

# APIs

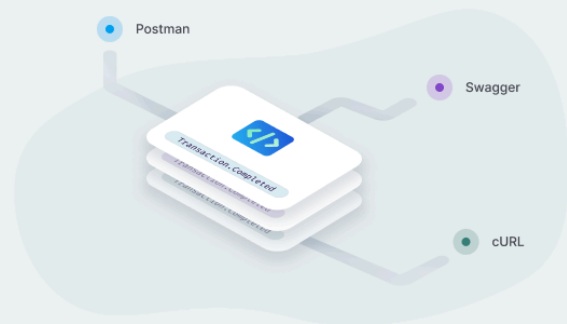


# 1. APIs intro

---

## APIs in action

Our APIs are built using the OpenAPI specification, making them easy to explore and integrate.



With Cross River's modern platform and easy-to-use APIs, you can create custom banking solutions. Built on the OpenAPI specification, our APIs are clear, consistent, and simple to integrate. Use tools like cURL, Swagger or Postman to test endpoints in our sandbox environment. This helps you tailor your solution to fit your business needs. Our API endpoint request examples are provided in multiple codes: cURL, Node.js, Python, Ruby and Go.

Stay updated with real-time events using our webhook system. No need to for API polling to constantly check for updates. Our webhooks notify you instantly about important changes like account updates and transaction statuses.

---

## Accounts

Seamless account functionality to create, open, and manage accounts.



### Customer management

Create customer records



### Account management

Open customer accounts

## Cards

Scale debit and credit card programs with a processor-agnostic issuing platform.



### Issuing and processing

Card products and virtual cards

## Payments

Streamlined payment solutions encompassing both local and international transfers.



### Instant payments

Clear and settle immediately



### Card payments

Send and collect funds



### International payments

Via SWIFT or local rails



### ACH

Originate standard and same day



### Wires

Transfer through Fedwire



### Checks

Check writing and deposits

## Lending

Robust set of API calls to create, update, and manage your loans.



### Create and update

Create and update a loan



### Application decisioning

Preapproval due diligence



### Loan funding/payment rails

Method of fund transfer



### Purchase a loan

Options using the selling system



## 2. Webhooks

---

Webhooks are an important component of any integration with Cross River. While APIs let you request data on demand, webhooks push real-time updates directly to your system eliminating the need for constant polling and reducing latency. In many cases, webhooks are more efficient and reliable than APIs alone, ensuring your application stays in sync with critical events like account changes, transaction updates, and processing outcomes.

Use webhooks within the Cross River systems to notify you when different events occur. The event returns a resource object that contains relevant details about the subject of each event. The full event details are included and sent to your system.

We use webhooks to update you on status and to report transaction changes. Webhooks report to your system with real-time notifications when an event happens.

You must register via API to use webhooks:

- [\*\*Register for accounts, cards and payment events\*\*](#)
- [\*\*Register for card payments events\*\*](#)
- [\*\*Register for Lending events\*\*](#)

## 2.1. Accounts, cards and payment events

---

When you work with Cross River accounts, card issuing, customer management and payments, you register for specific webhooks, unlike in [Card Payments](#) and [Lending](#) where you register for a set of webhooks.

These events each return a resource object that contains relevant details about the subject of each event. This eliminates the need to poll the API to discover changes. The full event details are included and sent to your system.

The system typically aggregates event objects every 30 seconds.

### IMPORTANT

Webhook delivery is guaranteed at-least-once. While rare, it's technically possible to receive the same event twice. Your message handlers should account for that edge case and be [idempotent](#).

## Event registration

There are several ways to receive event notifications:

- **Push:** Webhook events that are reported to a registered partner endpoint (a callback URL).
- **Poll:** Webhook events that are reported after a partner polls the events API.

The `POST` calls for the registration methods are the same. Make sure to select the correct registration `type`, either `Push` or `Poll`. We recommend that you use **Push** registration.

## Register for push events

Use `POST /webhooks/v1/registrations` to [register for webhook events](#) which includes defining the callback URLs where event reports should be delivered.

## IMPORTANT

Your system must respond to each Push event with a 200 status to avoid getting suspended.

In this example, your `partnerId` identifies you as the entity requesting webhook registration to the `eventName: Core.Account.Opened`, which is a `Push` registration. The `eventName` is being registered to the `callbackUrl: https://cos.yourcompanysite.com/account-events`.

### Sample webhook push registration



```
{
  "partnerId": "8gj76s99-jhso-89as-ns18-119nss8ch7ab",
  "eventName": "Core.Account.Opened",
  "type": "Push",
  "callbackUrl": "https://cos.yourcompanysite.com/account-events",
  "authUsername": "{userName}",
  "authPassword": "{password}",
  "format": "Basic"
}
```

## IMPORTANT

To confirm your callback URL is registered, call `PUT /v1/registrations{id}/ping`. It simulates a webhook event sent to your callback URL but doesn't include any resources. In addition, the `isPing` flag will be set to make it easy to handle the event appropriately.

## Register to poll for events

Use `POST /webhooks/v1/registrations` to [register for webhook events](#) to poll for an event status.

We recommend you poll for events only if you're not able to receive webhooks via Push.

In the example below, your `partnerId` identifies you as the entity requesting registration to be able to poll the `eventName` : `Core.Account.Opened` .

#### Sample Poll Registration



```
{
  "partnerId": "8gj76s99-jhso-89as-ns18-119nss8ch7ab",
  "eventName": "Core.Account.Opened",
  "type": "Poll",
  "format": "Basic"
}
```

## Poll for events

Call `GET /webhooks/v1/events/poll` to poll for events.

When you poll the system, the events are sent in `status` : `Pending` and every event has to be manually acknowledged.

Don't poll more than once every 30 seconds.

In the example below, the `eventName` : `Core.Account.Opened` polls the deposit account `resources` : `https://sandbox.crbcos.com/core/v1/dda/accounts/...` to check the registration status.

```
{
  "id": "g98sxn timer-9000-nks7-jjsy-xn6554bv9h10",
  "eventName": "Core.Account.Opened",
  "status": "Pending",
  "partnerId": "8gj76s99-jhso-89as-nsl8-119nss8ch7ab",
  "createdAt": "2022-11-04T17:12:34.806Z",
  "lastAttemptedAt": "2022-11-04T17:12:34.806Z",
  "resources": [
    "https://sandbox.crbcos.com/core/v1/dda/accounts/1234567890",
    "https://sandbox.crbcos.com/core/v1/dda/accounts/1234567891"
  ],
  "isPing": false
}
```

Attribute	Description
id	ID of the event being polled
eventName	Name of the event being polled
status	When polling for an event, the status will always be <b>Pending</b>
partnerId	Your ID in the Cross River system. This ID is in GUID format.
createdAt	When the event was created.
lastAttemptedAt	When the last attempt was.
resources	Elements (such as an account number or an event name) that the event is reporting on.
isPing	

## Acknowledging an event

Failing to acknowledge an event will cause the `GET /webhooks/v1/events/poll` endpoint to respond with the same information.

Call `POST /webhooks/v1/events/{id}/acknowledge` to acknowledge that you received the event.

Use the `id` you received when you checked the event `status` in this call. When you acknowledge an event, the status changes from `Pending` to `Success`.

## Delete a registration

Use the event registration `id` you received when you polled the event to delete an event registration.

## Event delivery unsuccessful

Monitor the status of your webhook registrations.

Event that we can't successfully deliver will expire after 7 days.

If the system can't deliver an event to one of your registrations, it makes several attempts to re-send the event. If we still can't deliver the event after several retries, its registration status changes to `Suspended` and we start to queue all your events for your registration. No further attempts are made to deliver previous or future events to this endpoint until your registration returns to an Active status.

If there is an event delivery failure, don't delete a registration and re-register for the same event. This prevents you from retrieving any events that were queued for delivery, as well as any events that fired in the time between deletion and re-registration.

If you get a `Suspended` status, review the logs of recent failed events to identify the issue. When the issue is resolved, restart your registration with the [Restart by ID](#) webhook API.

The status transitions to `Restarting`. If we can deliver an event successfully the status returns to `Active`. When the status returns to `Active` we deliver all the queued events

from when your registration status was `suspended` . If we can't successfully deliver at least 1 event, the status returns to `suspended` .

When a suspended webhook is restarted, all queued webhook notifications are delivered immediately.

## Event status

Whenever you receive an event, it is always in `Pending` status and refers to the event status (an event was created but not yet delivered), **and not the resource status**.

If you want to see if webhooks were delivered successfully, call `GET` `/webhooks/v1/events/{id}` . If the event was delivered successfully you will see `status:Success` .

## Authentication

Our system optionally supports basic authentication on each registration. You can supply an `AuthUsername` and `AuthPassword` which will be base64 encoded and included as an Authorization header on each webhook you receive.

### IMPORTANT

The preferred security practice is to use webhook signature verification instead of basic authentication.

## Signatures

Every webhook is signed with a standard HMAC with SHA256 hash. This provides an added layer of protection from replay attacks. Without a signature, an attacker could intercept a valid payload and retransmit it.

The signature can be found in the event's request header and includes the event timestamp. Our system generates a timestamp and signature each time we send an event

to your endpoint. If our system retries an event after a previous failure, we generate a new signature and timestamp for the new delivery attempt. If a timestamp is subsequently changed, the signature is then invalid.

