Fundbox Pay API Integration

Overview

This guide describes how to integrate Fundbox Pay into your e-commerce checkout or payment portal, so you can *provide Fundbox Pay as a payment option* to your business customers.

The Checkout API is organized around REST and defined using Swagger (aka OpenAPI) API modeling specification. The API is hosted on SwaggerHub through which you may download API clients consuming the API in a wide variety of programming languages. Our API has predictable, resource-oriented URLs, and uses HTTP response codes to indicate API errors. We use built-in HTTP features, JSON is returned by all API responses, including errors, although our API Clients, available from our SwaggerHub repository, convert responses to appropriate language-specific objects.

Sandbox development

Accounts have test (sandbox) mode and live mode API keys. To change between mode, use the appropriate key and base urls to perform a live or test transaction. Requests made with test mode credentials are performed at no cost and do not trigger any financial transactions.

Develop and test the Fundbox Pay integration in your development environment connected to our sandbox.

Contact your Success Manager at Fundbox Pay to receive an email inviting you to Fundbox Pay's Swagger API.
After development and testing, you will need to update your integration with live API keys and change the API base url to production.

On-boarding customers:

Customers can onboard to Fundbox Pay in the flow, or you can invite your customer to register via your Fundbox Pay unique URL - find it in the invite tab in the Fundbox Pay dashboard and add '/checkout' postfix to the URL.

Once the customers are registered, they can log in to the checkout flow.
For the purpose of development the integration against the sandbox - we will provide you customer credentials you can complete the checkout flow with.
On-boarding customers in the sandbox environment is not supported at this time.

The transaction flow with Fundbox Pay:

1. Client side Merchant store - The buyer begins checkout from the shopping cart.
2. Client side Merchant store - The merchant creates a Fundbox Pay checkout javascript handler object by passing it's public key and an onComplete callback function to be invoked upon checkout modal completion.
3. Client side Merchant store -The merchant creates an order object detailing cart contents and prices.
4. Client side Merchant store -The buyer selects the Fundbox Pay payment method option.
5. Client side Merchant store -The buyer clicks on the place order button .
6. Client side Merchant store - The merchant opens the Fundbox checkout modal, by invoking the checkout handler .open method, passing it the order object.
7. Client side Fundbox checkout modal - The buyer is presented with the Fundbox Pay modal.
8. Client side Fundbox checkout modal -The buyer signs in or creates a Fundbox Pay account.
9. Client side Fundbox checkout modal -The buyer confirms payment terms.
10. Client side Fundbox checkout modal - The Fundbox Pay modal returns the a unique order token back to the merchant store client side.
11. Client side Merchant store - The callback configured in step 2 is invoked with the unique order token as an argument.
12. Client side Merchant store - The callback should complete the checkout, submitting the page including the unique order token to the Merchant store backend.
    a. Apply (Capture):
        i. Server side Merchant store - The order token and transaction amount is sent by the merchant to the Fundbox Pay apply order API call (Capture).
        ii. Server side Fundbox Pay - Fundbox Pay API validates the order token and amount beginning the process of crediting the merchant account.
        iii. Server side Fundbox Pay - The merchant redirects the customer to the order confirmation page.
    b. Auth and then Capture
        i. Server side Merchant store - The order token and transaction amount is sent by the merchant to the Fundbox Pay authorize order API call (Authorize).
        ii. Server side Fundbox Pay - Fundbox Pay API validates the order token and amount and holds the buyer credit limit for the purchase amount.

iii. Server side Fundbox Pay - The merchant redirects the customer to the order confirmation page.

iv. Server side Merchant store - when the order is shipped/invoice is created - The order token and transaction amount is sent by the merchant to the Fundbox Pay apply order API call (Capture).

v. Server side Fundbox Pay - Fundbox Pay API validates the order token and amount beginning the process of crediting the merchant account.

13. The order is complete.

Any actions after the payment is captured should be done in sync with your Customer Success Manager at Fundbox Pay.

