marketplace manager:

Request

```
POST https://secure.lyra.com/marketplace/refunds/
```

Body

```
{
 "order": "9537e049-8862-400a-ae8d-da2ec9ca6051",
 "reference": "remb000045",
 "description": "Refund Mrs Smith 001",
 "currency": "EUR",
 "items": [{
  "seller":  "dfc42a76-10b5-421a-91cd-c288c8265c92",
  "reference": "remb000045a",
  "description": "Phoneshop",
  "amount": 12000
 },
 {
  "seller":"975e2a43-7e72-438c-a2b2-b61347aa160c",
  "reference":"remb000045b",
  "description": "Manager",
  "amount": 1000
 }]
}
```

Example response of POST REFUND

The **amount** attribute in the response is calculated automatically based on the posted amounts of each of the parts.

```
{
 "uuid": "16ad9da8-b9cb-11e4-97c6-b1229586dec7",
 "href":"https://secure.lyra.com/marketplace/refunds/16ad9da8-b9cb-11e4-97c6-b1229586dec7",
 "created_at": "2018-06-08T12:36:56.681073Z",
 "updated_at":"2018-06-08T12:39:46.859402Z",
 "order": "9537e049-8862-400a-ae8d-da2ec9ca6051",
 "reference": "remb000045",
 "description": "Refund Mrs Smith 001",
 "amount": 13000,
 "currency": "EUR",
 "status":"CREATED",
 "items": [ {
  "seller": "dfc42a76-10b5-421a-91cd-c288c8265c92",
  "reference": "remb000045a",
  "description":"Phoneshop",
  "amount":12000,
  "is_commission": false
 },
 {
  "seller":"975e2a43-7e72-438c-a2b2-b61347aa160c",
  "reference": "remb000045b",
  "description": "Manager",
  "amount": 1000,
   "is_commission": false
 }]
}
```

17.2. Following up the refund request

Refunds are executed with regard to the available balance of the Merchant and the Marketplace. For example, a refund of 100 euros with 80 euros charged to the Merchant and 20 euros charged to the Marketplace will only be executed if:

- The available balance of the sub-merchant is greater than or equal to 80 euros.
- If the Marketplace balance is 20 euros or more.

To track the progress of the request, set up a *webhook* on the refund object and/or check the status of the REFUND with a GET:

```
GET /refunds/{uuid}
```

Throughout its lifecycle, a refund request can go through the following statuses:

- **CREATED** : Refund request registered by the API, waiting to be processed.
- **PENDING** : Request sent to the processing chains and is being processed.
- **SUCCEEDED** : Refund processing successfully completed.
- **FAILED** : Refund request rejected, for example due to insufficient funds on the seller's account.
- **CANCELLED** : Refund request canceled.

Example:

Request

```
GET https://secure.lyra.com/marketplace/refunds/16ad9da8-b9cb-11e4-97c6-b1229586dec7
```

Response

```
{
  "uuid": "16ad9da8-b9cb-11e4-97c6-b1229586dec7",
  "href": "https://secure.lyra.com/marketplace/refunds/16ad9da8-b9cb-11e4-97c6-b1229586dec7",
  "created_at": "2018-06-08T12:36:56.681073Z",
  "updated_at": "2018-06-08T12:51:21.241448Z",
  "order": "9537e049-8862-400a-ae8d-da2ec9ca6051",
  "reference": "remb000045",
  "description": "Remboursement Mme Lafont 001",
  "amount": 13000,
  "currency": "EUR",
  "status": "PENDING",
  "items":
 [
   {
  "seller": "dfc42a76-10b5-421a-91cd-c288c8265c92",
  "reference": "remb000045a",
  "description": "Phoneshop",
  "amount": 12000,
  "is_commission": false
   },
   {
  "seller": "975e2a43-7e72-438c-a2b2-b61347aa160c",
  "reference": "remb000045b",
  "description": "Gestionnaire",
  "amount": 1000
  "is_commission": false
   }
 ]
}
```

17.3. Modifying a refund request

A refund in progress can be modified within the limit of its initial amount as long as it has not been executed (CREATED or PENDING status).

The modification is made using the UUID of the corresponding refund, via the resource:

```
PUT /refunds/{uuid}
```

Note: If the refund amount is unchanged, the server returns an **HTTP 400** error.

Example:

Request

```
PUT https://secure.lyra.com/marketplace/refunds/16ad9da8-b9cb-11e4-97c6-b1229586dec7
```

Body

```
{
  "order":  "9537e049-8862-400a-ae8d-da2ec9ca6051",
  "reference": "remb000045",
  "description": "Remboursement Mme Lafont 001",
  "currency": "EUR",
  "items": [
      {
        "seller": "dfc42a76-10b5-421a-91cd-c288c8265c92",

        "reference": "remb000045a",
        "description":"Phoneshop",
        "amount": 9450,
        "is_commission": false
      },
      {
        "seller":"975e2a43-7e72-438c-a2b2-b61347aa160c",

        "reference": "remb000045b",
        "description": "Gestionnaire",
        "amount": 900,
        "is_commission":false
      }
  ]
}
```