#### Sample Request Header



```
cos-signature:t:2019-04-02T11:33:26.6672036-04:00,  
v1:{secret}
```

### IMPORTANT

If you receive an event with an old timestamp we recommend you discard or flag it. The recommended tolerance between timestamp and time of delivery will vary by your application's requirements, but should usually be less than 20 minutes. We do however, recommend special consideration for situations where there has been an extended outage at either party.

- All signatures are version 1 (v1). Any other schemes besides v1 should ignored.
- The signature is a standard HMAC with SHA256 hash.

## Validating the signature

1. Extract the timestamp and signature from the cos-signature header.
2. Concatenate the timestamp with the event body with a period in between the two values (as illustrated below).
3. Compute an HMAC with the SHA256 hash function. Use the signing secret (provided by the Integration Team) as the key and use the string created in step #2 as the message. Also be sure to base64 encode the computed hash value.
4. Compare the hash generated in step #3 with the signature extracted from the cos-signature header. If the hash and signature match, this is a valid request from COS.
5. Compare the timestamp to the time received. If the difference exceeds your tolerance for accepting messages, you can reject the message or flag it for further review.



```
2019-04-02T11:33:26.6672036-04:00>{"id":"123","eventName":"Core.Customer.Email"}
```

### Example:

A Node.js script that shows how a webhook signature would typically be validated.

```
const crypto = require('crypto');

function validateSignature(header, payload, signingSecret) {
    let headerParts = header.split(',');
    let timestamp = headerParts[0].replace('t:', '');
    let signature = headerParts[1].trim().replace('v1:', '');

    let secret_buffer = Buffer.from(signingSecret, 'base64');
    let hmac = new crypto.createHmac('sha256', secret_buffer);

    let hmac_digest = hmac.update(`${timestamp}.${payload}`, 'bin
    console.log(hmac_digest);

    return hmac_digest === signature;
}

let payload = '{"id":"e7ead744-d6ff-4521-863d-abab0176f849","eventName":"Core
let signatureHeader = 't:2020-04-28T18:45:15.6360965-04:00, v1:MvGXdx101P8+Yj
let secret = 'uVdwwB9HIFZ+5/8nmta5PXu6p1kxZcQmXPCNBRhiVNuKNBhIgh8Mvm1D7FYoVf

let result = validateSignature(signatureHeader, payload, secret);

console.log(result);
```

## FAQ

Why is my webhook status *Suspended*?

*Suspended* status means that we could not successfully deliver an event to one of your registrations. See Webhooks for COS.

### **What is the retention period of webhooks?**

Currently there is no limitation to the retention period.

### **What webhook events are available?**

All available webhook events can be retrieved via Swagger by calling [https://sandbox.crbcos.com/webhooks/swagger/ui/index#!/Meta/Meta\\_GetAll](https://sandbox.crbcos.com/webhooks/swagger/ui/index#!/Meta/Meta_GetAll).

### **What is the maximum number of resources included in a single webhook event?**

50k for events received in basic format

1k for events in extended format.

### **How are webhook failures handled?**

After a delivery failure, COS will retry 3 additional times after approximately 30 - 60 seconds before your registration updates to a *Suspended* status. If we still can't deliver the event after several retries, its registration status changes to *Suspended* and we start to queue all your events for your registration. No further attempts are made to deliver previous or future events until your registration returns to an *Active* status.

### **What is the IP address from which COS webhooks are sent?**

For sandbox it's 66.206.202.40 and for production its 66.206.202.41.

## 2.1.1. Event formats

---

An event payload is delivered in a standard format and contains the resources it's reporting on. Our system supports basic and extended formats.

The table below shows the different fields included in the event together with descriptions and explanations of the information they represent.

Column	Description
id	The unique ID for the event. In GUID format.
eventName	The name of the event.
status	This column shows the event status, which could be: <ul style="list-style-type: none"><li>• <b>Pending:</b> the event has been created, but no attempt to send it has been made yet.</li><li>• <b>Success:</b> (internal only) the event was successfully sent to the registered URL.</li><li>• <b>Failed:</b> (internal only) the event could not be delivered to the registered URL.</li></ul>
createdAt	The date and time the event was created.
lastAttemptedAt	The date and time the event was last sent.
partnerId	The unique identification number for the partner resource.
resources	Refers to the resource relating to the event. For example, an account that was opened. It contains the API URI that triggered the event.
details	Contains information related to the API call. These details are only displayed in an extended webhook

## Basic

Basic events have a small payload, excluding additional details. They can deliver up to 50k resources at once.

The ID in the resources object (in this example 73da01c7-b85b-4a58-9395-b04900de43cf) corresponds to the `id` field returned in the `POST /v1/payments` response body. This payment `id` is used to track the payment.

#### Basic



```
{
  "id": "51d2e4e8-5fee-4ad8-8b03-b1c6012a8459",
  "eventName": "Ach.Payment.Sent",
  "status": "Pending",
  "partnerId": "1e5d3f04-ae24-4af6-9e30-aecf012b99dd",
  "createdAt": "2024-08-08T14:06:52.013-04:00",
  "resources": [
    "ach/v1/payments/55f641cf-102b-4920-83fd-b1c6012a3d2d"
  ],
}
```

## Extended

Extended events have a larger payload, including additional details. They can deliver up to 1k resources at once.

We recommend using extended webhooks for all `xxx.xxx.Received` events.

This format returns all the information included in the basic format and a `details` object. The `details` object includes information relevant to the event type.

### Example

`ACH.Payment.Sent` webhook event in the extended webhook format.

- The first part of the webhook event payload provides information about the webhook event itself.
- The `resources` object of the webhook event shows information about the ACH payment.

- The `details` object gives specific information.

Extended



```
{
  "id": "51d2e4e8-5fee-4ad8-8b03-b1c6012a8459",
  "eventName": "Ach.Payment.Sent",
  "status": "Pending",
  "partnerId": "1e5d3f04-ae24-4af6-9e30-aecf012b99dd",
  "createdAt": "2024-08-08T14:06:52.013-04:00",
  "resources": [
    "ach/v1/payments/55f641cf-102b-4920-83fd-b1c6012a3d2d"
  ],
  "details": [
    {
      "paymentId": "55f641cf-102b-4920-83fd-b1c6012a3d2d",
      "coreTransactionId": "fddea7f7-556e-4c93-acc4-b1c6012a6bf3",
      "memoPostId": "fddea7f7-556e-4c93-acc4-b1c6012a6bf3",
      "clientBatchId": null,
      "clientBatchSequence": null,
      "accountNumber": "2151546989",
      "postingCode": null,
      "clientIdentifier": null,
      "purpose": null,
      "fedBatchId": "29f4b78f-f056-4ac5-9a4e-b1c6012a60d2",
      "fedBatchSequence": null
    }
  ]
}
```

## 2.1.2. Webhook management

---

Webhooks return events after an API endpoint is called. The returned `webhook` event contains details relevant to the API call.

Find these APIs in the webhook module of our COS sandbox.

Each webhook event has **request and response codes**.

**Get started** with Cross River APIs

### GET calls

- Add query parameters to filter the response the API returns to general GET calls.
- Use **pagination** to control presentation of your results.

This table presents APIs we describe in detail.

Action	API call	Description
<b><u>Register</u></b>	POST /v1/registrations	Registers for webhook event delivery
<b><u>Registered</u></b>	GET /v1/registrations/	Returns a list of all webhook registrations and their statuses
<b><u>Registered by ID</u></b>	GET /v1/registrations/{id}	Returns a list of all webhook registrations and their statuses by ID
<b><u>Verify active</u></b>	GET /v1/registrations/{id}	Returns a webhook registration by ID
<b><u>Restart by ID</u></b>	PUT /v1/registrations/{id}/ restart	Restarts a webhook registration by ID
<b><u>Delete by ID</u></b>	DELETE /v1/registrations/{id}	Deletes a webhook registration by ID
<b><u>Ping by ID</u></b>	PUT /v1/registrations/{id}/ ping	Calls a webhook registration to check its status
<b><u>Update by ID</u></b>	PUT /v1/registrations/{id}	Updates the details of a webhook registration by ID

## 2.1.2.1. Register

---



## Endpoint: /webhooks/v1/registrations

Registers for webhook event delivery

POST

https://sandbox.crbcos.com/webhooks/v1/registrations



Request

Response

### BODY PARAMETERS

**partnerId** String **required**

Your ID in the Cross River system. This ID is in GUID format.

**eventName** String **required**

Webhook event being reported

**callbackUrl** String **optional**

The value is a URL. Webhooks are reported to this URL as a result of a triggered action. Make sure the callback URL is added to your allowlist. SSL required.

**authUsername** String **optional**

Basic authentication username to include in header of event. 255 character limit.

**authPassword** String **optional**

Basic authentication password to include in event header. 255 character limit.

**type** String **optional**

Type of registration:

- Push
- Poll
- File

**format**

String

optional

Type of JSON:

- Basic
- Extended



Curl



Node.js



Python



Ruby



Go



```
curl --location 'https://sandbox.crbcos.com/Webhooks/v1/registrations' \
--data '{
  "partnerId": "19222b81-0e1e-452d-a842-b2f1011c16f3",
  "eventName": "Cards.Card.Activated",
  "callbackUrl": "https://webhook.site/27b9814b-16eb-4099-9999-32594aee5",
  "type": "Push",
  "format": "Extended"
}
```

## Responses

● 200

● 400



```
{
  "id": "8d6115c0-2a88-4a58-b731-b306002e8bc2",
  "partnerId": "19222b81-0e1e-452d-a842-b2f1011c16f3",
  "eventName": "Cards.Card.Activated",
  "callbackUrl": "https://webhook.site/27b9814b-16eb-4099-9999-32594aee5",
  "consecutiveErrors": 0,
  "status": "Active",
  "type": "Push",
  "format": "Extended"
}
```

## 2.1.2.2. Registered

---

## Endpoint: /webhooks/v1/registrations

Returns a list of all webhook registrations by event delivery or by polling the system.