Checkout Flow Integration steps

1 - Add checkout.js

Fundbox checkout.js is available at https://checkout.fundboxpay.com/static/merchant/checkout.js

When the checkout page loads, initiate the FbxCheckout handler object using **FbxCheckout.configure()** passing it a configuration object containing:

- **fbxKey:** your Fundbox public key
- **onComplete:** a callback function, this function will be called upon a successful checkout through Fundobx and will receive the fbxOrderToken.
  The fbxOrderToken needs to be sent to your backend server for completing the transaction by capturing the payment calling the **orders/{ fbx_order_token}/apply** API endpoint.

Production Environment Configuration

```
var fbxHandler = FbxCheckout.configure({
  fbxKey: 'pk_prod_.....',
  onComplete: function (fbxOrderToken) {
    // Token to be submitted to the backend for seller approval. this token maps to the 'order'.
  }
});
```

For **testing** purposes, you need to include an **env** variable so that all requests are directed to our testing environment.

- remove the **env** variable when you are ready to go live with your production API keys

Testing Environment Configuration

```
var fbxHandler = FbxCheckout.configure({
  fbxKey: 'pk_test_.....',
  env: 'https://checkout-integration.fundboxpay.com',
  onComplete: function (fbxOrderToken) {
    // Token to be submitted to the backend for seller approval. this token maps to the 'order'.
  }
});
```

2 - Initiate a checkout

Once you have the handler initiated, you call it's open() method that will initiate the Fundbox financing flow.
the **.open** method receives the following parameters:

Order

Key	Type	Required/Optional	Example

orderDetails
Note:
Object that describes current order
	OrderDetails	**Required**	{
			amount_cents: “100500”, shipping_amount_cents: “500”,
			
			checkout_items: [
			{
			name: "Toy",
			sku: "SKU-XYZ", description: "some description",
			amount_cents: “100000”,
			item_amount_cents: “100000”,
			quantity: 1
			}
			]
			}

ctaType **Note:** Defines a text that will appear on CTA button	String	**Optional** **(default: "confirm-terms")**	"confirm-and-pay" / "confirm-terms"

OrderDetails

Key	Type	Required/Optional	Example
amount_cents **Note**: The purchase amount in cents. this amount will be validated to match the items amounts + shipping. This amount will be used as the financing amount for your customers.	string	**Required**	"100050"
transaction_type **Note:** Signifies whether the transaction is of type Capture or Authorize. Affects the checkout modal language.	string	**Optional** **(default**: "**capture")**	"capture" / "authorize"
shipping_amount_cents **Note**: The shipping amount in cents	string	**Required (can be 0)**	"1050"
amount_cents_before_discount **Note:** In case you apply a discount, or take a partial payment in another form, on the checkout cart - you may specify the amount before discount using this property. If you do provide an amount_cents_before_discount - this amount will be validated to match the items amounts + shipping instead of the amount_cents property. The actual financing amount as will be displayed in the UI, is still dictated by the amount_cents property.	string	**Optional**	"100150"
`description` **Note**: The order description, it will be shown in the buyer dashboard.	unicode-string	**Optional**	"Order description"

customer_token **Note:** see Customers API section	string (uuid4)	**Optional**	5bf5908e-93f2-477b-8c8c-c393a926ce98
external_order_identifier **Note:** free text identifier for order	unicode-string	**Optional**	"Toys4Dogs"
shipping_date **Note:** in case the shipping date known , it can be added to the order details If you do provide shipping_date - this date will be validated to match the date format - DD/MM/YYYY	date	**Optional**	"29/07/2017"

checkout_items
Note:
An optional list of ordered items each with
	list[**OrderItem**]	**Required**	`[{`
			`name: "Toy",`
			`sku: "SKU-XYZ",`
			`description: "some description",`
			`total_amount_cents: "100000",`
			`item_amount_cents: "100000",`
			`quantity: 1`
			`}]`

