Global Affirm integration guide

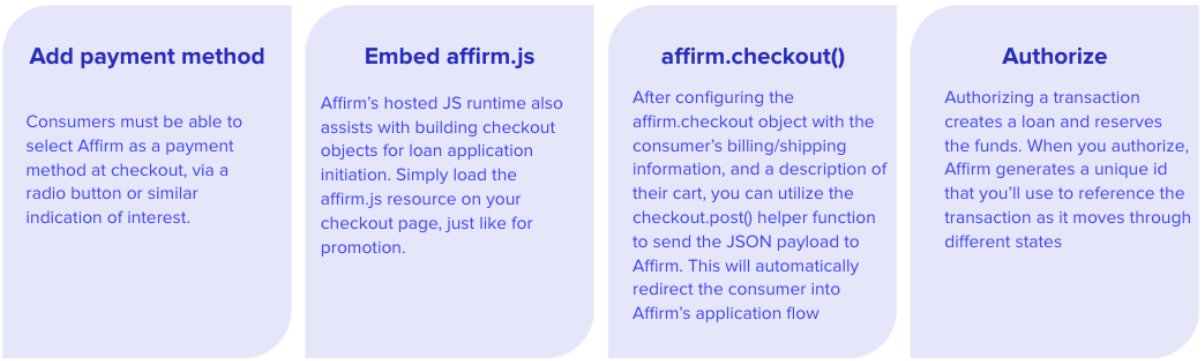
This step-by-step guide outlines how to implement global integration powered by Affirm APIs.

Overview

Affirm's global integration allows you to deploy Affirm in multiple countries and regions by utilizing a universal domain `global.affirm.com` and API schema. With the introduction of a universal domain, it provides a seamless expansion into international markets.

How it works

There are four steps to initiate the Affirm application:



Once the customer selects Affirm at checkout, the checkout initiation begins.

Step 1: Embed Affirm.js

The integration with Affirm.js used for promos and checkout allows for a quick and simple way to implement interactive components of Affirm to your platform. Our global integration

features extended parameters to reach international customers, which consist of an exposed `locale` and `country_code` field in the Affirm config.

public_api_key

The API keys will differ for each country. Currently, the following countries are represented by individual API keys:

- The United States
- Canada

> 🚧 API Keys
>
> Please ensure that the correct API keys are being used for your region.

Locale

The `locale` parameter allows Affirm to identify which locale you are serving your site in for any particular user. For example, let's say that you are rendering your site in Canadian French; perhaps this occurred because the user's browser setting detected Canadian French. You can provide Affirm with the user's language setting by passing it in the `locale` parameter. Affirm will then read the locale and translate the pages accordingly, thus, matching the language the user had seen on your site. However, we will not attempt to read the user's locale directly.

Country_code

The `country_code` parameter represents the country of legal incorporation of your store, which is shown to any given user. So, if you are showing your Canadian website/store to a user and you have a legal presence in Canada, you would pass `CAN` into the `country_code` parameter. This allows us to determine which regulations to abide by and which banks to partner with for this transaction.

> 🚧 Affirm's identity check
>
> In the above scenario, if the user doesn't turn out to be a Canadian resident later in the Affirm checkout flow, they won't pass Affirm's identity check; they are ineligible to receive an Affirm loan.

Additionally, Affirm expects an “Alpha 3” three-letter country code for this parameter. If nothing is sent, Affirm will assume "USA".

Supported values

Affirm supports the following combinations for these arguments:

country_code	locale	Country default locale
USA	en_US	en_US
CAN	en_CA | fr_CA	en_CA

The following will occur when a `locale` and/or `country_code` is not provided:

- When a `country_code` is *not* provided, the `country_code` will default to `USA`.
- When a `locale` is *not* provided, the `locale` the default locale for the `country_code` from the above table will be used. Locale will fall back to `en_US` (English-speaking US) if a `country_code` is also not provided

Including the Affirm.js script

The script (as shown below) must be included in the section on every page of your site.