GET

https://sandbox.crbcos.com/webhooks/v1/registrations



Request

Response

### QUERY PARAMETERS

**partnerId**

String

**required**

Your ID in the Cross River system. This ID is in GUID format.

**eventName**

String

**optional**

Webhook event being reported

**callbackUrl**

String

**required**

The value is a URL. Webhooks are reported to this URL as a result of a triggered action. Make sure the callback URL is added to Cross River's allowlist. SSL required.

**pageNumber**

Integer

**optional**

Current page number determined by the total number of records and the number of records per page

**pageSize**

Integer

**optional**

Number of records to list on a page. Maximum is 50.



Curl



Node.js



Python



Ruby



Go



```
curl
```

```
--location 'https://sandbox.crbcos.com/webhooks/v1/registrations?partnerId'
```

Responses

● 200





## 2.1.2.3. Registered by ID

### Endpoint : /webhooks/v1/registrations/{id}

Returns a webhook registration by ID

GET

https://sandbox.crbcos.com/webhooks/v1/registrations/{id}



Request

Response

#### PATH PARAMS

id String required

The webhook registration ID. You receive this ID in the response when you register a webhook event. This ID is in GUID format.



Curl



Node.js



Python



Ruby



Go



```
curl --location --globoff 'https://sandbox.crbcos.com/webhooks/v1/registra
```

#### Responses

● 200



```
{
  "id": "8d6115c0-2a88-4a58-b731-b306002e8bc2",
  "partnerId": "19222b81-0e1e-452d-a842-b2f1011c16f3",
  "eventName": "Cards.Card.Activated",
  "callbackUrl": "https://webhook.site/27b9814b-16eb-4099-9999-32594aee5",
  "consecutiveErrors": 0,
  "status": "Active",
  "type": "Push",
  "format": "Extended"
}
```



#### 2.1.2.4. **Verify active**

---

## Endpoint : /webhooks/v1/registrations

Returns a list of webhooks which are active

GET

https://sandbox.crbcos.com/webhooks/v1/registrations



Request

Response

### PATH PARAMS

**id** String optional

The webhook registration ID. You receive this ID in the response when you register a webhook event. This ID is in GUID format.



Curl



Node.js



Python



Ruby



Go



```
curl --location 'https://sandbox.crbcos.com/webhooks/v1/registrations'
```

Responses

● 200





## 2.1.2.5. Restart by ID

---

## Endpoint: /webhooks/v1/registrations/{id}

Restarts webhook registration by ID

**PUT**

<https://sandbox.crbcos.com/webhooks/v1/registrations/{id}/restart> 

**Request**

Response

### PATH PARAMS

<b>id</b>	String	optional
-----------	--------	----------

The webhook registration ID. You receive this ID in the response when you register a webhook event. This ID is in GUID format.



Curl



Node.js



Python



Ruby



Go



```
var https = require('follow-redirects').https;
var fs = require('fs');

var options = {
  'method': 'PUT',
  'hostname': 'sandbox.crbcos.com',
  'path': '/webhooks/v1/registrations/{id}/restart',
  'headers': {
  },
  'maxRedirects': 20
};

var req = https.request(options, function (res) {
  var chunks = [];

  res.on("data", function (chunk) {
    chunks.push(chunk);
  });

  res.on("end", function (chunk) {
    var body = Buffer.concat(chunks);
    console.log(body.toString());
  });

  res.on("error", function (error) {
    console.error(error);
  });
});

req.end();
```



## Responses

● 200

● 400



```
{
  "id": "b38c1805-3f4c-470d-86bb-b288015024ce",
  "partnerId": "ba9d0234-26e9-43a2-9070-aedd0108eadb",
  "eventName": "Ach.Payment.Received",
  "callbackUrl": "https://webhook.site/ef6bdf1e-665c-42e8-9387-e2e89cfc89d",
  "authUsername": "",
  "consecutiveErrors": 2,
  "lastError": "2025-02-21T01:15:55.61-05:00",
  "status": "Restarting",
  "type": "Push",
  "format": "Extended"
}
```

## 2.1.2.6. Delete by ID

### Endpoint: /webhooks/v1/registrations/{id}

Deletes a webhook registration by ID

**DELETE**

https://sandbox.crbcos.com/webhooks/v1/registrations/{id}



**Request**

Response

#### PATH PARAMS

**id** String **required**

The webhook registration ID. You receive this ID in the response when you register a webhook event. This ID is in GUID format.



Curl



Node.js



Python



Ruby



Go



```
curl --location --globoff --request DELETE 'https://sandbox.crbcos.com/webhooks/v1/registrations/{id}' --data ''
```

#### Responses

● 200



```
{
  "id": "8d6115c0-2a88-4a58-b731-b306002e8bc2",
  "partnerId": "19222b81-0e1e-452d-a842-b2f1011c16f3",
  "eventName": "Cards.Card.Activated",
  "callbackUrl": "https://webhook.site/27b9814b-16eb-4099--32594aee5e04",
  "consecutiveErrors": 0,
  "status": "Active",
  "type": "Push",
  "format": "Extended"
}
```



## 2.1.2.7. Ping by ID

### Endpoint: /webhooks/v1/registrations/{id}/ping

Ping a webhook registration by ID to check the status.

**PUT**

<https://sandbox.crbcos.com/webhooks/v1/registrations/{id}/ping> 

**Request**

Response

#### PATH PARAMS

**id** String **required**

The webhook registration ID. You receive this ID in the response when you register a webhook event. This ID is in GUID format.



Curl



Node.js



Python



Ruby



Go



```
curl --location --globoff --request PUT 'https://sandbox.crbcos.com/webhook'
```

#### Responses

● 202



## 2.1.2.8. Update by ID

---

## Endpoint: /webhooks/v1/registrations/{id}

Updates the details of a webhook registration by ID

PUT

https://sandbox.crbcos.com/webhooks/v1/registrations/{id}



Request

Response

### PATH PARAMS

**id** String optional

The webhook registration ID. You receive this ID in the response when you register a webhook event. This ID is in GUID format.

**partnerId** String required

Your ID in the Cross River system. This ID is in GUID format.

**eventName** String required

Webhook event being reported 255 character limit.

**callbackUrl** String required

The value is a URL. Webhooks are reported to this URL as a result of a triggered action. Make sure the callback URL is added to Cross River's allowlist. SSL required. 255 character limit. Required for Push registrations.

**authuserName** String optional

Basic authentication username to include in header of event. 255 character limit.

**authPassword** String optional

Basic authentication password to include in event header. 255 character limit.

**consecutiveErrors**

Integer

optional

**lastError**

String

optional

**status**

String

optional

Values:

- Suspended
- Active
- Restarting

**type**

String

optional

Values:

- Push
- Poll
- File

**format**

String

optional

Event format:

- Basic
- Extended



Curl



Node.js



Python



Ruby



Go



```
curl --location --globoff --request PUT 'https://sandbox.crbcos.com/webhook' --data '{
  "partnerId": "19222b81-0e1e-452d-a842-b2f1011c16f3",
  "eventName": "ACH.batch.imported",
  "callbackUrl": "https://webhook.site/27b9814b-16eb-4099-9999-32594aee5e0"
}'
```

## Responses

● 200



```
{
  "id": "d0dc5a57-ce2a-40d8-b7c4-b306003dd243",
  "partnerId": "19222b81-0e1e-452d-a842-b2f1011c16f3",
  "eventName": "ACH.Batch.Imported",
  "callbackUrl": "https://webhook.site/27b9814b-16eb-4099-9999-32594aee5",
  "consecutiveErrors": 0,
  "status": "Active",
  "type": "Push",
  "format": "Basic"
}
```



## 2.1.3. Webhook events

---

The tables below describe the different webhook events you can register for. Where relevant, additional information is provided for some of the events.

### Customer events

See Customer Management [webhook events](#) for some examples.

Event Name	Description
<code>Core.Customer.Onboarded</code>	New customer was onboarded
<code>Core.Customer.Profile.Changed</code>	Customer profile was changed
<code>Core.Customer.Name.Changed</code>	Customer name changed
<code>Core.Customer.Address.Added</code>	Customer address was added
<code>Core.Customer.Address.Changed</code>	Customer address was changed
<code>Core.Customer.Address.Removed</code>	Customer address was removed
<code>Core.Customer.Phone.Added</code>	Customer phone number was added
<code>Core.Customer.Phone.Changed</code>	Customer phone number was changed
<code>Core.Customer.Phone.Removed</code>	Customer phone number was removed
<code>Core.Customer.Identification.Added</code>	Customer identification was added
<code>Core.Customer.Identification.Changed</code>	Customer identification changed
<code>Core.Customer.Identification.Removed</code>	Customer identification was removed
<code>Core.Customer.Email.Added</code>	Customer email was added
<code>Core.Customer.Email.Changed</code>	Customer email was changed
<code>Core.Customer.Email.Removed</code>	Customer email was removed
<code>Core.Customer.Ofac.Changed</code>	Customer OFAC scan resulted in a status change
<code>Core.Customer.PepScan.Changed</code>	Customer PEP Scan resulted in a status change
<code>Core.Statement.Generated</code>	Account statement generated
<code>Core.TaxDocument.Generated</code>	Tax document generated

## General account webhook events

See Accounts [webhook events](#) for some examples.