OrderItem

Key	Type	Required/Optional	Example
`name` **Note**: The item name	unicode-string	**Required**	"Toy"
`sku` **Note**: The item stock keeping unit number	unicode-string	**Optional**	"SKU-XYZ"
`description` **Note**: The item store description	unicode-string	**Optional**	"some item description"
total_amount_cents **Note**: The items amount in cents	string	**Required**	1000
item_amount_cents **Note**: The items amount in cents	string	**Required**	1000
`quantity` **Note**: The number of items	integer	**Required**	1

Example, binding the checkoutHandler .open method to place order button:

```
document.getElementById('place_order').addEventListener('click',
function (e) {
  // Open Checkout with further options:
       fbxHandler.open({
   ctaType: "confirm-and-pay",
         orderDetails: {
            amount_cents: "100500",
            shipping_amount_cents: "500",
            checkout_items: [
              {
                 name: "Toy",
                 sku: "SKU-XYZ",
                 description: "some description",
                 total_amount_cents: "100000",
                 item_amount_cents: "100000",
                 quantity: 1
              }
            ]
         }
      });
      e.preventDefault();
    });
```

Closing the Fundbox checkout:

If you need to abort the Checkout process—for example, when navigation occurs in a single-page application, call `close()` on the handler.

```
window.addEventListener('popstate', function() {
  fbxHandler.close();
});
```

3 - Capture a charge for the order

Complete API docs are available in our SwaggerHub repository Please request access from your Customer Success Manager at Fundbox Pay.

Our backend API specifications are available through SwaggerHub, please provide us an email to invite you viewing the API.

Swagger API URL: https://app.swaggerhub.com/apis/Fundbox2/Checkout/1.0.0#/

API base url: https://checkout.fundboxpay.com/api/v1

General notes:

The API uses basic HTTP authentication with:

- **username**: your Fundbox provided public key.
- **password:** your Fundbox provided private key.

Capturing via API:

After posting the fbx_transaction_token, returned from the checkout modal, to the merchant store server it can be used for Capturing charges.

Api endpoint - api/v1/orders/{fbx_order_token}/apply

curl example:

Testing Environment CURL

```
curl
https://checkout-integration.fundboxpay.com/api/v1/orders/{fbx_transaction_token}/apply
     -X POST
     -u "<public_api_key>:<private_api_key>"
     -H "Content-Type: application/json"
     -d "{\"amount_cents\": \"10050\"}"
```

Response:

A successful response is http 200 status code.

4 - Authorize a charge for the order

Authorize via API:

After posting the fbx_transaction_token, returned from the checkout modal, to the merchant store server it can be used for Authorizing charges.

Api endpoint - api/v1/orders/{fbx_order_token}/authorize

curl example:

```
curl
https://checkout-integration.fundboxpay.com/api/v1/orders/{fbx_transaction_token}/authorize
     -X POST
     -u "<public_api_key>:<private_api_key>"
     -H "Content-Type: application/json"
     -d "{\"amount_cents\": \"10050\"}"
```

Response:

A successful response is http 200 status code.

Capturing an authorized transaction:

After a transaction has been authorized, you may capture it calling the /apply api endpoint. Any amount captured up to the originally authorized amount would succeed immediately.

Capturing amount higher then the originally authorized amount

If you attempt to capture an amount which is higher then the original authorized amount, your buyer will be notified of the amount adjustment via email and he will need to explicitly approve it via dashboard. Your system can be notified on the buyer approval or decline via webhooks.

5 - Create your order management functions

Integrating order API actions into your the order management system where you fulfill orders, and process payments.

Canceling order via API:

You may cancel an order via the API if you are unable to fulfill at time of checkout - meaning before order was captured or if the apply (capture) API call returned a non 200 response code.

Api endpoint - api/v1/orders/{fbx_order_token}/cancel

```
curl
https://checkout-integration.fundboxpay.com/api/v1/orders/{fbx_transaction_token}/cancel
     -X POST
     -u "<public_api_key>:<private_api_key>"
     -H "Content-Type: application/json"
```

7 - Test your integration

After setting up the extension, we recommend testing your checkout flow using the Fundbox Pay payment option.

