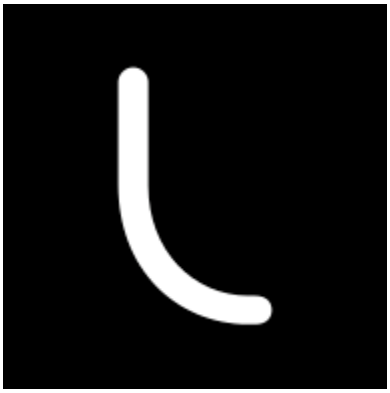
LUNU INTEGRATION DOCUMENTATION

SFRA v22.1.0



Summary

1. Prerequisites

- Salesforce B2C Commerce v.21.X
- An implementation of SFRA v.6
- Wallet for testing
- Environments
    - **testing.lunu.io** - sandbox testing server
    - **alpha.lunu.io** - production server

2. Lunu payment

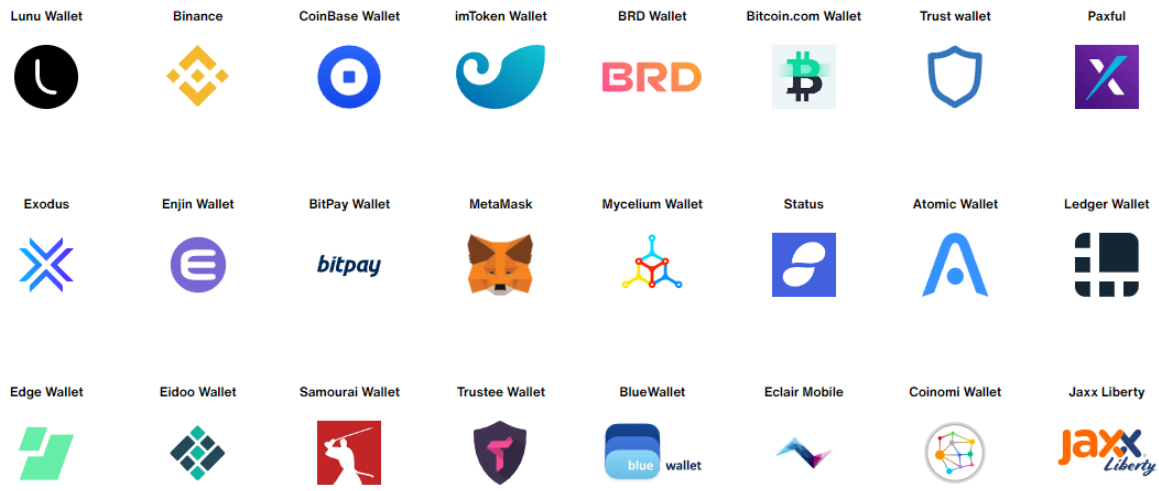
Lunu represents a crypto payment gateway that functions with any software wallet that recognizes or supports the input of a cryptocurrency address and transaction amount. Lunu uses the EIP-67 format of generating QR-Codes for Ethereum and BIP-21 for Bitcoin.

Failover/Recovery Process:

In case of failure, please contact: support@lunu.io or go to: https://support.lunu.io/
Provide as much information as possible, and make sure to check the logs.
The FAQ for Lunu is located at: https://faq.lunu.io/

Recommended wallets:

The full list can be found here: https://lunu.io/recommended-wallets



Supported countries:

As Lunu is a crypto payment service, there should be no restrictions on locales for payment. However, some Lunu products and services are not supported in every country. The list of supported countries is growing, and the plans for the supported country expansions can be found here: https://lunu.io/countries

Austria	France	Latvia	Netherlands
Estonia	Greece	Lithuania	Slovenia
Finland	Italy	Malta	Spain

Coming soon

Andorra	Germany	Luxembourg	Slovakia
Australia	Japan	Monaco	United Kingdom
Belgium	Korea	Portugal	USA