Event Name	Description
Core.Account.Opened	A new account was opened
Core.Account.Status.Changed	Status of an account changed
Core.Account.Closed	Account was closed
Core.Account.Title.Changed	Account title changed
Core.Account.Settled	Account settled
Core.Account.Balance.Changed	Indicates account balance fluctuations
Core.Relationship.Added	Relationship was added to the account
Core.Relationship.Removed	Relationship was removed from the account
Core.Restriction.Placed	Restriction was placed on the account
Core.Restriction.Removed	Restriction was removed for the account
Core.StopPayment.Placed	Stop payment placed on the account
Core.StopPayment.Removed	Stop payment removed from the account
Core.Transaction.Completed	Transaction completed successfully
Core.Transaction.Rejected	Transaction was rejected on account
Core.Activity.Posted	Activity was posted to the account
Core.MemoPost.Created	Memo post was created on the account
Core.MemoPost.Changed	Memo post was changed
Core.MemoPost.Removed	Memo post was removed
Core.Hold.Placed	Hold placed on the account
Core.Hold.Changed	Hold status has changed
Core.Hold.Removed	Hold was removed from the account

# Subledgers webhook events

See Accounts [webhook events](#) for some examples.

Event Name	Description
Core.SubAccount.Opened	Subledger (subaccount) was opened
Core.SubAccount.Status.Changed	Status of a subledger (subaccount) changed
Core.SubAccount.Closed	Subledger (subaccount) was closed
Core.SubAccount.Title.Changed	Title of subledger (subaccount) changed
Core.SubAccount.Balance.Changed	The balance of the subledger has changed

## Deposit sweeps webhook events

Event Name	Description
Sweeps.Intrafi.Account.Updated	Sweeps-related data on the account record has changed

## Cards events

See cards webhook events for some examples.

Event Name	Description
<code>Cards.Card.Ordered</code>	Card ordered
<code>Cards.Card.Created</code>	Card created
<code>Cards.Card.Activated</code>	Card activated
<code>Cards.Card.Status.Changed</code>	Card state has changed
<code>Cards.Card.Pin.Set</code>	PIN set for card
<code>Cards.Card.Suspended</code>	Card use temporarily suspended
<code>Cards.Card.Unsuspended</code>	Suspension removed from card
<code>Cards.Card.Profile.Changed</code>	Profile of cardholder changed
<code>Cards.Card.Replaced</code>	New card created to replace an existing card
<code>Cards.Card.Admin.Suspended</code>	Card has been suspended by an Admin
<code>Cards.Card.Closed</code>	Card closed
<code>Cards.Transaction.Authorized</code>	Card transaction approved
<code>Cards.Transaction.Cleared</code>	Card transaction successfully completed
<code>Cards.Transaction.Declined</code>	Card transaction not approved
<code>Cards.Transaction.Reversed</code>	Payment credited to the originating account

## RTP events

See Instant Payments [webhook events](#) for some examples.

Event Name	Description
Rtp.Payment.Sent	RTP payment has been successfully sent to the receiving institution and the funds are available in the receiver's bank account
Rtp.Payment.Received	Inbound payment has been received from another institution and has successfully posted to an account in COS
Rtp.Payment.Rejected	Payment has been rejected by the receiving institution or RTP Network. The payment originated from an account with insufficient funds.
Rtp.Payment.Canceled	Payment was canceled at the request of the partner or a payment with a status of ResearchRequired has been canceled
Rtp.Payment.ResearchRequired	No response was received for an outbound payment, manual research is required to determine if the payment should be completed or canceled
Rtp.Payment.Failed	Payment has failed due to technical reasons.
Rtp.Hold.Escalated	A hold relating to either an outbound payment was marked for escalation by a member of the Operations team. Commonly this is used to indicate additional documentation or action is needed by the partner.
Rtp.Refund.Requested	Request has been received to refund a previously accepted credit transfer
Rtp.Payment.Queued	The receiving institution is offline. The payment will be processed when the receiving institution is back online.
Rtp.Limits.Utilization.Changed	This webhook fires when the percentage of successful transfers hits 5%, and fires again every time the transfer interval goes up 5% (at 5%, 10%, 15% and so on).

## International payments events

See International Payments [webhook events](#) for some examples.

Event Name	Description
International.Hold.Cleared	Hold on payment cleared. Funds sent.
International.Hold.Escalated	Payment on hold and status escalated. Additional actions needed.
International.Payment.Sent	Funds sent via a wire payment to the receiver bank
International.Payment.Canceled	Payment canceled at partner request
International.Payment.Returned	Payment rejected by the receiving bank

## ACH events

See ACH [webhook events](#) for some examples.

Event Name	Description
Ach.Batch.Canceled	Any pending or on-hold payments in an ACH client batch canceled
Ach.Batch.Imported	Entire ACH client batch import process into the CR system complete
Ach.Payment.Canceled	ACH payment canceled at partner request
Ach.Payment.Rejected	ACH payment rejected due to a compliance failure or fail to fund
Ach.Payment.Sent	ACH payment sent to the Federal Reserve
Ach.Payment.Received	ACH inbound payment received
Ach.Payment.ReceivedEarly	ACH early direct deposit payment received
Ach.Return.Received	ACH payment was returned from the receiving bank and has been marked as received
Ach.Noc.Received	A Notification of Change was sent from the receiving bank regarding a previous origination
Ach.Hold.Escalated	ACH inbound or outbound payment is on hold and it's status was escalated. This is often used to indicate that additional actions are needed

## Wire events

See Wires [webhook events](#) for some examples.



Event Name	Description
Wire.Payment.Approved	Wire payment was approved via the API. This can happen because authorization or dual authorization is required.
Wire.Payment.Cancelled	Outbound wire was canceled at the partner's request
Wire.Payment.Rejected	Outbound wire could not be processed due to compliance reasons or was rejected by the Federal Reserve
Wire.Payment.Processed	Outbound wire has been transmitted to the Federal Reserve. Still possible the wire could be rejected
Wire.Payment.Sent	Outbound wire has been transmitted to the Federal Reserve and has been successfully acknowledged. IMAD number is now available.
Wire.Payment.Received	Inbound wire has been received from another bank
Wire.Hold.Escalated	A hold relating to either an inbound or outbound wire was marked for escalation by a member of the Operations team. Commonly this is used to indicate additional documentation or action is needed by the partner.

## Checks events

See Checks [webhook events](#) for some examples.

Event Name	Description
Check.Payment.Sent	A check deposit has been sent to the Federal Reserve for clearing
Check.Payment.Received	A check drawn on an account in COS has been deposited at another bank and has been received from the Federal Reserve for posting
Check.Payment.Canceled	An outbound payment has been canceled at the request of the partner
Check.Payment.Rejected	Outbound check could not be processed due to compliance reasons or was rejected by the Federal Reserve
Check.Return.Received	A previously deposited check has been returned by the receiving bank
Check.Details.Changed	Details of a check have been changed
Check.Hold.Escalated	A hold relating to either an inbound or outbound check was marked for escalation by a member of the Operations team. Commonly this is used to indicate additional documentation or action is needed by the partner.
Check.Policy.Changed	A check's Reg CC Policy has been updated
Check.PositivePay.Created	A check has been authorized for Positive Pay
Check.PositivePay.Revoked	A check's Positive Pay status has been revoked
Check.Payment.Unauthorized	A check drawn on an account in COS has been presented but was not authorized for Positive Pay

## 2.1.4. Request and response codes

### Registration status

The following table describes the status field returned in a webhook registration.

Status	Description
Active	Everything is OK.
Suspended	Something went wrong. Several attempts to send events to this registration failed. No further attempts will be made until registration is restarted.
Restarting	We are attempting to restart this registration. An event must be successfully transmitted before the status will transition back to Active.

### Registration types

The following table describes the type of webhook event you want to register for.

Type	Description
Push	Events will be pushed to a partner endpoint. A 200 OK response is expected in order for it to be considered successful (default).
Poll	Events will be consumed by partner polling the events API periodically and calling acknowledge API method for each event successfully processed.
File	Events will be sent to the partner via <a href="#">SFTP</a> in a CSV file once per day

### Event status

The following table describes the status of the webhook event.

Status	Description
Pending	Event has been created, but no attempt to send it has been made yet
Success	Event was successfully delivered to the registered URL
Failed	We could not deliver the event to the registered URL. See the event logs for more detail on the failure reason.

## Format

The following table describes the amount of detail in the webhook event payload that you request.

Format	Description
Basic	Webhook is delivered with an object containing resource identifiers related to that event
Extended	With this format you get everything in a 'Basic' event, in addition you will receive a 'Details' object containing various meta data pertaining to each resource. For example, on the <code>ACH.Return.Received</code> event you would receive the Return Code and Original Payment ID.

## 2.2. Card payments

---

We use webhooks to update you on status and to report transaction changes. Webhooks report to your system with real-time notifications when an event happens.

The event returns a resource object that contains relevant details about the subject of each event. This eliminates the need to poll the API to discover changes. The full event details are included and sent to your system.