Response

```
{
  "uuid": "83402cc0-d969-443b-a72a-0f2fe9557879",
  "href": "https://secure.lyra.com/marketplace/refunds/83402cc0-d969-443b-a72a-0f2fe9557879",
  "created_at": "2019-06-11T09:31:59.659687Z",
  "updated_at": "2019-06-11T09:52:51.959406Z",
  "order": "67ca248e-99cd-4345-9795-5873bb6fd8f2",
  "reference": "remb000045",
  "description": "Remboursement Mme Lafont 001",
  "amount": 10905,
  "currency": "EUR",
  "status": "PENDING",
  "transaction": null,
  "items": [
      {
        "seller": "dfc42a76-10b5-421a-91cd-c288c8265c92",
        "refund": "83402cc0-d969-443b-a72a-0f2fe9557879",
        "item": "9f7c6d31-deff-4299-a3db-71bb214cae9e",
        "reference": "remb000045a",
        "description":"Phoneshop",
        "amount": 9450,
        "is_commission": false
      },
      {
         "seller":"975e2a43-7e72-438c-a2b2-b61347aa160c",
         "refund": "83402cc0-d969-443b-a72a-0f2fe9557879",
         "item": "7b99691b-0492-42ff-a819-302ac178fc7e",
         "reference": "remb000045b",
         "description": "Gestionnaire",
         "amount": 900,
         "is_commission": false
       }
   ]
 }
```

17.4. Canceling a refund request

A refund in progress can be canceled as long as it has not been executed (CREATED or PENDING status).

```
DELETE /refunds/{uuid}
```

For example:

```
DELETE https://secure.lyra.com/marketplace/refunds/83402cc0-d969-443b-a72a-0f2fe9557879
```

In case of success, the server responds by a **HTTP 204** code (NO CONTENT).

If the refund status does not allow cancellations, the server returns an **HTTP 405** error.

18. MANUALLY TRIGGERING ITEM PAYMENT

Normally, the payment of a commission item occurs automatically after the expiry of the delay for withholding the seller’s funds in relation to the item.

The Marketplace may retain this payment in order to control when it is triggered.

This is the case for a service that spreads over a period of time. In this case, the client has made their payment before the beginning of the service, but the seller will only receive the payment once the Marketplace verifies that the service has been provided.

Then, the Marketplace:

- during the order creation, indicates the item(s) concerned by the withheld payment by setting the **hold_payment** attribute to **true**:

```
"items": [
  {
      "seller": "4d20a9d4-0526-4474-b452-e936dc25418d",
      "reference": "cruise12345",
      "description": "Boat cruise",
      "amount": 245000,
      "quantity": 1,
       "is_commission": false,
      "hold_payment": true
  },
  {
      "seller": "4d20a9d4-0526-4474-b452-e936dc25418d",
      "reference": "nauticalchart",
      "description": "Nautical chart - Martinique",
      "amount": 1250, "quantity": 1,
      "is_commission": false
  },
 {
      "seller": "72ccc2ff-b455-4653-847e-deb6fee99f8d",
      "reference": "cruise12345com",
      "description": "Commission on boat cruise",
      "amount": 16450,
      "quantity": 1,
      "is_commission": true
  }
  ]
```

- and, and the desired moment, unlocks the transfer by making a POST request (without a request body) to the following address:

```
POST /items/{uuid}/activate
```

In case the operation is successful, the request is redirected (status_code = 302) to the item details, which then indicates **hold_payment**: **false**.

Note:

- Payments can only be withheld for cart items of a Marketplace type seller (seller with "is_marketPlace = true").
- Once the item payment has been activated, it cannot be withheld again.
- The **hold_payment** attribute is not required. On the contrary, it is recommended to omit it (or to set it to “**hold_payment : null**”) for all items whose transfer must be handled automatically, in order to distinguish the items that have been withheld from others.

19. VIEWING MARKETPLACE CASHOUTS

19.1. Understanding cashouts

A cashout is a transfer from the sub-merchant's holding account to their own bank account.

The operation is carried out automatically on **D + d** days after the payment, unless the Marketplace operator has blocked it in order to *activate it manually*.

- ***D** being the transaction capture delay.*
- ***d** being the "cashout delay" of the sub-vendor as it was configured by the Marketplace operator (see the step **Creating the seller** in Marketing-Onboarding KYC).*

For example, Mr Jones pays for an order today (***D***). If the capture takes place tomorrow (***D+1***) and if the Marketplace has defined a cashout period of 2 days for the sub-merchant, the transfer will take place in 3 days.

Thus, its amount corresponds to the sum of transactions carried out (and captured) ***d*** days earlier minus the refunds made (and captured) in the meantime.

For example, if the cashout delay is 2 days and 4 transactions of €250 have been captured the day before yesterday, including an item refunded yesterday, the cashout amount for this day will be equal to: **(4 - 1) x €250 = €750**.

19.2. The cashout process

The `transfer` object mediates between the transaction and the cashout.

In case of a Marketplace:

- A `transaction` may concern several items of several different sub-vendors for a single *buyer*.
- A `cashout` aggregates several `transactions` of several different orders for a single *vendor*.