We will provide you with customer credentials, email and password, through which you can complete the checkout process using Fundbox Pay when then extension is configured for ‘Test’ Environment. You may also use the credentials to login to Fundbox Pay customer dashboard on you integration environment at https://integration.fundboxpay.com/login.

Suggested scenarios to implement and verify when testing the integration:

- Complete a simple checkout with the Fundbox Pay payment method option, via the demo buyer credentials supplied.
    - Verify that the checkout modal opens upon hitting the place order button.
    - Validate the order amount total in the checkout page matches the amount presented in the fundbox modal.
    - Verify that completing the modal successfully places an order in your checkout page via the onComplete callback you’ve implemented.
    - Apply (Capture) a charge for the order via the apply api endpoint.
- If you offer coupons/discounts/internal credit lines, repeat test above applying the various propositions which may alter the checkout behaviour.
- Test Canceling an order, in case your store backend is unable to fulfill it at checkout or it encounters any errors during the checkout processing.

8 - Deploy to production

Deploying to production simply requires switching the API keys used from the test API keys to the production API keys and dropping the 'env' property in the checkout.js configure method and switching the API url to production (https://checkout.fundboxpay.com/api/v1).

Reporting API

Transactions report

General:

Reporting endpoints will reveal an API to download data tables in different formats. Currently, we provide an ability to download the data table as:

1. **CSV** - by passing `accept:` **`text/csv`** a client defines that it wants to receive a data table in CSV format
2. **JSON** - by passing `accept:` **`application/json`** a client defines that it wants to receive a data table in JSON format

The report returns the following fields:

amount (The transaction amount)
created_at (format: utc date example: July 29, 2018)
customer (customer token associated with the customer, see customers API).
description (The order description)
seller_fees (fee amount paid by the seller)
t**ransaction_token** (the token which identifies the order, the one passed to the apply/auth endpoints)

Authorize via API:

You have to authenticate yourself as a merchant with public and private keys using basic authentication.

Endpoints:

Transactions history data table endpoint - **/api/v1/transactions**

CURL example (JSON):

```
curl -X GET
"https://checkout-integration.fundboxpay.com/api/v1/transactions" -H
"accept: application/json"
```

CURL example (XML):

```
curl -X GET
"https://checkout-integration.fundboxpay.com/api/v1/transactions" -H
"accept: text/csv"
```

Response:

A successful response is http 200 status code and a body with the report data.

Customers API

Tracking customer status with Fundbox

The checkout integration allows you to track the registration status and availability of your customers with Fundbox.
There are 2 steps to implement customer API:

- Provide a unique customer token in a UUID4 format along with the OrderDetails object when opening the Fundbox checkout modal.
- Query the customers API (v2/customers/{customer_token}/availability) for the customer status, the status returned is one of the following (see swagger for full API documentation):
  - **not_available** - customer has not registered with Fundbox.
  - **unfinished_onboarding** - Customer has started his registration with Fundbox, but has not completed his on-boarding.
  - **approved** - customer is approved for Fundbox credit.
  - **rejected** - customer was rejected for Fundbox credit.
  - **disabled** - customer credit is currently disabled/not available.

Migrating existing customers, and migrating from previous version of customers API

Migration existing customer - if you provide a customer_token in the OrderDetails object, it will set and replace existing customer token once the customer logs in to his Fundbox account via the checkout modal, allowing you to start querying for his Fundbox status via the customers API.

If you've used previous version of the customers API, the API is deprecated but will continue to function, and you can query the new API with any existing customer tokens.
The new API presents a more stable solution - as it allows the seller to generate the customer token versus relying on customer registration email to retrieve it.
We suggest any existing seller to start using the new customers API.