Typically, P2C aggregates event objects every 30-120 seconds.

In P2C, a single registration registers the callback URL for all webhook events.

### Event registration

Use `POST /api/WebhookRegistrations` to register to poll for event status.

### Event details

Refer to [Webhook events](#) for details.

Call	Description
CardAuthorized	<p>Reports:</p> <ol style="list-style-type: none"> <li>1. When a card authorization attempt is completed and the status of the card.</li> <li>2. When a card was signed up via an iFrame.</li> </ol>
CardStatusChanged	Reports if a card changed status to active or inactive
TransactionCompleted	<p>Reports the completion of a transaction request. This webhook is sent only 1 time. If a transaction has changed after the initial execution, the <code>TransactionStatusChanged</code> webhook will be sent.</p>
TransactionStatusChanged	Reports the changed status of a transaction
AccountCreditedReceived	The COS account of the merchant is credited with the amount mentioned on the payload at the timestamp mentioned on the payload.

# 2.2.1. Event formats

---

An event payload is delivered in a standard format and contains the resources it's reporting on. Our system supports basic and extended formats.

The table below shows the different fields included in the event together with descriptions and explanations of the information they represent.

Column	Description
EventId	The unique ID for the event.
EventName	The name of the event.
Data	Different depending on the event (see table below).

Event name	Actions	Description
Card Authorized	RequestID	ID of the request
	CardToken	The card token
	Authorized	"true" or "false"
	ResponseReceived	"true" or "false"
	ResponseDescription	Can be "OK"
	Error	Can be "null" or a description of the error
	AddressVerified	Verification of the address. "true" or "false"
	Rail	
	CustomerReferenceNumber	
TransactionCompleted	TransactionId	ID of the transaction
	Amount	Amount of transaction in \$'s.
	TransactionRequestedAt	The date and time the transaction was requested
	TransactionStatus	
	ErrorDescription	Example: "Card token not found"
	CreditCardId	Card token
	Rail	
CardStatusChanged	StatusChangedAt	The date and time the card status changed
	CreditCardId	Card token
	OldStatus	New status of the card. For example "active"
	NewStatus	New status of the card. For example "inactive"



Event name	Actions	Description
TransationStatusChanged	TransactionId	ID of the transaction
	Amount	Amount of transaction in \$'s.
	TransactionRequestedAt	The date and time the transaction was requested
	OldTransactionStatus	Status of previous transactions. For example: succeeded
	NewTransactionStatus	Status of new transactions. For example: pending
	StatusChangedAt	The date and time the status changed
	CreditCardId	Card token
	Rail	
Transaction	BatchId	ID of the batch
	TotalTransactionsCount	Total number of transactions
	FailedTransactionCount	Number of failed transactions
	InvalidTransactionCount	Number of invalid transactions
	SuccessTransactionCount	Number of successful transactions
	TransactionInProcessCount	Number of transactions in process
	RequestedAt	The date and time the transaction was requested.
	CompletedAt	The date and time the transaction was completed.



## 2.2.2. Webhook management

Webhooks return events after an API endpoint is called. The returned `webhook` event contains details relevant to the API call.

Find these APIs in the webhook module of our Push to card (P2C) sandbox.

**Get started** with Cross River APIs

This table presents APIs we describe in detail.

Action	API call	Description
<b><u>Register</u></b>	<code>POST</code> <code>/api/WebhookRegistrations</code>	Registers for all webhook events delivery
<b><u>Registered</u></b>	<code>GET</code> <code>/api/WebhookRegistrations</code>	Returns a list of all webhook registrations by event delivery or by polling the system
<b><u>Registered by ID</u></b>	<code>GET</code> <code>/api/WebhookRegistrations/{registrationId}</code>	Returns a webhook registration by ID
<b><u>Update by ID</u></b>	<code>PUT</code> <code>/api/WebhookRegistrations/{registrationId}</code>	Updates the details of a webhook registration by ID
<b><u>Delete by ID</u></b>	<code>DELETE</code> <code>/api/WebhookRegistrations/{registrationId}</code>	Deletes a webhook registration by ID
<b><u>Delete by IDs</u></b>	<code>DELETE</code> <code>/api/WebhookRegistrations/{registrationId}/{sourceSenderId}</code>	Deletes a webhook registration by ID and sender ID



## 2.2.2.1. Register

---

## Endpoint: /api/WebhookRegistrations

Registers you for webhook events. There is no option to sign up for individual webhooks.

**POST**

https://pushtopaystaging.crbnj.net/api/WebhookRegistrations



Request

Response

### BODY PARAMETERS

**requestId** String **required**

The GUID that enables the application to link request with response

**callbackUrl** String **required**

The value is a URL. Webhooks are reported to this URL as a result of a triggered action. Make sure the callback URL is added to our allowlist. SSL required.

**callbackUserName** String **required**

Authentication data for webhook API calls if the server requires authentication

**callbackPassword** String **required**

Authentication data for webhook API calls if the server requires authentication

**sourceSenderId** String **required**

The GUID that identifies the merchant partner

```
{
  "requestId": "R4J39Q02DC-YP3B-3BT2-5AC9-FF0P111SLGN0",
  "callbackUrl": "string",
  "callbackUserName": "string",
  "callbackPassword": "string"
}
```

## Responses

● 200

```
{
  "registrationId": "hk76vsmh-k7hs-4ch6-7mks-nj76clk8h0xk",
  "requestId": "7gmlpo54-sn87-2nlg-8clk-fhk90xmhu12x",
  "callbackUrl": "www.yoururl.com",
  "callbackUserName": "jroll",
  "callbackPassword": "1234",
  "merchantId": "hk7xmlhs-7lkc-a0cn-aluv-hk7xkg67c141",
  "isActive": true,
  "deactivatedAt": "2020-12-16T09:31:35.622Z",
  "isInErrorState": true,
  "errorStateReason": "string",
  "detectedErrorStateAt": "2020-12-16T09:31:35.622Z"
}
```

## 2.2.2.2. Registered

---



## Endpoint: /api/WebhookRegistrations

Returns the webhook events you are registered for

GET

https://pushtopaystaging.crbnj.net/api/WebhookRegistrations



Request

Response

### BODY PARAMETERS

**registrationId** String optional

Unique registration ID for a webhook

**requestId** String optional

The GUID that enables the application to link request with response

**callbackUrl** String optional

The value is a URL. Webhooks are reported to this URL as a result of a triggered action. Make sure the callback URL is added to Cross River's allowlist. SSL required.

**callbackUserName** String optional

Authentication data for webhook API calls if the server requires authentication

**callbackPassword** String optional

Authentication data for webhook API calls if the server requires authentication

**merchantId** String optional

Identifier of the merchant partner

**isActive** Boolean optional

True if it is active, otherwise false

**deactivatedAt** String optional

Returns the date and time a webhook was deactivated

**isInErrorState** Boolean optional

True if the webhook is in error status, otherwise false

**errorStateReason** String optional

Reason the registration is in an error state

**detectedErrorStateAt** String optional

The moment that an error was detected. This is important information when troubleshooting a problem.

```
{
  "requestId": "R4J39Q02DC-YP3B-3BT2-5AC9-FF0P111SLGN0",
  "callbackUrl": "string",
  "callbackUserName": "string",
  "callbackPassword": "string"
}
```

## Responses

● 200

```
{
  "registrationId": "hk76vsmh-k7hs-4ch6-7mks-nj76clk8h0xk",
  "requestId": "7gmlpo54-sn87-2nlg-8clk-fhk90xmhu12x",
  "callbackUrl": "www.yoururl.com",
  "callbackUserName": "jroll",
  "callbackPassword": "1234",
  "merchantId": "hk7xmlhs-7lkc-a0cn-aluv-hk7xkg67c141",
  "isActive": true,
  "deactivatedAt": "2020-12-16T09:31:35.622Z",
  "isInErrorState": true,
  "errorStateReason": "string",
  "detectedErrorStateAt": "2020-12-16T09:31:35.622Z"
}
```

### 2.2.2.3. Registered by ID

---

## Endpoint: /api/WebhookRegistrations/{registrationId}

Returns a webhook by registration ID

GET

<https://pushtopaystaging.crbnj.net/api/WebhookRegistrations/{registrationId}> 

Request

Response

### PATH PARAMS

**registrationId** String **required**

Unique registration ID for a webhook

**sourceSenderId** String **required**

The GUID that identifies the merchant partner

```
curl -X GET --header  
'Accept: application/json'  
--header 'Authorization: {Bearer token}'  
'https://pushtopaystaging.crbnj.net/api/WebhookRegistrations/5cfbc494-fcfc'
```

## Responses

● 200



```
{  
  "registrationId": "hk76vsmh-k7hs-4ch6-7mks-nj76clk8h0xk",  
  "requestId": "7gmlpo54-sn87-2nlg-8clk-fhk90xmhu12x",  
  "callbackUrl": "www.yoururl.com",  
  "callbackUserName": "jroll",  
  "callbackPassword": "1234",  
  "merchantId": "hk7xmlhs-7lkc-a0cn-aluv-hk7xkg67c14l",  
  "isActive": true,  
  "deactivatedAt": "2020-12-16T09:31:35.622Z",  
  "isInErrorState": true,  
  "errorStateReason": "string",  
  "detectedErrorStateAt": "2020-12-16T09:31:35.622Z"  
}
```

#### 2.2.2.4. Update by ID

---

## Endpoint: /api/WebhookRegistrations/{registrationId}

Updates the web address (URL) for webhooks you are registered to according to your registration ID

PUT

https://pushtopaystaging.crbnj.net/api/WebhookRegistrations/{registrationId} 

Request

Response

### BODY PARAMETERS

**registrationId** String **required**

Unique registration ID for a webhook

**requestID** String **optional**

The GUID that enables the application to link request with response

**callbackUrl** String **optional**

The value is a URL. Webhooks are reported to this URL as a result of a triggered action. Make sure the callback URL is added to Cross River's allowlist. SSL required.