The `transfer` represents the share that an item occupies for a single transaction (knowing that an order can itself be divided into several transactions).

It is entirely managed by the Marketplace API.

For example, a Marketplace gives Mr Jones the possibility to pay for the following 3 items in 3 installments. Transfers represent the body of the table, while the first column contains the items and the other columns contain the transactions.

		Transactions			**Total Items**
		1st installment	2nd installment	3rd installment	
Items	Computer	€740,88	€740,88	€518,23	**€1,999,99**
	Large screen	€926,10	€926,10	€647,79	**€2,499,99**
	Sound bar	€333,02	*€333,02*	€232,96	**€899,00**
	Total Transactions	**€2,000,00**	**€2,000,00**	**€1,398,98**	**€5,398,98**

In this order 9 transfers are generated using a proportional distribution rule. For example, the transfer corresponding to the large screen in the first transaction equals **€926,10 (= 2000,00 x 2499,99 / 5398,98)**. If you do the exact calculations, you will notice that an item (here, *the sound bar*) and a transaction (*the last one*) are used to correct the rounding to the nearest cent.

In the global Marketplace process, the `transfer` is *the smallest unit of an order, and the key to transforming cash-in into cash-out*. Since it is at the junction of the two operations, it is generated as soon as the transaction is identified as captured (i.e. its status is **SUCCEEDED**). It then takes the **CREATED** status.

Once the cash-out is captured, it changes to the **SUCCEEDED** status. This status is communicated to the `transfer` associated with it, whose status also changes to **SUCCEEDED**.

NOTE
By definition, products sold by the Marketplace operator itself are not subject to cashouts. Therefore, *only the Marketplace sub-merchants are concerned by the transfers*. As a result, transfers to the Marketplace (own sales and `item commission`) retain their CREATED status throughout their lifetime.

19.3. Identifying the cashout and the associated orders

How to establish a connection between a seller's cashout and an order made on the Marketplace using the information returned by the Expert Back Office and that of the Marketplace API?

In the Merchant Back Office and the transaction statements, cashouts can be identified by their `capture_label`.

For example, `LC 22372582900407` for "*Lyra Collect*". Another example: an association enters the name of the Marketplace and a transfer number, such as `MYMKPSLR88888444`.

By locating the cashout in question in the *list of cashouts*, you can then interrogate the *cashout details* and find the associated orders, items and refunds.

19.4. List of cashouts

The list of marketplace cashouts can be viewed at the following URL:

```
GET /cashouts/
```

The results are returned as follows:

```
{
  "count": 3,
  "next": null,
  "previous": null,
  "results": [
    {
      "href": "/marketplace/cashouts/749e668e-526d-41ec-8e7c-e5afc3d89ddc",
      "uuid": "749e668e-526d-41ec-8e7c-e5afc3d89ddc",
      "seller": "b646ea68-a145-4a5a-8a6f-55e8f68643dd",
      "seller_external_ref": "ref039412",
      "ref": "demoeYDPnZuC",
      "status": "CREATED",
      "amount": 1900,
      "currency": "EUR",
      "captured_at": "2019-04-04",
      "capture_label": "MYMKPSLR1012324"
    },
    {
      "href": "/marketplace/cashouts/b92d904a-8613-4368-98b7-96d75a623d97",
      "uuid": "b92d904a-8613-4368-98b7-96d75a623d97",
      "seller": "f8dcc611-bbaa-411a-8f28-ea2d6e4f49a8",
      "seller_external_ref": "ref09523",
      "ref": "demoeYDPnZuC",
      "status": "CREATED",
      "amount": 1900,
      "currency": "EUR",
      "captured_at": "2019-03-03",
      "capture_label":"MYMKPSLR1012325"
    },
    {
      "href": "/marketplace/cashouts/f9132b0b-8d42-4409-b3e1-c1c6d711688b",
      "uuid": "f9132b0b-8d42-4409-b3e1-c1c6d711688b",
      "seller":"4d20a9d4-0526-4474-b452-e936dc25418d",
      "seller_external_ref": "ref012345",
      "ref": "demoeYDPnZuC",
      "status": "CREATED",
      "amount": 1450,
      "currency": "EUR",
      "captured_at": "2019-01-01",
        "capture_label": "MYMKPSLR1012326"
      }
    ]
  }
```

These results are sorted by capture date in descending order (attribute **captured_at**).

They are displayed in a paginated manner. There are 100 cashouts per page.

The "**next**" and "**previous**" links, when they are populated, allow to navigate between pages.

It is also possible to filter the results of this list by their capture date.

In this case, the **capture_start_date** and/or **capture_end_date** parameters must be defined as follows:

```
GET https://.../marketplace/cashouts/?capture_start_date=2019-02-01&capture_end_date=2019-03-31
```

```
GET https://.../marketplace/cashouts/?capture_start_date=2019-02-01
```

```
GET https://.../marketplace/cashouts/?capture_end_date=2019-01-01
```

If the **capture_start_date** parameter is specified but **capture_end_date** is not, the latter is set to the current date ("today").

On the other hand, if the **capture_start_date** parameter is omitted but **capture_end_date** is specified, the selection will take the cashouts from the beginning until the specified date.