The following URLs can be used for the AFJS `script`:

Sandbox: https://cdn1-sandbox.affirm.com/js/v2/affirm.js

Production: https://cdn1.affirm.com/js/v2/affirm.js

Global Embed Affirm.js

Open Recipe

global_affirm_config

```
<script>
	_affirm_config = {
		public_api_key: "{YOUR_PUBLIC_API_KEY}",
		script: "https://cdn1-sandbox.affirm.com/js/v2/affirm.js",
		locale: "fr_CA",
		country_code: "CAN",
	};

(function(m,g,n,d,a,e,h,c){var
b=m[n]||{},k=document.createElement(e),p=document.getElementsByTagName(e)[0],l=function
```

```
(a,b,c){return function(){a[b]._.push([c,arguments])}};b[d]=l(b,d,"set");var
f=b[d];b[a]={};b[a]._=[];f._=[];b._=[];b[a][h]=l(b,a,h);b[c]=function(){b._.push([h,arg
uments])};a=0;for(c="set add save post open empty reset on off trigger ready
setProduct".split("
");a<c.length;a++)f[c[a]]=l(b,d,c[a]);a=0;for(c=["get","token","url","items"];a<c.lengt
h;a++)f[c[a]]=function(){};k.async=
  !0;k.src=g[e];p.parentNode.insertBefore(k,p);delete
g[e];f(g);m[n]=b})(window,_affirm_config,"affirm","checkout","ui","script","ready","jsR
eady");
</script>
```

Once the `Affirm.js` is added to your site, you will gain access to methods within the Affirm object, which then triggers multiple actions.

Step 2: Render Affirm checkout

When using `Affirm.js`, merchants pass a checkout object during the checkout flow. You can create a `checkout` object and launch the Affirm checkout by utilizing the `affirm.checkout()` function.

For the global integration, merchants will pass an extended International Checkout Object and this schema can be used for all Affirm-supported countries. Please see the extended fields in the example below:

For "address," the expanded fields are as follows:

- `street1`
- `street2` (*optional*)
- `region1_code`
- `postal_code`

Please make sure the following field is accurately populated within the root of the checkout object:

- `currency`

Affirm.checkout() function

Calling the `affirm.checkout()` function initiates the following actions:

- Sends the checkout object to the Affirm backend.
- Redirects the customer to the Affirm checkout process on the Affirm domain or shows them an Affirm modal.

- Validates the required data in the checkout object.

Additionally, the `checkout` object should be configured with the following items:

- A `user_confirmation_url` to redirect your customer to a confirmation page after they confirm their loan.
- A `user_cancel_url` to redirect your customer to a cancellation page if they do not complete their loan application.
- The customer's billing/shipping address.
- A description of their cart.

checkout object

```
{
  "merchant": {
    "public_api_key": "{PUBLIC_API_KEY}",
    "user_cancel_url": "https://www.google.com",
    "user_confirmation_url": "https://www.affirm.com",
    "user_confirmation_url_action": "POST",
    "name": "Quebec Shoes"
  },
  "shipping": {
    "name": {
      "full": "John Doe"
    },
    "address": {
      "street1": "4519 Rue Levy",
      "street2": "Apt 1",
      "city": "Saint-Laurent",
      "region1_code": "QC",
      "postal_code": "H4R2P9",
      "country": "CAN"
    },
    "phone_number": "250-555-0199",
    "email": "john.doe@affirm.com"
  },
  "items": [
    {
      "sku": "SKU-1234",
      "item_url": "https://www.quebecshoes.com/shop",
      "display_name": "Grey Sneakers",
      "unit_price": 42500,
      "qty": 1
    }
  ],
  "order_id": "00de5cee-7226-4aec-b729-a571f773a58c",
  "shipping_amount": 2500,
  "tax_amount": 5000,
```

```
  "total": 50000,
  "currency": "CAD"
}
```

The code for currency is an ISO 4217 international currency code.

Step 3: Handle Callbacks

After you initiate a checkout and the customer confirms their Affirm loan, we send an `HTTP` request with the `checkout_token` to the URL you defined in the checkout object (`user_confirmation_url`). By default, Affirm sends this request via a `POST` request. However, you can configure the checkout object to have Affirm send this request via `GET`.

You choose how we send the `checkout_token` by setting the `user_confirmation_url_action` parameter in the checkout object.

- Setting it to `POST` sends the `checkout_token` in the body of the `HTTP` request (default setting).
- Setting it to `GET` sends the `checkout_token` in the query string of the `HTTP` request.