**callbackUserName** String **optional**

Authentication data for webhook API calls if the server requires authentication

**callbackPassword** String **optional**

Authentication data for webhook API calls if the server requires authentication

**merchantId** String **optional**

Identifier of the merchant partner



**IsActive** Boolean optional

True if it is active, otherwise false

**deactivatedAt** String optional

Returns the date and time a webhook was deactivated. In this case, the date and time are in this format: yyyy-mm-ddThh:mm:ss[.mmm]

**isInErrorState** Boolean optional

True if the webhook is in error status, otherwise false

**errorStateReason** String optional

Reason the registration is in an error state

**detectedErrorStateAt** String optional

The moment that an error was detected. This is important information when troubleshooting a problem.. In this case, the date and time are in this format: yyyy-mm-ddThh:mm:ss[.mmm]

```
{
  "requestId": "2c0d9149-9de0-465e-a556-4caaa8830503",
  "sourceSenderId": "b4e85c02-afcc-4901-bf4d-6d3d954fa2b9",
  "callbackUrl": "string",
  "callbackUserName": "string",
  "callbackPassword": "string",
  "isActive": true
}
```

## Responses

● 200

```
{
  "registrationId": "2c0d9149-9de0-465e-a556-4caaa8830503",
  "requestId": "1e5e1ddb-dd91-4228-942d-ce45a6fd548d",
  "callbackUrl": "string",
  "callbackUserName": "string",
  "callbackPassword": "string",
  "subMerchantId": "2c0d9149-9de0-465e-a556-4caaa8830503",
  "isActive": true,
  "deactivatedAt": "2023-01-26T10:15:49.097Z",
  "isInErrorState": true,
  "errorStateReason": "string",
  "detectedErrorStateAt": "2023-01-26T10:15:49.097Z"
}
```

## 2.2.2.5. Delete by IDs

---

## Endpoint: /api/WebhookRegistrations/{registrationId}/{sourceSenderId

Deletes a webhook registration using its registration and source sender IDs

**DELETE**

https://pushtopaystaging.crbnj.net/api/WebhookRegistrations/{registrationId}/{sourceSenderId} 

**Request**

Response

### BODY PARAMETERS

**requestId** String **required**

The GUID that enables the application to link request with response

**callbackUrl** String **required**

The value is a URL. Webhooks are reported to this URL as a result of a triggered action. Make sure the callback URL is added to Cross River's allowlist. SSL required.

**callbackUserName** String **optional**

Authentication data for webhook API calls if the server requires authentication

**callbackPassword** String **optional**

Authentication data for webhook API calls if the server requires authentication

**SourceSenderId** String **optional**

The GUID that identifies the merchant partner

```
{
  "requestId": "fhak78ch-9f7x-fh86-1nv8-hd84v0n4a82d",
  "callbackUrl": "string",
  "callbackUserName": "string",
  "callbackPassword": "string",
  "sourceSenderId": "9b52a9b1-759e-4665-98c7-18e61015f77a"
}
```

## Responses

● 200



```
{
  "registrationId": "fhk76smv-12a4-21d4-b45p-nbd98l05sc78",
  "requestId": "fhak78ch-9f7x-fh86-1nv8-hd84v0n4a82d",
  "callbackUrl": "www.yoururl.com",
  "callbackUserName": "jroll",
  "callbackPassword": "1234",
  "merchantId": "12345qfs8-1a34-n5v5-2kl3-234hko891cvn",
  "isActive": false,
  "deactivatedAt": "2020-12-16T09:31:35.617Z",
  "isInErrorState": true,
  "errorStateReason": "string",
  "detectedErrorStateAt": "2022-12-16T09:31:35.617Z"
}
```

## 2.2.2.6. Delete by ID

---

## Endpoint: /api/WebhookRegistrations/{registrationId}

Deletes a webhook registration by its registration ID

**DELETE**

https://pushtopaystaging.crbnj.net/api/WebhookRegistrations/{registrationId} 

**Request**

Response

### BODY PARAMETERS

**requestId** String **required**

The GUID that enables the application to link request with response

**callbackUrl** String **required**

The value is a URL. Webhooks are reported to this URL as a result of a triggered action. Make sure the callback URL is added to Cross River's allowlist. SSL required.

**callbackUserName** String **optional**

Authentication data for webhook API calls if the server requires authentication

**callbackPassword** String **optional**

Authentication data for webhook API calls if the server requires authentication

**sourceSenderId** String **optional**

The GUID that identifies the merchant partner

```
{
  "requestId": "fhak78ch-9f7x-fh86-1nv8-hd84v0n4a82d",
  "callbackUrl": "string",
  "callbackUserName": "string",
  "callbackPassword": "string",
  "sourceSenderId": "3772e240-bc7c-45ec-b072-ee2949578481"
}
```

## Responses

● 200

```
{
  "registrationId": "fhk76smv-12a4-21d4-b45p-nbd98l05sc78",
  "requestId": "fhak78ch-9f7x-fh86-1nv8-hd84v0n4a82d",
  "callbackUrl": "www.yoururl.com",
  "callbackUserName": "jroll",
  "callbackPassword": "1234",
  "merchantId": "12345qfs8-1a34-n5v5-2kl3-234hko891cvn",
  "isActive": false,
  "deactivatedAt": "2020-12-16T09:31:35.617Z",
  "isInErrorState": true,
  "errorStateReason": "string",
  "detectedErrorStateAt": "2022-12-16T09:31:35.617Z"
}
```



## 2.2.3. Webhook events

The table below describes the different Card Payments webhook events you can register for. Where relevant, additional information is provided for some of the events.

Refer to Card Payments [webhook events](#) for some examples.

Call	Description
<code>CardAuthorized</code>	Reports: <ol style="list-style-type: none"><li>1. When a card authorization attempt is completed and the status of the card.</li><li>2. When a card was signed up via an iFrame.</li></ol>
<code>CardStatusChanged</code>	Reports if a card changed status to active or inactive
<code>TransactionCompleted</code>	Reports the completion of a transaction request. This webhook is sent only 1 time. If a transaction has changed after the initial execution, the <code>TransactionStatusChanged</code> webhook will be sent.
<code>TransactionStatusChanged</code>	Reports the changed status of a transaction
<code>AccountCreditedReceived</code>	The COS account of the merchant is credited with the amount mentioned on the payload at the timestamp mentioned on the payload.

## 2.3. Lending

*Hooks* is a notification system, used by Arix, to update you on the status of your loan. *Hooks* report to your system with real-time notifications when an event happens on Arix. You have to register to receive hooks for each relevant event type by configuring the destination and delivery method, most commonly webhooks. When that event occurs, a Hook is triggered and reports the updates to your system via the chosen delivery method.

For example, if a rail status changes to *returned*, a `RailUpdated` hook event would be triggered, and reported to the partner notifying them of the event. This notification will only be sent out, if you have previously registered to receive this type of event ( `RailUpdated` ), and the registration is active.

### IMPORTANT

To access the links below you must have the following IP addresses allowlisted:

Sandbox - 66.206.202.39 , 66.206.202.12

Production -66.206.202.62 , 66.206.202.15

- Swagger: <https://lendingsandbox.crbcos.com/hooks/swagger/index.html>
- Base URL for sandbox: <https://lendingsandbox.crbcos.com/>
- Sandbox UI: <https://lendingappsandbox.crbcos.com/hooks/registrations>
- Production UI: <https://lendingapp.crbcos.com/hooks/registrations>
- API Scope:
  - CosLending:Hooks:stg for sandbox
  - CosLending:Hooks:prd for production

By default, a webhook that fails will retry 3 times with a delay interval of 5 minutes. Different Intervals can be configured for a registration by the CR support team.

Events can also be sent out again manually, if necessary, via an API request.

# Sample Events

## LoanStatusUpdate

This event is reported every time the status of a loan is changed.

JSON



```
{
  "Id": 0,
  "LoanId": "3f07773b-5ba5-43e6-bf30-aea200361dad",
  "Status": 104,
  "TimeStamp": "2022-05-30T03:17:43.6233616+00:00",
  "DateInserted": "0001-01-01T00:00:00+00:00"
}
```

## ComplianceLoanFailed

This event is reported every time the loan fails to pass all compliance checks.

JSON



```
{
  "MPLId": "xxx",
  "LoanId": "...guid...",
  "CreateDate": "date",
  "FailedRulesReasons": [
    {
      "RuleName": "rule name",
      "Rule": "some string",
      "Data": "some string",
      "Result": true,
      "FailedComplianceID": 67
    }
  ]
}
```

## RailUpdated

This event is reported when there is a status update to a rail that was requested

JSON



```
{
  "Id":0,
  "RailId":"...ID of rail that is received when rail created...",
  "LoanId":"fd331267-6ac2-4818-9d0f-ab2f00747b61",
  "RailType":8,
  "AmountFunded":8500.0,
  "RailTransactionId":"...railid from cos...",
  "FundResult": int(this is enum RailResultType),
  "Message":"Description of rail created if no error returned",
  "ProcessedAt":"2019-12-26T07:05:09Z",
  "TransactionGroupId":"9dde776d-0c04-4bc8-9c9c-e0ad9a83c447"
}
```

### 3. Customer management

---

#### Customer onboarding

Use Cross River APIs to manage your customer information (called a customer record). You need a customer record, for example, to open a deposit account or create a debit card.