Note: Cashouts with an unspecified "captured_at" attribute are excluded from the results.

19.5. Cashout details

The list of cashouts provides the URL for accessing the details of each of the cashouts, for example:

`"/marketplace/cashouts/f9132b0b-8d42-4409-b3e1-c1c6d711688b"`.

A GET on the URL allows to retrieve information on the transfers and refund associated with the cashout (including order and item details):

```
GET /cachouts/{uuid}
```

Example:

Request

```
GET https://secure.lyra.com/marketplace/cashouts/f9132b0b-8d42-4409-b3e1-c1c6d711688b
```

Response

```
{
 "href": "/marketplace/cashouts/f9132b0b-8d42-4409-b3e1-c1c6d711688b",
 "uuid": "f9132b0b-8d42-4409-b3e1-c1c6d711688b",
 "seller": "4d20a9d4-0526-4474-b452-e936dc25418d",
 "seller_external_ref": "ref012345",
 "ref": "demoeYDPnZuC",
 "status": "CREATED",
 "amount": 1450,
 "currency": "EUR",
 "captured_at": "2019-01-01",
 "capture_label": null,
 "transfers": [
   {
       "uuid": "0a5a8c7e-9c8b-4f16-8a29-0018c5aa20ef",
       "created_at": "2019-04-12T12:45:54.705798Z",
       "updated_at": "2019-05-06T12:15:09.942029Z",
       "item": {
       "uuid": "e8950426-f13c-4b18-8d27-17f2e6bbca8b",
       "ref": "demoeYDPnZuC",
       "desc": "Meal"
   },
   "order":
       {
              "uuid": "0bfa5ebf-cd09-4fd8-bb78-465d17854b55",
              "ref": "TestMKP",
              "desc": " Test marketplace "
       },
   "amount": 1900,
   "currency": "EUR"
   }
 ],
 "refunds": [
   {
       "uuid": "2760e1dd-252b-4109-816c-efbeafa2eaa2",
       "created_at": "2019-04-10T14:06:34.076796Z",
       "updated_at": "2019-04-10T14:06:35.719923Z",
       "order": {
              "uuid": "06f0fbd9-4a1f-4828-a014-aafda50df703",
              "ref": "TestMKP",
              "desc": "Test marketplace"
       },
       "item": {
              "uuid": "00f5a201-3d0c-49e7-9d05-0e706d04a385",
              "ref": "demoeYDPnZuC",
              "desc": "Meal"
       },
       "amount": 450,
       "currency": "EUR"
   }
 ]
}
```

20. GENERATING A CLIENT VIA OPENAPI

The Marketplace API uses **Swagger** (*https://swagger.io/*) to facilitate the generation of a ready-to-use client in a wide variety of languages.

How does it work?

1. Retrieve the Marketplace API template via this address:
   - In the YAML format: *https://secure.lyra.com/marketplace/open_api.yaml*
   - In the JSON format: *https://secure.lyra.com/marketplace/open_api.json*
2. Copy the model in the left window of the Swagger editor (*https://editor.swagger.io/*).

   You can safely ignore the error "Semantic error at paths./orders/.post.operationId Operations must have unique operationIds".
3. In the menu of the Swagger editor, click **Generate Client** and select the language of your choice (PHP, Python, etc.).
4. After downloading and extracting the generated code, follow the instructions of the README.md file.

That's it!

21. DATA DICTIONARY

21.1. Address object

All the attributes of this object are required if at least one attribute is populated. Therefore, if no attributes are populated, no attributes are required.

Name	Description	Format	Mandatory	Example
street_number	Street number	an..5		*12*
street	Street name	ans..255		*market street*
district	Address supplement	ans..127		*central district*
zipcode	Zip code	an..64		*75015*
city	City	an.128		*London*
state	State	ans..127		*London*
country	Country	a2		*FR for France* *GP for Guadeloupe* *PF for French Polynesia*

21.2. Alias object

Name	Description	Format	Example
brand	Card brand	string	*CB*
expiry_month	Card expiry month between 1 and 12	n..2	*3*
expiry_year	Expiration year in 4 digits	n4	*2023*
number	Masked card number	string	*497010XXXXXX0000*
first_name	Buyer's first name	string	John
last_name	Buyer's last name	string	*Dupond*
email	Buyer's e-mail address	string	*jean.dupond@example.com*
creation_date	Token creation date	see example	*2020-04-21T16:01:19Z*

21.3. Buyer object

Name	Description	Format	Mandatory	Example
reference	buyer reference used in Marketplace	an..63	X	*000012*
legal_name	Company name	an..63		*Dupond & Co*
title	Buyer's title	an..63		*Ms*
type	Type of buyer	enum	X	*PRIVATE | COMPANY*
first_name	Buyer's first name	an..63		*Mary*
last_name	Buyer's last name	an..63		*SMITH*
phone_number	Buyer's phone number	an..32	X	*0612324565*
email	Buyer's e-mail address	ans..150	X	*m.smith@site.com*
address	Buyer's postal address	Address object		