Platform Solution - On-Boarding sellers using oAuth2

Marketplaces and saas platforms can use the Checkout oAuth API to onboard sellers at the click of a button, by having sellers connect and authorize access to their Fundbox Pay account, programmatically sharing API credentials with the the platform. Once credentials are shared in the oAuth flow - the platform can present the checkout integration and do API calls on the sellers behalf.

oAuth terminology:

client_id - an id identifying your platform/application, will be provided by Fundbox.

client_secret - the platform secret key, will be provided by Fundbox.

authorization_code - an authorization code represents the sellers consent to connect his Fundbox account with your platform. The authorization code is created upon seller consent, sent to your platform (see **redirect_url**), and exchanged for an access token.

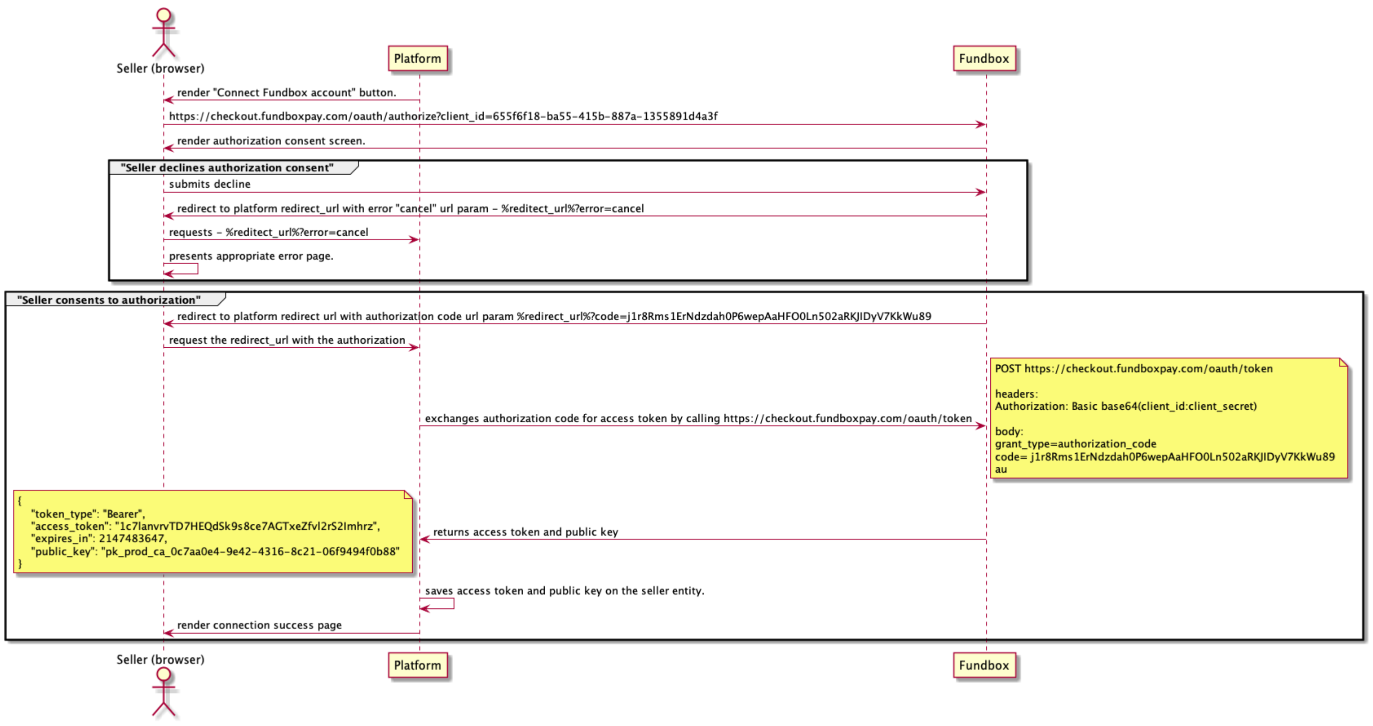
access_token - the access token allows your platform to make requests on behalf of the sellers account. Use this as you would with the **seller private key** in the checkout documentation.