**Get started** with Cross River APIs.

Once you have access to our **sandbox** you can access these API endpoints in the `core` module through **Swagger**.

#### GET calls

- Add query parameters to filter the response the API returns to general GET calls.
- Use **pagination** to control presentation of your results.

This table presents APIs that we describe in detail.

Action	API	Description
<b><u>Add customer</u></b>	POST /core/v1/cm/customers	Create new customer record
<b><u>Customer by ID</u></b>	GET /core/v1/cm/customers/{id}	Retrieve a customer record by id
<b><u>Add beneficial owner</u></b>	POST /core/v1/cm/customers/{customerId}/beneficial-owners	Add a beneficial owner
<b><u>Add ID details</u></b>	POST /core/v1/cm/customers/{customerId}/identifications	Add ID details
<b><u>Update address</u></b>	POST /core/v1/cm/customers/{customerId}/addresses	Update customer address
<b><u>Update phone</u></b>	POST /core/v1/cm/customers/{customerId}/phones	Update customer phone number
<b><u>Update email</u></b>	POST /core/v1/cm/customers/{customerId}/email	Update customer email
<b><u>Update name</u></b>	PUT /core/v1/cm/customers/{customerId}/name	Update customer name

## Tutorials

- **Onboard a customer**: Create a basic customer record in our system. Most products and services require a customer record for each of your customers.
- **Add a beneficial owner**: Associate a beneficial owner with a business customer record or use the beneficial owner field for BIN sponsorship.
- **Add ID details**: Add the details of an identification document such as a driver's license or passport to a customer record.



## 3.1. APIs

### 3.1.1. Add customer

---



## Endpoint: /core/v1/cm/customers

Creates a new customer record. The customer record is the primary resource that contains customer information and supports the classification of Personal or Business. Once onboarded, a customer can be associated with one or more accounts.

**POST**

https://sandbox.crbcos.com/core/v1/cm/customers



**Request**

Response

### BODY PARAMETERS

**partnerID** String **required**

Your partner ID in the Cross River system

**name ▶** Object optional

Customer name in detail

**classification** String **required**

The classification type of the account:

Personal (an individual)

Business (a legal entity)

**profile ▶** Object **required**

Customer banking-relevant information

**dueDiligence ▶** Object optional

Details for due diligence compliance for an individual customer. To be used when required. Some customer endpoints include a dueDiligence resource that may or may not be required when onboarding personal customers. Your assigned compliance liaison communicates requirements for any Due Diligence fields during your onboarding process.

**businessDueDiligence ▶**

Object

optional

Details for due diligence compliance for a business entity. To be used when required.

**clientIdentifier**

String

optional

Use this attribute to add your own unique identifying string to a payment call or COS record. This attribute is useful for idempotency purposes.  
50 characters maximum

**primaryAddress ▶**

Object

optional

Main address for the customer

**primaryPhone ▶**

Object

optional

Main phone number for the customer

**primaryEmail ▶**

Object

optional

Main email address for the customer

**primaryIdentification ▶**

Object

optional

Identification metadata of customer-identifying documents, such as a driver's license or passport



Curl



Node.js



Python



Ruby



Go



```
curl --location 'https://sandbox.crbcos.com/core/v1/cm/customers' \
--header 'Content-Type: application/json' \
--data-raw '{
  "partnerId": "1101a33d-fbd9-4b9e-b169-b29500f12407",
  "name": {
    "firstName": "Brandon",
    "lastName": "Sanderson"
  },
  "classification": "Personal",
  "profile": {
    "reg0": false,
    "citizenshipCountryCode": "US",
    "politicallyExposedPerson": false,
    "enableBackupWithholding": false,
    "backupWithholdingPercent": 0,
    "taxIdType": "Ssn",
    "taxId": "",
    "birthDate": "1980-03-06",
    "riskRating": "Low",
    "privacyOptOut": true
  },
  "dueDiligence": {
    "employmentStatus": "Employed",
    "occupation": "Technology",
    "occupationDescription": "Support",
    "industry": "Finance",
    "incomeSource": "Salary",
    "incomeState": "NJ",
    "incomeCountryCode": "US",
    "employer": "Cross River",
    "employerState": "NJ",
    "employerCountryCode": "US",
    "sourceOfFunds": "Salary",
    "wealthSource": "Salary",
    "wealthSourceDescription": "Salary",
    "annualIncome": 4500000
  },
  "primaryAddress": {
    "addressType": "Home",
    "classification": "Residential",
    "isPrimary": true,
    "street1": "101 Main St",
    "street2": "Apt 2",
    "city": "New York",
```

## Responses

● 200



```
{
  "id": "6c545b91-f401-4308-bad3-b300007b7885",
  "cifNumber": "245085933838",
  "classification": "Personal",
  "status": "Active",
  "ofac": "Pending",
  "pepScan": "Pending",
  "internalList": "Pending",
  "name": {
    "firstName": "Brandon",
    "lastName": "Sanderson",
    "fullName": "Brandon Sanderson"
  },
  "profile": {
    "reg0": false,
    "citizenshipCountryCode": "US",
    "politicallyExposedPerson": false,
    "enableBackupWithholding": false,
    "backupWithholdingPercent": 0,
    "taxIdType": "Ssn",
    "taxId": "312860737",
    "birthDate": "1980-03-06",
    "riskRating": "Low",
    "privacyOptOut": true
  },
  "createdAt": "2025-06-18T03:29:32.6027543-04:00",
  "lastModifiedAt": "2025-06-18T03:29:32.6027543-04:00",
  "partnerId": "1101a33d-fbd9-4b9e-b169-b29500f12407",
  "dueDiligence": {
    "employmentStatus": "Employed",
    "occupation": "Technology",
    "occupationDescription": "Support",
    "industry": "Finance",
    "incomeSource": "Salary",
    "incomeState": "NJ",
    "incomeCountryCode": "US",
    "employer": "Cross River",
    "employerState": "NJ",
    "employerCountryCode": "US",
    "sourceOfFunds": "Salary",
    "wealthSource": "Salary",
    "wealthSourceDescription": "Salary",
    "annualIncome": 4500000
  },
  "primaryAddress": {
```

"id": "b2fb4cd1-aaa7-47aa-9113-b300007b7885".

### 3.1.2. Customer by ID

---



## Endpoint: /core/v1/cm/customers/{id}

Retrieves a specific customer record by customerId

GET

https://sandbox.crbcos.com/core/v1/cm/customers/{id}



Request

Response

### QUERY PARAMETERS

**Id**

String

**required**

The customerId of the customer record being updated



Curl



Node.js



Python



Ruby



Go



```
curl -L 'https://sandbox.crbcos.com/core/v1/cm/customers/bbgfe09f0a-f38b-4  
-H 'Authorization: Bearer eyJhfgfgdgk9XeEhVN19RRkdxNCIsIng1dCI6IjdQREJ4dGV'
```

Responses

● 200





### 3.1.3. Add beneficial owner

---

## Endpoint: /core/v1/cm/customers/{customerId}/beneficial-owners

A beneficial owner is a person who owns 25% or more of a business. For regulatory reasons, you need to create a Beneficial Owner resource in a Business customer record for each beneficial owner of that business. Each beneficial owner will have a personal customer record that includes a unique customer ID. The ownerCustomerId attribute in the call described here references the Personal customer ID for each beneficial owner. You usually add beneficial owner resources to the business customer record immediately after onboarding a business.

**POST**

<https://sandbox.crbcos.com/core/v1/cm/customers/{customerId}/> 

**Request**

Response

### PATH PARAMS

**customerId** String **required**

The customerId of the customer record being updated

### BODY PARAMETERS

**ownerCustomerId** String **required**

Beneficial owner Cross River customer ID (provided when the beneficial owner was onboarded to COS)

**ownerTitle** String **required**

Title or role of the beneficial owner

**ownershipPercentage** Number **optional**

Percentage of the business this beneficial owner controls. Enter the percentage in real numbers. For example, if you enter 0.20 it means 0.20%. If you enter 20, it means 20%.



Curl



Node.js



Python



Ruby



Go



```
curl --location --globoff 'https://sandbox.crbcos.com/core/v1/cm/customers' \
--data '{
  "ownerCustomerId": "",
  "ownerTitle": "CEO",
  "ownershipPercentage": "26"
}'
```

## Responses

● 200● 400

```
{
  "id": "bf3fe246-626e-4960-8e2b-b300008b44e1",
  "customerId": "554fa0db-ee28-4671-8d02-b300008abb4a",
  "ownerCustomerId": "bbe09f0a-f38b-4daa-a0a2-b2ff00f4e887",
  "status": "Active",
  "ownerTitle": "CEO",
  "partnerId": "1101a33d-fbd9-4b9e-b169-b29500f12407",
  "createdAt": "2025-06-18T04:27:03.7887551-04:00",
  "lastModifiedAt": "2025-06-18T04:27:03.7887551-04:00",
  "ownershipPercentage": 26
}
```

### 3.1.4. Add ID details

---



## Endpoint: /core/v1/cm/customers/{customerId}/identifications

Update a COS customer record with physical ID metadata or other identification information.