21.4. Item object

Name	Description	Format	Mandatory	Example
uuid	Unique identifier of the service *Generated by Lyra Collect*	an..36	N/A	*3ea5e574-e198-55d4-ba23-f9405ec4226f*
href	URL to access this resource *Generated by Lyra Collect*	see example	N/A	*https://secure.lyra.com/marketplace/items/3ea5e574-e198-55d4-ba23-f9405ec4226f*
created_at	Date and time of creation of the resource *Generated by Lyra Collect*	see example	N/A	*2015-01-17T09:39:54.948Z*
updated_at	Date and time of the last modification of the resource *Generated by Lyra Collect*	see example	N/A	*2015-01-17T09:39:54.948Z*
seller	UUID of the provider	an..36	X	*1ea5e574-e198-55d4-ba23-f9405ec4226a*
order	UUID of the order	an..36		*8ea5e574-e198-55d4-ba23-f9405ec4226c*
reference	Reference of the item	an..32+"-"	X	*0000000181*
description	Description of the service	an..255		*Blablabla service*
type	Defines the type of the article in the cart.	enum		see table below.
amount	If **is_commission** is set to false: Service amount. If **is_commission** is set to true: The amount of the commission is defined within the order.	n..12	X	*50000*
commission_amount	Commission amount defined for the item.	n..12		*500*
is_commission	Indicates that this item is a commission.	enum		*true | false* *false* by default

Name	Description	Format	Mandatory	Example
status	Status of the item *Generated by Lyra Collect*	an..10	N/A	*SUCCESSFUL*

Type attribute values

Value	Description
FOOD_AND_GROCERY	Food and grocery
AUTOMOTIVE	Cars / Moto
ENTERTAINMENT	Entertainment / Culture
HOME_AND_GARDEN	Home and gardening
HOME_APPLIANCE	Household appliances
AUCTION_AND_GROUP_BUYING	Auctions and group purchasing
FLOWERS_AND_GIFTS|	Flowers and presents
COMPUTER_AND_SOFTWARE	Computers and software
HEALTH_AND_BEAUTY	Health and beauty
SERVICE_FOR_INDIVIDUAL	Services for individuals
SERVICE_FOR_BUSINESS	Services for companies
SPORTS	Sports
CLOTHING_AND_ACCESSORIES	Clothes and accessories
TRAVEL	Travel
HOME_AUDIO_PHOTO_VIDEO	Sound, image and video
TELEPHONY	Telephony

21.5. Marketplace object

Name	Description	Format	Example
uuid	Unique identifier of the Marketplace *Generated by Lyra Collect*	an..36	*6ea5e574-e198-55d4-ba23-f9405ec4226b*
href	URL to access this resource *Generated by Lyra Collect*	see example	*https://secure.lyra.com/marketplace/marketplaces/6ea5e574-e198-55d4-ba23-f9405ec4226b*
created_at	Date and time of creation of the resource *Generated by Lyra Collect*	see example	*2015-01-17T09:39:54.948Z*
updated_at	Date and time of the last modification of the resource *Generated by Lyra Collect*	see example	*2015-01-17T09:39:54.948Z*
reference	Name of the Lyra Collect shop created for the Marketplace	an..255	*Ex: Marketplace Shop name*
description	Name of the Lyra Collect shop created for the Marketplace	an..255	*Ex: Marketplace Shop name*
bic	Bank code of the Marketplace bank account	an..11	*CMCIC1234*
iban	Account number of the Marketplace bank account	an..34	*FR1212341234123412341234*
vads_key	Identifier of the Lyra Collect shop created for the Marketplace *Generated by Lyra Collect*	an..8	*12345678*
vads_cert	Production certificate of the Lyra Collect shop created for the Marketplace *Generated by Lyra Collect*	an..16	*123456789123456789*
webhook_url	Notification URL of the Marketplace	an..1024	*http://marketplace.com/page_traitement_notification.php*
status	Status of the Marketplace resource *Generated by Lyra Collect*	an..10	*ACTIVE*
links	Links to the Sellers and Orders resources of the Marketplace *Generated by Lyra Collect*	see example	`{ "sellers": { "href": "https://secure.lyra.com/marketplace/marketplaces/6ea5e574-e198-55d4-ba23-f9405ec4226b/sellers" }, "orders": { "href": "https://secure.lyra.com/marketplace/marketplaces/6ea5e574-e198-55d4-ba23-f9405ec4226b/orders" } }`