3. Supported coins

The list can be found here: https://lunu.io/coins

Bitcoin (BTC)
Networks: BTC, LNX

Ethereum (ETH)
Networks: ETH

USDT
Networks: ETH, BSC, TRX

USDC
Networks: ETH, BSC

BNB
Networks: BSC

Tron (TRX)
Networks: TRX

Ripple (XRP)
Networks: XRP

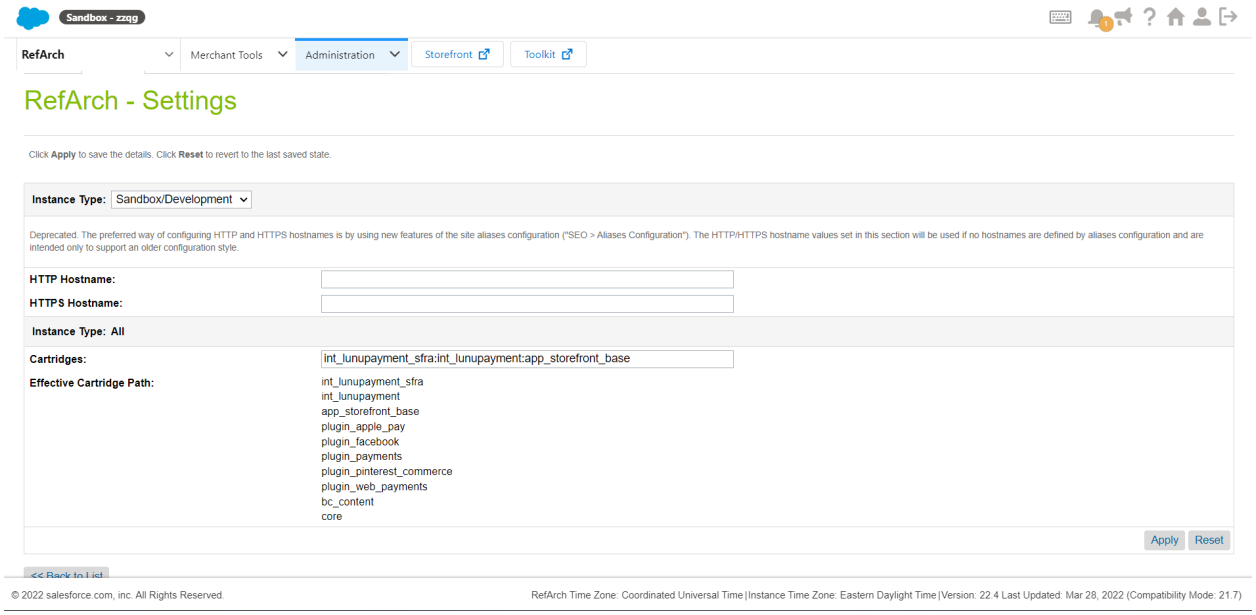
4. Install the cartridge

Lunupayment provides a LINK cartridge to integrate with Salesforce Commerce Cloud (SFCC). This cartridge enables the Lunu payment service for Salesforce Commerce Cloud SFRA.

1. Open the provided LINK_lunupayment.zip package
2. Extract the LINK_lunupayment folder which contains two folders with the following cartridges:
    a int_lunupayment
    b. Int_lunupayment_sfra
    c. bm_lunupayment
3. Add int_lunupayment, int_lunupayment_sfra and bm_lunupayment cartridges into your SFRA's code
4. Upload all cartridges into your SFRA sandbox

5. Setup cartridge path

1. Select Administration > Sites > Manage Sites
2. Open the site where you like to install the cartridge and select "Settings" tab
3. Add int_lunupayment_sfra:int_lunupayment followed by a colon the Cartridges textbox.
4. Click Apply



Besides setting these cartridges, add bm_lunupayment cartridges to to BM path.