redirect_url - a url in your application that receives the authorization_code and exchanges it for an access token. Supply this url to Fundbox rep prior to integration.

oAuth connection high level flow:

- Your seller register and on-boards with Fundbox.
- Your platform exposes a 'Connect Fundbox Account' button which sends the seller to Fundbox website to login and authorize the platform to do actions on their behalf.
- Once the seller authorizes, Fundbox sends them back to your platform ***redirect_url*** address with the **authorization_code** in the query params.
- Your application sends a POST request with the authorization code to Fundbox's OAuth token endpoint to complete the connection.

oAuth flow sequence diagram:


Seller (browser)
Platform
Fundbox
"Seller declines authorization consent"
"Seller consents to authorization"

render "Connect Fundbox account" button.

https://checkout.fundboxpay.com/oauth/authorize?client_id=655f6f18-ba55-415b-887a-1355891d4a3f

render authorization consent screen.

submits decline

redirect to platform redirect_url with error "cancel" url param - %reditect_url%?error=cancel

requests - %reditect_url%?error=cancel

presents appropriate error page.

redirect to platform redirect url with authorization code url param %redirect_url%?code=j1r8Rms1ErNdzdah0P6wepAaHFO0Ln502aRKJIDyV7KkWu89

request the redirect_url with the authorization

exchanges authorization code for access token by calling https://checkout.fundboxpay.com/oauth/token

```
POST https://checkout.fundboxpay.com/oauth/token

headers:
Authorization: Basic base64(client_id:client_secret)

body:
grant_type=authorization_code
code= j1r8Rms1ErNdzdah0P6wepAaHFO0Ln502aRKJIDyV7KkWu89
au
```

returns access token and public key

```
{
    "token_type": "Bearer",
    "access_token": "1c7lanvrvTD7HEQdSk9s8ce7AGTxeZfvl2rS2Imhrz",
    "expires_in": 2147483647,
    "public_key": "pk_prod_ca_0c7aa0e4-9e42-4316-8c21-06f9494f0b88"
}
```

saves access token and public key on the seller entity.

render connection success page

API description:

GET https://checkout.fundboxpay.com/oauth/authorize

This endpoint sends the user to Fundbox to connect to your platform.
In testing environment replace checkout.fundbox.com with checkout-integration.fbpay.im.

Request:

Parameter	Description
client_id	The unique identifier of your platform/application. will be provided by Fundbox

Response:

The seller's browser is redirected back to your configured redirect url (provide this url to Fundbox prior to integration).
When successful your should receive the following query parameters:

Parameter	Description
code	An authorization code you ca use to exchange for an access token for your seller. This authorization code can only be used once.

Error Response:

In case the user did not consent to the authorization request, the redirect url will include the following parameter:

Parameter	Description
error	an error code, currently the only the only error value can be "cancel" in case seller declined authorization

POST https://checkout.fundboxpay.com/oauth/token

This endpoint is used for exchanging an authorization_code for an access_token.

In testing environment replace checkout.fundbox.com with checkout-integration.fbpay.im.

Request:

This call should be make using your **client_id** and **client_secret** as the **Authorization header**, specifically using **Basic HTTP Authentication** with the client_id as username and client_secret as password.

CURL example:

```
curl https://checkout-integration.fundboxpay.com/oauth/token
     -X POST
     -u "<client_id>:<client_secret>"
```

Parameter	Description
grant_type	Currently only 'authorization_code' is supported
code	The value of the authorization_code returned by Fundbox to your redirect url the authorization response.

Response:

Parameter	Description
access_token	The access token you can use to make requests on behalf of this seller. the access token acts as the **private_key** in the checkout integration.
public_key	This is **public_key** to use as described in the checkout integration.
token_type	Will always have the value **bearer**
expires_in	time in seconds until the token expires. you can disregard the value as we don't expire the token.

POST https://checkout.fundboxpay.com/oauth/revoke

This endpoint is used for token revocation. After revocation succeeded you can no longer use the token.
This endpoint is particularly useful when testing the oAuth integration - after revocation, you can re-test the authorization grant flow with the same seller.