21.6. Order object

Name	Description	Format	Mandatory	Example
uuid	Unique identifier of the order *Generated by Lyra Collect*	an..36	N/A	*8ea5e574-e198-55d4-ba23-f9405ec4226c*
href	URL to access this resource *Generated by Lyra Collect*	see example	N/A	*https://secure.lyra.com/marketplace/orders/8ea5e574-e198-55d4-ba23-f9405ec4226c*
created_at	Date and time of resource creation *Generated by Lyra Collect*	see example	N/A	*2015-01-17T09:39:54.948Z*
updated_at	Date and time of the last resource modification *Generated by Lyra Collect*	see example	N/A	*2015-01-17T09:39:54.948Z*
marketplace	UUID of the Marketplace	an..36	X	*6ea5e574-e198-55d4-ba23-f9405ec4226b*
reference	Order reference	an..32+"-"	X	*CMD-123456*
description	Order description	an..255		*Order 123456 - Margaritas x2 + 4 seasons x1*
vads_transaction_id	Transaction identifier Lyra Collect	n..6	N/A	*379* *Note: Field deprecated here.*
vads_transaction_date	Date and time of the Lyra Collect transaction in the YYYYMMDDHHMMSS format	an..14	N/A	*20150320094512* *Note: Field deprecated here.*
buyer	Buyer details	Buyer object	X	
alias	Token ID in the gateway	ans..64		1144951ea7ab42989c97159b3dfc0382
shipping	Shipping details	Shipping object		
amount	Order total amount, expressed in the smallest currency unit.	n..12	N/A	*50000* for EUR 500.
currency	Currency of the order (ISO 4217)	a3	X	*EUR*
language	Display language of the payment pages (ISO 639-1)	a2		*FR*
capture_delay	Indicates the delay (in days) before the capture at the bank	Positive integer between 0 and 6	No. Only available in creation (POST)	*3*
expected_capture_date	Expected capture date, in UTC timezone, in the YYYYMMDDHHMMSS format.	Date	No. Only available in update (PUT)	*20180716083000*
status	Order status *Generated by Lyra Collect*	an..10	N/A	*PENDING*
webhook_result	Result of the notification sent to the Marketplace *Generated by Lyra Collect*	n..3	N/A	*200*
url_return	Default buyer return URL on the Marketplace website.	an..512		*http://www.sitemarketplace.com/url_de_retour.html*
url_success	Buyer return URL on the Marketplace website after a successfully completed payment.	an..200		*http://www.sitemarketplace.com/url_success.html*
url_refused	Buyer return URL on the Marketplace website after a refused payment.	an..200		*http://www.sitemarketplace.com/url_refused.html*
url_cancel	Buyer return URL on the Marketplace website after a canceled payment.	an..200		*http://www.sitemarketplace.com/url_cancel.html*

Name	Description	Format	Mandatory	Example
url_error	Buyer return URL on the Marketplace website after a payment that resulted in an error.	an..200		*http://www.sitemarketplace.com/url_error.html*
return_mode	Data transmission method used when returning to the merchant website.	a4		*POST*

links	Links to the Items resources
	Generated by Lyra Collect
		see example	N/A	`{`
				`  "items": {`
				`    "href":`
				` "https://secure.lyra.com/marketplace/orders/8ea5e574-e198-55d4-ba23-f9405ec4226c/items"`
				`   }`
				`}`

persist_url	Persistent payment link	url	N/A	"*https://secure.lyra.com/marketplace/t/ioeD1uRP*"
expiry_date	Expiration date of the persistent link	see example	N/A	"2021-02-15T15:37:47.632009Z"
form_token	Token of the embedded form	an		Example: ffa0yfV2FQuR3aak1SwsgcuZg195eyJhbW9OjEwOTAwLCJjdXJyZW5jeSI6IkVVUiIsIm1vUiOiJURVNUIiwidmVyc2lvbiI6Mywib3JkZXJJI6IjVjNDk1N2E5LWNkOGEtNGJhZC1hMGNgxMTZiMWE3ZDdmNiIsInNob3BOYW1lIjoiyYSBTTVMiLCJicmFuZFByaW9yaXR5IjpbIkJTkNPTlRBQ1QiLCJDQiIsIkUtQ0FSVEVCTEVVRSIsIk1BU1RFUkNBUkQiLCVGQVVMVCJ9fX0a702

21.7. Order voucher object

Name	Description	Format	Mandatory	Example
contract_type	Contract (MID) type	“CONECS” or “CVCONNECT”	X	“CONECS”
eligible_amount	Amount eligible in the smallest currency unit	n..12	Mandatory if contract_type = "CONECS"	1900

21.8. Refund object

Name	Description	Format	Mandatory
order	Identifier of the order to refund	an..36	X
reference	Technical reference of the refund	an..32	X
status	Status of refund	an..10	X
description	Description of the refund	ans..255	
currency	Currency of the order (ISO 4217) Only the **EUR** value is authorized	a3	X
items	List of refund parts to be made (by seller)	enum	X
items.seller	Identifier of the seller who must take on this part of the refund	an..36	X
items.reference	Technical reference	an..32	X
items.description	Description	ans..255	
items.amount	Amount of the part to be refunded by the seller	n..12	X

Name	Description	Format	Mandatory
items.item	Reference of the item being refunded within the original order	an..50	