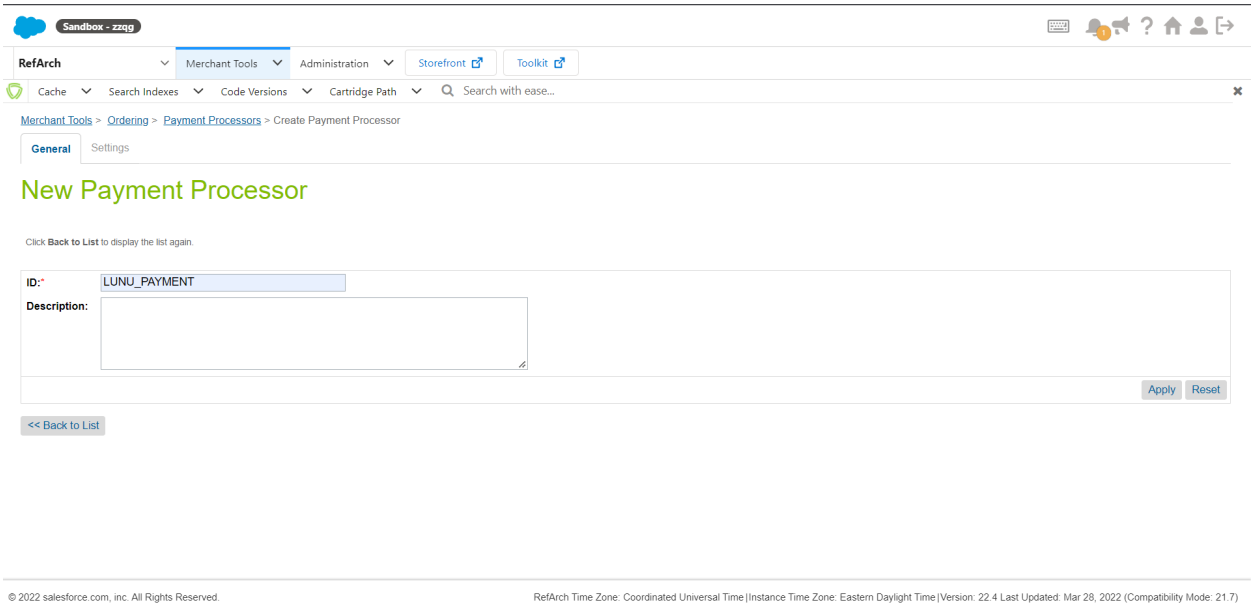
6. Import metadata

1. Select Administration > Site Development > Site Import & Export

2. In the File section, click Upload and select the zipped folder "metadata"
3. When mentioned folder is successfully uploaded, select it and click Import button
4. Make sure the import process was successful

7. Setup payment method

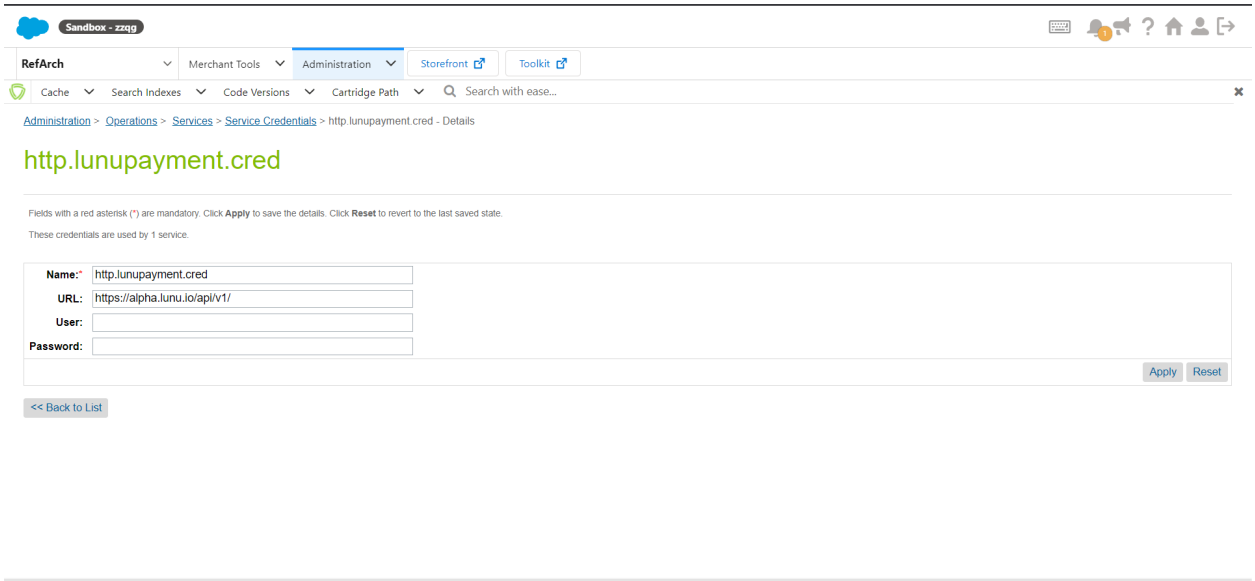
1. Select Merchant Tools > Ordering > Payment Processors
2. Click New to create a new payment processor with name: LUNU_PAYMENT
3. Click Apply