In testing environment replace checkout.fundbox.com with checkout-integration.fbpay.im.

Request:

This call should be make using your **client_id** and **client_secret** as the **Authorization header**, specifically using **Basic HTTP Authentication** with the client_id as username and client_secret as password.

CURL example:

```
curl https://checkout-integration.fundboxpay.com/oauth/revoke
     -X POST
     -u "<client_id>:<client_secret>"
```

Parameter	Description
token	The value of the access token returned by Fundbox.
token_type_hint	Currently only 'access_token' is supported

Response:

On Success a 200 OK HTTP status code.

Remote Payment Request

Remote payment request allows a seller to place orders and re-orders for a customer without him being present..

High level flow:

This feature is currently enabled by contacting your Fundbox representative.

- When opening the checkout modal, instead of login, user chooses the remote payments flow.
- The buyer and sales agent email (optional) are requested, pre-filled with data from the order details object that was sent to the modal.
- Upon modal completion - an API call triggers an email to the buyer with a payment request. It's also available via the buyer's dashboard.
  - If the buyer isn't on Fundbox, they can onboard by clicking through the email.
  - If the buyer is using a different email address, clicking through the email will lead them to login and they can use a different email and the order will carry through.
- We report back to the platform via webhook if the buyer approved/declined or if the payment request was expired/cancelled.

the integration steps are the same as the regular checkout flow with few additions.

1 - Add checkout.js

Fundbox checkout.js is available at https://checkout.fundboxpay.com/static/merchant/checkout.js

Usage - when the checkout page loads, initiate FbxCheckout handler object using **FbxCheckout.configure()** passing it a configuration object containing:

- **fbxKey:** Your Fundbox public key
- **onComplete:** callback function, this function will be called upon a successful checkout through Fundobx and will receive the **fbxOrderToken** and the **transactionType** (capture/authorize/payment_request). with **payment_request** indicating the remote payment flow.

by receiving the transactionType you can choose the action (api call) that follows the order (capture/authorize/payment_request).
In payment_request flow the fbxOrderToken needs to be sent to your backend server for completing the payment request by calling the **orders/{fbx_order_token}/payment_request** api endpoint.

We mandate an API call to imitate the payment request to mitigate edge cases of cart abandonment after the checkout modal is completed but with no order ending up being placed, and to validate the order amount given by the platform in the API call is the same as the one in the checkout modal.

```
var fbxHandler = FbxCheckout.configure({
  fbxKey: 'fbx_pk_.....',
  onComplete: function (fbxOrderToken, transactionType) {
    // Token to be submitted to the backend for seller approval. this
token maps to the 'order'.
  }
});
```

2 - Initiate a checkout

In order to pre-populated buyer email and agent email you need to add them to the **orderDetails** json

OrderDetails

Key	Type	Required/Optional	Example
buyer_email **Note**: The buyer email you wish to send a payment request to.	string	**Optional**	"buyer@gmail.com"
agent_email **Note:** The agent email for receiving notification on the payment request progress	string	**Optional**	"agent@gmail.com"

3 - Send a payment request

Send a payment request via API:

After posting the fbx_transaction_token, returned from the checkout modal, to the merchant store server it can be used for send a payment request to the buyer.

Api endpoint - api/v2/orders/{fbx_order_token}/payment_request

curl example:

```
curl
https://checkout-integration.fundboxpay.com/api/v1/orders/{fbx_transaction_token}/payment_request
     -X POST
     -u "<public_api_key>:<private_api_key>"
     -H "Content-Type: application/json"
```

Response:

A successful response is http 200 status code.

4 - Reporting back payment request status via Webhook

Buyer approval/decline and transaction request expiry is

WEBHOOKS:

You can register webhook URLs for Fundbox to notify your system of various events relating to your account. When an event occurs - such as a buyer accepting a payment request via his dashboard. Fundbox creates and Event object.