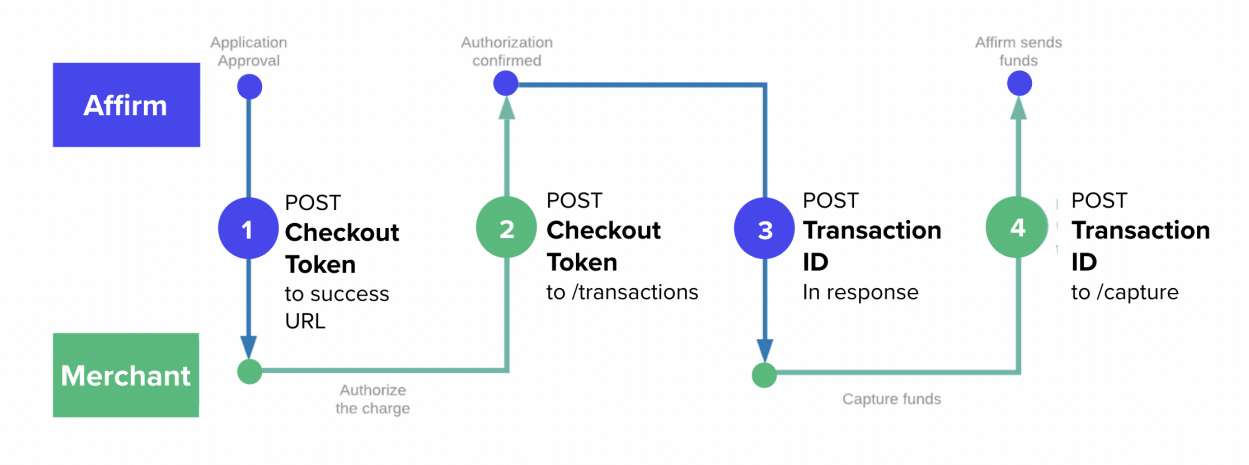
🚧 Modal checkout

> If you set the metadata mode to modal, you can retrieve the checkout_token from a javascript callback. Learn more about modal checkout.

Step 4: Authorize

When a customer successfully completes a checkout, it is recorded as a new purchase attempt. To be fulfilled via our Transactions API, this must be handled on your server side.

To better understand the process, the chart below illustrates what happens during a successful checkout on Affirm's and the merchant's end.



Transactions API

The global integration utilizes Affirm's Transaction API.

If you currently use the Affirm Charges API, please reference our guide to switch from Charges to Transactions. If you are currently using the non-global Affirm Transactions API, please reference this guide to make global updates.

Country-code request header

To transact outside of the United States, include a custom HTTP header parameter in any API calls made to Affirm, such as Auth, Capture Transaction, etc.:

- `country-code`

The country-code parameter will be followed by the three-letter country code as shown below:

- `country-code: "CAN"`

The country-code parameter specifies a 3-letter ISO 3166-1 alpha-3 country code indicating the country of legal incorporation for the merchant store where the transaction originated.

This parameter determines the Affirm regional environment used to route the API request. It also sets the language for end-user disclosures or notification emails generated from this transaction. If no country-code is provided, a default value of "USA" will be used.

For example, if "fr_CA" was specified during transaction initialization in Affirm.js, all subsequent emails and notifications related to this transaction will use Canadian French. Specifying a country-code in this API call overrides any previous values from the Affirm.js initialization.

Authorization process

1. Retrieve the `checkout_token`

Retrieve and save the `checkout_token` sent by Affirm via a `HTTP` request to the success callback.

Checkout_token

```
"checkout_token": "CJXXM8RERR0LC066"
```

2. Authorize the `checkout_token`

Authorize the `checkout_token` by passing the value in the `transaction_id` parameter to /transactions within 24 hours.

authorize checkout_token

```
curl --request POST \
     --url https://api.global.sandbox.affirm.com/api/v1/transactions \
     --header 'Accept: */*' \
     --header 'Content-Type: application/json' \
     --header 'country-code: CAN' \
     --data '
{
     "transaction_id": "CJXXM8RERR0LC066",
     "order_id": "JKLM4321"
}
'
```

3. Save the returned id

After authorizing the transaction, save the `id` from the transaction object returned in the response.

JSON

```
{
    "id": "9RG3-PMSE"
    "currency": "CAD",
    "amount": 11500,
}
```

Error handling

When initiating a checkout process, any resulting errors trigger a pop-up modal on the same page, displaying specific error details (e.g., "Invalid phone number"). This modal aims to inform users directly about the nature of the error encountered during the checkout request.

Developers have the option to implement a callback function that is executed upon the modal's closure.

However, it's important to note that this callback function cannot currently pass along any error-specific information displayed in the modal. This limitation means that while the callback can signify the modal's closure, it does not provide insights into the error message content.

Here's an example of how this event callback would be defined:

JavaScript

```
affirm.ui.ready(
    function() {
        affirm.ui.error.on("close", function(){
            alert("Please check your contact information for accuracy.");
        });
      }
);
```