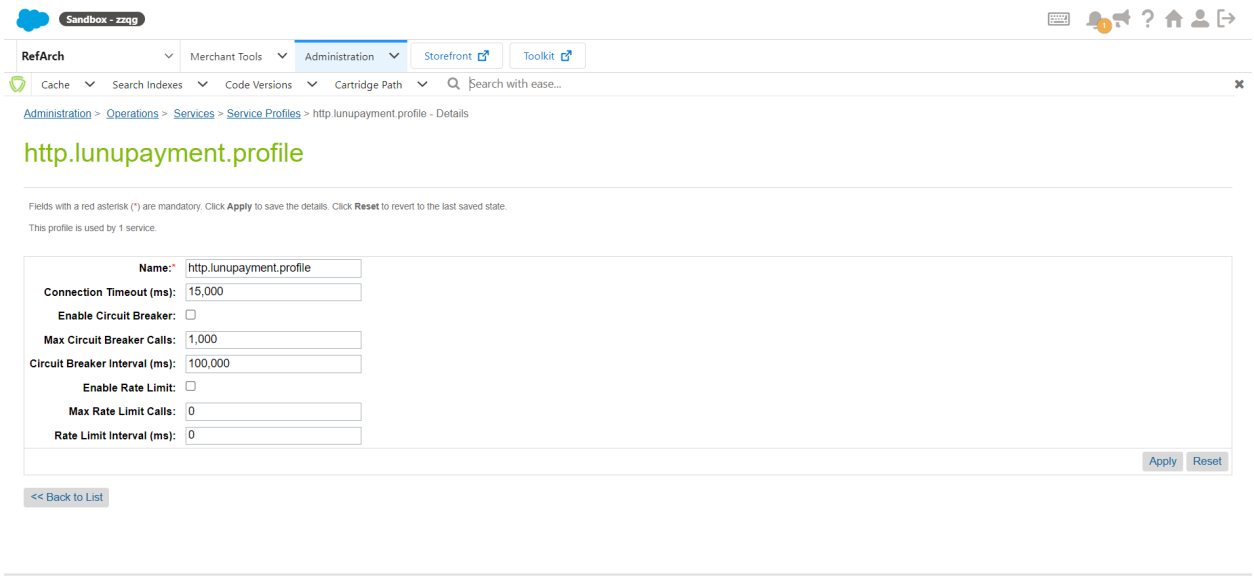
Since we used Site Import and Export for importing necessary data, payment method should be already created.

8. Web Services

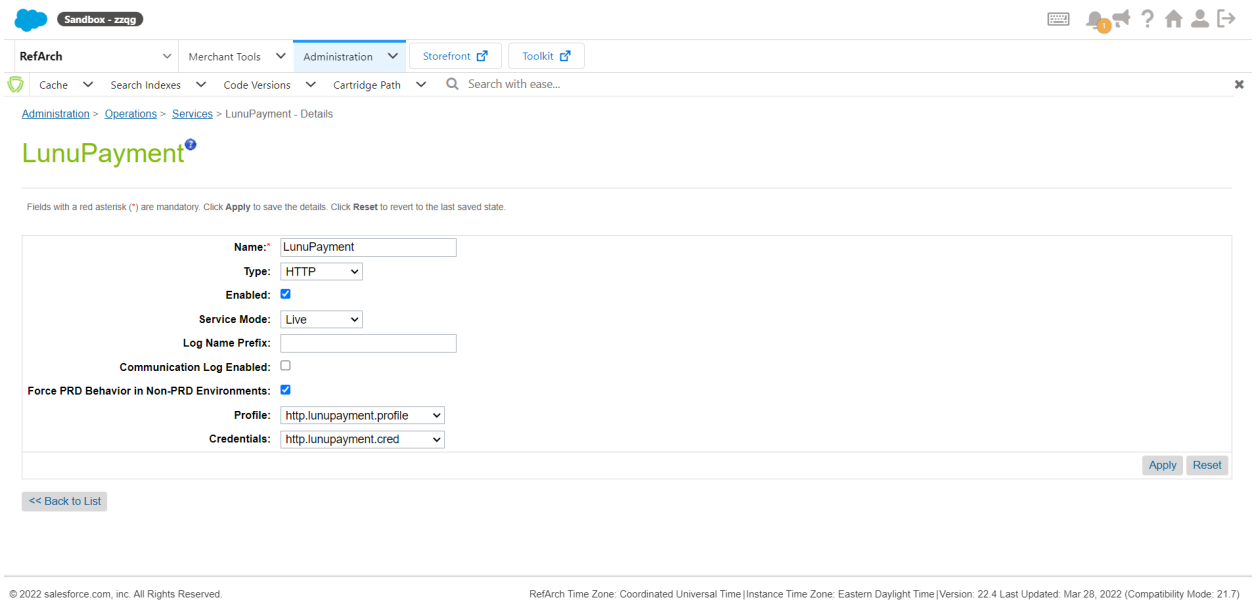
Same goes for Web services. Because of the way we imported necessary files, services should be already in place. Example of service credentials:



Example of service profile:

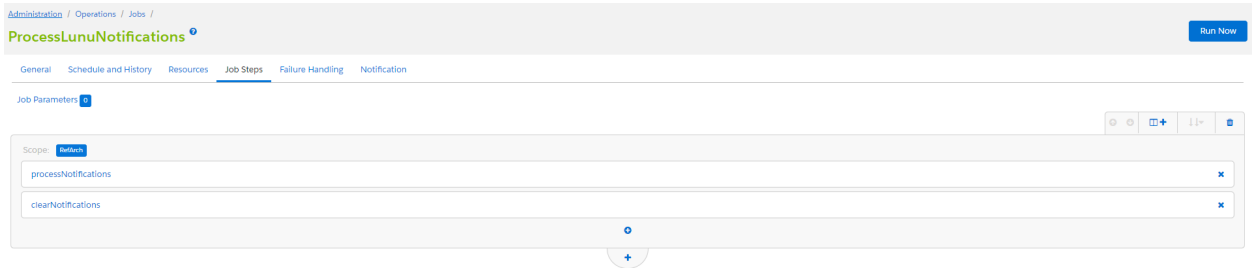


And lastly example of the actual service:



9. Setup Jobs

A job is created in order to process Lunu payment notifications and clear the processed ones. Please configure the job as described in jobs.xml. Please keep in mind that scheduling can be changed according to business requirements and depending on how quickly Lunu processes the payments. In the picture bellow you can see how the job is configured

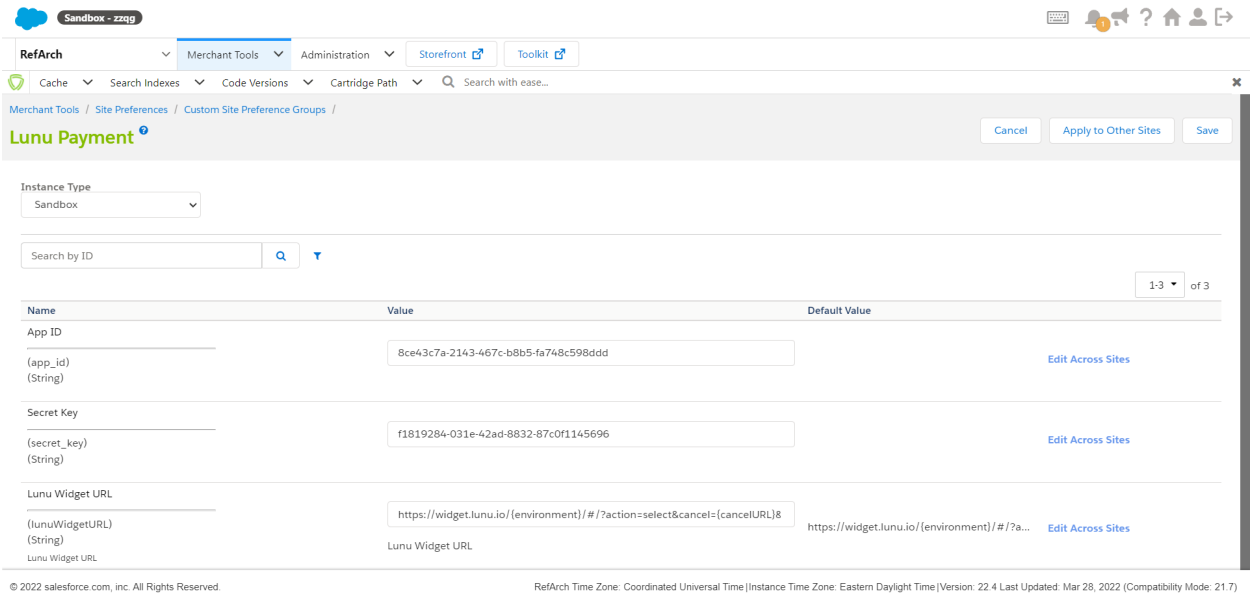


10. Setup Site Preferences

When it comes to custom preferences, in order to make Lunu payment work we use only a couple of them. We need:

- LunuAppID
- LunuSecretKey
- LunuWidgetURL

Lunu provides the LunuAppID and LunuSecretKey (https://console.lunu.io), whereas the LunuWidgetURL should remain unchanged.