This `Event` object contains all the relevant information about what just happened, including the type of event and the data associated with that event. Fundbox then sends the `Event` object, via an HTTP POST request, to any endpoint URLs that you have registered with Fundbox.

To set up an endpoint, you need to define a route on your server for receiving events, configure the webhook via API or via contacting your

Fundbox representative, so Fundbox knows where to POST events to, verify your endpoint is valid, and acknowledge your endpoint is receiving events successfully.

Event types reported via webhooks:

Event name	Description
BUYER_APPROVED_REQUEST	Buyer has approved the seller's transaction request via his dashboard
BUYER_DECLINED_REQUEST	Buyer has declined the seller's transaction request via his dashboard
REQUEST_EXPIRED	Transaction request has expired
BUYER_ADJUST_ACCEPT	Buyer has accepted the adjusted amount for a sellers capture request.
BUYER_ADJUST_DECLINE	Buyer has declined the adjusted amount for a sellers capture request.

1 - Create a webhook endpoint

Webhook data is sent as a **JWT** base64 encoded text in the POST request. The full event details are included and can be used directly after decoding the message using **JWT.** the decrypted JSON generally looks as following :

- **name** - the event type
- **body** - the event details
- **transaction_uuid** - the related transaction token (token used in the apply/authorize/transaction_request APIs).

example webhook message:

```
{"name": "BUYER_APPROVED_REQUEST",
 "body": {"status": "sent", "buyer": "Good Stuff Inc", "amount":
"21.20", "seller": "My Vineyard"},
 "transaction_uuid": "282f3e5e-7e63-495a-89c2-92d9f2af81b2"}
```

In JWT encoded format:

eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiIsImlzcyI6ImZieCJ9.eyJuYW1lIjoiQlVZRVJfQVBQUk9WRURfUkVRVUVTVCIsImJvZHkiOnsic3RhdHVzIjoic2VudCIsImJ1eWVyIjoiR29vZCBTdHVmZiBJbmMiLCJhbW91bnQiOiIyMS4yMCIsInNlbGxlciI6Ik15IFZpbmV5YXJkIn0sInRyYW5zYWN0aW9uX3V1aWQiOiIyODJmM2U1ZS03ZTYzLTQ5NWEtODljMi05MmQ5ZjJhZjgxYjIifQ.eEy7o2kR7tcFJ2yLxqcJy0QSDH4skKad2gATGUcKDIk

How to decode:

Notice that base64 string is actually 3 strings separated by a "." the second of which is the message body which can simply be decoded using base64decode.

We urge you to use a proper JWT decoder, because it can also verify the signature of the message (the 3rd part) - thus verifying that authenticity of it's origin (Fundbox):

- The secret key needed to verity the JWT signature is the sellers **private key**.

You can inspect the JWT format with online tools such as https://jwt.io/, in the example above, the secret key for the message is "sk_test_06b46d9a-e3a6-417d-a825-5bf4be329447", paste it where it says "your-256-bit-secret".

2 - Configure webhook settings

In order to get the webhook messages, you need to tell Fundbox about where to send events. Webhook endpoints are configured using api calls.

The API call body specifies the url and the event to listen to, use ALL to register this url for all events.

POST api/v2/webhook :

CURL example:

```
curl  "https://checkout-integration.fundboxpay.com/api/v2/webhook"
  -X POST
  -u "<public_api_key>:<private_api_key>"
  -H "Content-Type: application/json"
  -d "{\"url\":\"http://yourdomain.com/some_callback\",
\"event_type\":\"ALL\"}"
```

Response:

A successful response is http 200 status code.

GET api/v2/webhook :

Lists your registered endpoints

CURL example:

```
curl "https://checkout-integration.fundboxpay.com/api/v2/webhook"
  -X GET
  -u "<public_api_key>:<private_api_key>"
  -H "accept: application/json"
```

Response:

Response is of format {"event_name":"url"}, example:

```
{
  "ALL": "http://yourdomain.com/some_callback"
}
```