21.9. Seller object

Name	Description	Format	Example
uuid	Unique item identifier of the provider *Generated by Lyra Collect*	an..36	*1ea5e574-e198-55d4-ba23-f9405ec4226a*
href	URL to access this resource *Generated by Lyra Collect*	see example	*https://secure.lyra.com/marketplace/sellers/1ea5e574-e198-55d4-ba23-f9405ec4226a*
created_at	Date and time of creation of the resource *Generated by Lyra Collect*	see example	*2015-01-17T09:39:54.948Z*
updated_at	Date and time of the last modification of the resource *Generated by Lyra Collect*	see example	*2015-01-17T09:39:54.948Z*
marketplace	UUID of the Marketplace	an..36	*6ea5e574-e198-55d4-ba23-f9405ec4226b*
reference	Identifier of the provider's Lyra Collect shop	an..255	*87654321*
description	Identifier of the provider's Lyra Collect shop	an..255	*Provider's shop*
bic	Bank code of the provider's bank account	an..11	*CMCIC1234*
iban	Account number of the provider's bank account	an..34	*FR1212341234123412341234*
is_marketplace_seller	allows to identify the marketplace seller. In the order, commission items (is_commission = true) must be linked to this seller. This seller is automatically created during the marketplace's creation.	boolean	*true* | *false*
status	Status of the Marketplace resource *Generated by Lyra Collect*	an.10	*ACTIVE*
links	Links to the Items resources of the provider *Generated by Lyra Collect*	see example	`{ "items": { "href": "https://secure.lyra.com/marketplace/sellers/1ea5e574-e198-55d4-ba23-f9405ec4226a/items" }, }`

21.10. Shipping object

Name	Description	Format	Mandatory	Example
delivery_company_na	transporter's name	ans..128		
shipping_speed	shipping mode	enum		*STANDARD | EXPRESS | PRIORITY*
shipping_method	shipping method	enum	X	*RECLAIM_IN_SHOP | RELAY_POINT | RECLAIM_IN_STATION | PACKAGE_DELIVERY_COMPANY | ETICKET*
shipping_delay	speed related to delivery mode when shipping_speed = PRIORITY	enum		*INFERIOR_EQUALS | SUPERIOR | IMMEDIATE | ALWAYS*
type	type of recipient	enum		*PRIVATE | COMPANY*
legal_name	company name	an..63		*Dupond & Cie*
name	recipient's name			
first_name	buyer's first name	an..63		*Mary*
last_name	buyer's last name	an..63		*SMITH*
phone_number	buyer's phone number	an..32		*0612324565*
address	buyer's postal address	Address object		

21.11. Token object

Name	Description	Format	Mandatory	Example
uuid	Unique identifier of the order *Generated by Lyra Collect*	an..36	N/A	*c730def9-e171-4f4f-92de-b7f09ed74b8d*
created_at	Date and time of creation of the resource *Generated by Lyra Collect*	see example	N/A	*2018-11-27T13:53:48.461191Z*
updated_at	Date and time of the last modification of the resource *Generated by Lyra Collect*	see example	N/A	*2018-11-27T13:53:48.461191Z*
marketplace	UUID of the Marketplace	an..36	N/A	*2434c0a2-9d46-4e96-9553-1536c898625b*
buyer	Buyer details	Buyer object	X	
language	Display language of the payment pages (ISO 639-1)	a2		*FR*
url_return	Buyer's return URL on the Marketplace website	an..512		*http://www.my-website.com*
status	Request status *Generated by Lyra Collect*	an..10	N/A	*CREATED*
alias	Token ID in the gateway	ans..64		1144951ea7ab42989c97159b3dfc0382
alias_to_update	Token ID to be modified in the gateway	ans..64		1144951ea7ab42989c97159b3dfc0382
payment_url	URL to give to the buyer for the creation of the token	ans..255		

21.12. Transaction object

Name	Description	Format	Mandatory	Example
trans_uuid	transaction identifier	ans..32	X	03934e6df4ec4f8ea0fe491e95fbc619
order	Identifier of the associated order		X	33ec0e5a-f25c-4bb1-8115-e54c70e30f16
sequence_number	Transaction index that can be useful in case of installment payment.	n..12	X	1 for the first transaction, 2 for the second one, etc.
paid_at	Payment date.	Date	No	2020-06-15T13:16:22Z
auto_code	Authorization return code.	a..2	No	See the list of values *below*.
payment_scheme	Description of the used payment method.	an..16	N/A	CB
amount	Transaction amount expressed in the smallest currency unit.	n..12	X	10450 for EUR 104.50
initial_amount	Transaction amount recorded upon the authorization request expressed in the smallest currency unit.	n..12	No	10450 for EUR 104.50
expected_capture_date	Expected capture date, in UTC timezone, in the YYYYMMDDHHMMSS format.	Date	X	20200615151622
status	Transaction status. *Generated by Lyra Collect*	an..10	No	TO_VALIDATE
links	Links to the transaction resources. *Generated by Lyra Collect*	see example	No	"links": { "transfers": { "href": "https://secure.lyra.com/marketplace/transactions/03a0f95fbc619/transfers" } }
created_at	Date and time of resource creation.	see example	No	2020-06-15T13:16:49.223042Z
updated_at	Date and time of the last resource update.	see example	No	

List of authorization return codes (auto_code):

Codes returned by **Amex Global** acquirer:

Code	Description
000	Approved
001	Approved with an ID
002	Partial approval (Prepaid Cards only)
100	Declined
101	Expired card / Invalid expiry date
106	Exceeded PIN entry attempts
107	Please Call Issuer
109	Invalid merchant
110	Incorrect Transaction Amount
111	Invalid account / Invalid MICR (Travelers Cheque)
115	Requested function not supported
117	Invalid PIN
119	Cardholder not enrolled / not allowed
122	Invalid card security code (a.k.a., CID, 4DBC, 4CSC)
125	Invalid effective date
130	Declined
181	Format error
183	Invalid currency code
187	Deny - New card issued
189	Deny - Account canceled
200	Deny - Pick up card
900	Accepted - ATC Synchronization
909	System malfunction (cryptographic error)
912	Issuer not available

Codes returned by the **CB** and **Paylib** network:

Value	Description	Value	Description
00	Approved or successfully processed transaction	**43**	Stolen card
02	Contact the card issuer	**51**	Insufficient balance or exceeded credit limit
03	Invalid acceptor	**54**	Expired card
04	Keep the card	**55**	Incorrect secret code
05	Do not honor	**56**	Card absent from the file
07	Keep the card, special conditions	**57**	Transaction not allowed for this cardholder
08	Confirm after identification	**58**	Transaction not allowed for this cardholder
12	Incorrect Transaction Code	**59**	Suspected fraud
13	Incorrect Transaction Amount	**60**	The acceptor of the card must contact the acquirer
14	Invalid cardholder number	**61**	Withdrawal limit exceeded
15	Unknown issuer	**63**	Security rules unfulfilled
17	Canceled by the buyer	**68**	Response not received or received too late
19	Retry later	**75**	Number of attempts for entering the secret code has been exceeded
20	Incorrect response (error on the domain server)	**76**	The cardholder is already blocked, the previous record has been saved
24	Unsupported file update	**80**	Contactless payment is not accepted by the issuer
25	Unable to locate the registered elements in the file	**81**	Unsecured payment is not accepted by the issuer
26	Duplicate registration, the previous record has been replaced	**82**	Revocation of recurring payment for the card of a specific Merchant or for the MCC and the card
27	File update edit error	**83**	Revocation of all recurring payments for the card
28	Denied access to file	**90**	Temporary shutdown

Value	Description	Value	Description
29	Unable to update	**91**	Unable to reach the card issuer
30	Format error	**94**	Duplicate transaction
31	Unknown acquirer company ID	**96**	System malfunction
33	Expired card	**97**	Overall monitoring timeout
34	Suspected fraud	**98**	Server not available, new network route requested
38	Expired card	**99**	Initiator domain incident
41	Lost card		

Codes returned by the **GICC** network:

Code	Description
0	Approved or completed successfully
2	Call Voice-authorization number; Initialization Data
3	Invalid merchant number
4	Retain card
5	Authorization declined
10	Partial approval
12	Invalid transaction
13	Invalid amount
14	Invalid card
21	No action taken
30	Format Error
33	Card expired
34	Suspicion of manipulation
40	Requested function not supported
43	Stolen card, pick up
55	Incorrect personal identification number
56	Card not in authorizer's database
58	Terminal ID unknown
62	Restricted card
78	Stop payment order
79	Revocation of authorization order
80	Amount no longer available
81	Message-flow error
91	Card issuer temporarily not reachable
92	The card type is not processed by the authorization center
96	Processing temporarily not possible
97	Security breach - MAC check indicates error condition
98	Date and time not plausible
99	Error in PAC encryption detected

Codes returned by the **PayPal** network:

Code	Description
0	Transaction accepted
10001	Internal error
10002	Restricted Account
10009	Transaction refused for one of the following reasons: • The partial refund amount must be less than or equal to the original transaction amount. • The partial refund must be in the same currency as the original transaction. • This transaction has already been fully refunded. • The time limit (60 days) for performing a refund for this transaction has been exceeded.
10422	Customer must choose new funding sources. The customer must return to PayPal to select new funding sources.

Code	Description
10486	This transaction couldn't be completed. Please redirect your customer to PayPal.
13113	The Buyer cannot pay with PayPal for this transaction. Inform the buyer that PayPal declined the transaction and to contact *PayPal Customer Service*.

Codes returned by **Edenred Belgium** acquirer

Code	Description
0	Payment accepted
1	Partially approved The buyer has been prompted to pay the remaining amount with another payment method.
5	Payment refused
102	Invalid card number
103	Retailer not authorized
104	Blocked card
105	Insufficient funds
106	Expired card
107	Incorrect pin
108	Unreferenced error
114	The card has not yet been activated
115	Balance over the max limit
116	Invalid currency
117	Daily spend limit exceeded
118	Weekly spend limit exceeded
119	Monthly spend limit exceeded
120	Yearly spend limit exceeded
121	Wrong PIN was provided too many times
122	Transaction amount too small
123	Transaction amount too big
124	Account blocked
125	Transaction amount limit exceeded
126	Transaction velocity limit exceeded (Spend limit exceeded)

22. OBTAINING HELP

Looking for help? Check our FAQ on our website

https://docs.lyra.com/en/collect/faq/sitemap.html

For any technical inquiries or if you need any help, contact *technical support*.

To help us process your demands, you will be asked to communicate your customer code (e.g.: **CLXXXXX**, **MKXXXXX** or **AGXXXXX**).

This information is available in the Merchant Back Office (top of menu).