11. SFRA code changes

11.1. Template changes

According to the merchant's customizations, the LINK_lunupayment cartridge requires to update your **paymentOptionsContent.isml**

```
<isloop items="${pdict.order.billing.payment.applicablePaymentMethods}" var="paymentOption">
    <isif condition="${paymentOption.ID === 'CREDIT_CARD'}">
        <isinclude template="checkout/billing/paymentOptions/creditCardContent" />
    <iselseif condition="${paymentOption.ID === 'LUNU'}">
        <isinclude template="checkout/billing/paymentOptions/lunuPaymentContent" />
    </isif>
</isloop>
```

According to the merchant's customizations, the LINK_lunupayment cartridge requires to update your **paymentOptionsSummary.isml**

```
<div class="payment-details">
    <isloop items="${pdict.order.billing.payment.selectedPaymentInstruments}" var="payment">
        <isif condition="${payment.paymentMethod === 'CREDIT_CARD'}">
            <isinclude template="checkout/billing/paymentOptions/creditCardSummary" />
        <iselseif condition="${payment.paymentMethod === 'LUNU'}">
            <isinclude template="checkout/billing/paymentOptions/lunuPaymentSummary" />
        </isif>
    </isloop>
</div>
```

According to the merchant's customizations, the LINK_lunupayment cartridge requires to update your **paymentOptionsTab.isml**

```
<isloop items="${pdict.order.billing.payment.applicablePaymentMethods}" var="paymentOption">
    <isif condition="${paymentOption.ID === 'CREDIT_CARD'}">
        <isinclude template="checkout/billing/paymentOptions/creditCardTab" />
    <iselseif condition="${paymentOption.ID === 'LUNU'}">
        <isinclude template="checkout/billing/paymentOptions/lunuPaymentTab" />
    </isif>
</isloop>
```

11.2. Controller changes

Controllers changed in SFRA:

- CheckoutServices-PlaceOrder - Extended to check session for current paymentMethod, if that paymentMethod is LUNU, we are getting confirmationToken, after which we are also getting widget URL from custom preference we imported earlier. When everything is in place properly we enrich widget URL with necessary success and fail URLs, those URLs will handle two potential states after redirection back to our site.

- Order-Confirm - Replaced to fetch orderNo and orderToken from the request form or querystring parameters.

It is important to mention that the processNotification job step is updated to fill in all needs required after paying with the Lunu widget, meaning setting appropriate statuses or orders.

12. Custom Objects

The payment status for Lunu can not be verified during the payment itself. The “lunuPaymentNotification” custom object is used for storing responses that Lunu sends every time the payment status is changed on their server.

The notifications are identified by a transaction ID, which is contained within a response from Lunu web service during payment creation. If a notification with the same ID already exists, the notification will be updated with new data.

A job is used to process the notifications, and only if the notification’s status is “paid” (and the notificationStatus is “PENDING”), the notification is processed. Once processed, the notifications are removed on the next step of the job which removes them. The rate at which the notifications are processed can be configured in the job through scheduling.

Retention time for the notifications is 3 days, which is based on experience with other 3rd party payment providers and best practices.

The notification is created with every Lunu payment, only if the Lunu payment status is “awaiting_payment_confirmation” or “paid”. For other statuses that indicate a failed, expired or canceled payment, the notification is not created, and the basket is reopened, if possible. As discussed, the notification with the same transaction ID will only be updated with new data, if the status transitions from “awaiting_payment_confirmation” to “paid”, so the rate of creating new notifications has been decreased. That means that a notification will be created per order (only those which use Lunu for payment).

13. Integration tests

In order to test the integration, a test for placing an order is included into the repository. It covers the whole checkout process with the Lunu payment method selected. Before the test is executed, the base url should be set to the correct instance in file test/integration/it.config.js. Correct format is given in the file.