**POST**

<https://sandbox.crbcos.com/core/v1/cm/customers/{customerId}/> 

**Request**

Response

### PATH PARAMS

**customerId** String **required**

The customerId of the customer record being updated

### BODY PARAMETERS

**isPrimary** Boolean **required**

True if this is the primary ID record for this customer. Otherwise, false

**idNumber** String **required**

The identification number being added

**idType** String **required**

The kind of ID:

- DriversLicense
- Passport
- IdCard
- SocialSecurity
- Other

**issuedDate** String **optional**

Date the ID was issued

**expDate**                      String              optional

Date the ID expires

**verifiedDate**                      String              optional

Date the ID was last verified

**issuingAuthority**                      String              optional

The official authority that issued the ID. For example, the Department of Motor Vehicles

**issuingStateOrProvince**                      String              optional

State or province where the ID was issued

**issuingCountryCode**                      String              optional

Country where the ID was issued expressed as a 2-character code



Curl



Node.js



Python



Ruby



Go



```
curl -L 'https://sandbox.crbcos.com/core/v1/cm/customers/bbe09f0a-f38b-4da
-d '{
  "isPrimary": true,
  "idNumber": "18e5b61d-fbac-4a52-9cce-3af544886199",
  "idType": "Other",
  "issuedDate": "2022-08-18",
  "expDate": "2027-08-18",
  "verifiedDate": "2025-08-18",
  "issuingAuthority": "Fintech",
  "issuingStateOrProvince": "NJ",
  "issuingCountryCode": "US"
}'
```

## Responses

200

400



```
{
  "errors": [
    {
      "code": 2234,
      "message": "Identification already exists"
    }
  ]
}
```

### 3.1.5. Update address

---

## Endpoint: /core/v1/cm/customers/{customerId}/address

The customer record is the primary resource that contains customer information. This API allows for updates to the customer address.

**POST**

<https://sandbox.crbcos.com//core/v1/cm/customers/{customerId}> 

Request

Response

### PATH PARAMS

**customerId** String **required**

The customerId of the customer record being updated

### BODY PARAMETERS

**addressType** String **required**

What this is the address of:

- Home
- Work
- Other

**classification** String **required**

What kind of address this is:

- Residential
- Business
- PoBox
- Rural
- Military
- Other

**isPrimary** Boolean **required**

True if this is the primary address record for this customer. Otherwise, false.

**street1** String **required**

Additional address information. 255 character maximum.

**street2** String **optional**

Additional address information. 255 character maximum.

**street3** String **optional**

Additional address information. 255 character maximum.

**city** String **optional**

City full name. 50 character maximum

**state** String **optional**

State name. 10 character maximum

**postalCode** String **optional**

Postal code, also known as ZIP code. 10 character maximum.

**countryCode** String **required**

2-letter ISO country code



Curl



Node.js



Python



Ruby



Go



```
package main
```

```
import (  
    "fmt"  
    "strings"  
    "net/http"  
    "io"  
)
```

```
func main() {
```

```
    url := "https://sandbox.crbcos.com/core/v1/cm/customers//addresses"  
    method := "POST"
```



### 3.1.6. Update phone

---

## Endpoint: /core/v1/cm/customers/{customerId}/phones

The customer record is the primary resource that contains customer information. This API allows for updates to the customer phone number.

**POST**

<https://sandbox.crbcos.com/core/v1/cm/customers/{customerId}/phones> 

Request

Response

### PATH PARAMS

**customerId** String **required**

The customerId of the customer record being updated

### BODY PARAMETERS

**isPrimary** Boolean **required**

True if this is the primary phone record for this customer. Otherwise, false.

**phoneType** String **required**

The phone number calls a:

- Home
- Mobile
- Work
- Fax

**phoneNumber** String **required**

Phone number with no spaces or other characters. Minimum 10, maximum 20

**extension** String **optional**

If the phone rings a switchboard, the entity's direct extension. 20 character maximum.

**notes**

String optional

Additional phone information. 255 character maximum



Curl



Node.js



Python



Ruby



Go



```
curl -L 'https://sandbox.crbcos.com/core/v1/cm/customers/bbe09f0a-f38b-4da
-d '{
  "isPrimary": true,
  "phoneType": "Work",
  "phoneNumber": "2015551212",
  "extension": "123"
}
```

## Responses

● 200

● 400



```
{
  "errors": [
    {
      "code": 2230,
      "message": "Phone number already exists"
    }
  ]
}
```

### 3.1.7. Update email

---

## Endpoint: /core/v1/cm/customers/{customerId}/emails

The customer record is the primary resource that contains customer information. This API allows for updates to the customer email.

**POST**

<https://sandbox.crbcos.com/core/v1/cm/customers/{customerId}/emails> 

**Request**

Response

### PATH PARAMS

**customerId** String **required**

The customerId of the customer record being updated

### BODY PARAMETERS

**isPrimary** Boolean **required**

True if this is the primary email record for this customer. Otherwise, false.

**emailType** String **required**

The email type is:

- Personal
- Work
- Other

**emailAddress** String **required**

Customer's primary email address. This must be a minimum of 6 characters, and a maximum of 255.



Curl



Node.js



Python



Ruby



Go



```
curl -L 'https://sandbox.crbcos.com/core/v1/cm/customers/bbe09f0a-f38b-4da
--data-raw '{
  "isPrimary": true,
  "emailType": "Work",
  "emailAddress": "bsanderson2@crossriver.com"
}'
```

## Responses

● 200● 400

```
{
  "errors": [
    {
      "code": 2227,
      "message": "Email address already exists"
    }
  ]
}
```

### 3.1.8. Update name

---

## Endpoint: /core/v1/cm/customers/{customerId}/name

The customer record is the primary resource that contains customer information. This API allows for updates to the customer name.

**PUT**

<https://sandbox.crbcos.com/core/v1/cm/customers/{customerId}/name> 

**Request**

Response

### PATH PARAMS

**customerId** String **required**

The customerId of the customer record being updated

### BODY PARAMETERS

**entityName** String **optional**

Name of the business if opening a business account. 255 character maximum.

**prefix** String **optional**

Prefix for name

**firstName** String **optional**

Customer first name. 100 characters maximum

**middleName** String **optional**

Customer middle name. 50 characters maximum

**lastName** String **optional**

Customer last name. 100 characters maximum



**suffix** String optional

Suffix for the name

**preferredName** String optional

Customer preferred name. 255 character maximum

**fullName** String optional

Customer's entire name including middle initial. 255 characters maximum

 Curl

 Node.js

 Python

 Ruby

 Go



```
curl -L -X PUT 'https://sandbox.crbcos.com/core/v1/cm/customers/4e2a0d37-3
-H 'Content-Type: application/json' ^
-H 'Authorization: Bearer token' ^
-d '{
  "firstName": "Brandon",
  "lastName": "Sanderson"
}'
```

## Responses

● 200



```
{
  "firstName": "Brandon",
  "lastName": "Sanderson",
  "fullName": "Brandon Sanderson"
}
```

## 3.2. Webhook events

When you work with Cross River customer management, you register for specific webhooks which have a specific event format.

This table presents common Customer Management webhook events that are described in detail below.

Event Name	Description
<u>Core.Customer.Onboarded</u>	New customer was onboarded
<u>Core.Customer.Address.Added</u>	Customer address was added
<u>Core.Customer.Phone.Added</u>	Customer phone number was added
<u>Core.Customer.Identification.Added</u>	Customer identification was added
<u>Core.Customer.Email.Added</u>	Customer email was added

## Event examples

### Core.Customer.Onboarded



Sample Event



```
{
  "id": "97b574c2-08d4-42c6-884c-b2e201267681",
  "eventName": "Core.Customer.Onboarded",
  "status": "Pending",
  "partnerId": "1e5d3f04-ae24-4af6-9e30-aecf012b99dd",
  "createdAt": "2025-05-19T13:52:06.393+00:00",
  "resources": [
    "core/v1/cm/customers/61dc4d4c-3b2c-47f3-b935-b2e201266e06"
  ],
  "details": [
    {
      "customerId": "61dc4d4c-3b2c-47f3-b935-b2e201266e06",
      "clientIdentifier": "35ba6120-f860-4caa-9479-b5aaebeca49d"
    }
  ]
}
```

## Core.Customer.Address.Added



Sample Event



```
{
  "id": "6d147ea6-2628-461f-9aa6-b2e20126769e",
  "eventName": "Core.Customer.Address.Added",
  "status": "Pending",
  "partnerId": "1e5d3f04-ae24-4af6-9e30-aecf012b99dd",
  "createdAt": "2025-05-19T13:52:06.483+00:00",
  "resources": [
    "core/v1/cm/customers/61dc4d4c-3b2c-47f3-b935-b2e201266e06/addresses/e30b"
  ],
  "details": [
    {
      "addressId": "e30bd094-54c9-4388-a8ca-b2e201266e0b",
      "customerId": "61dc4d4c-3b2c-47f3-b935-b2e201266e06"
    }
  ]
}
```

## Core.Customer.Phone.Added



Sample Event



```
{
  "id": "27834c98-9474-4298-bd2c-b2e20126767f",
  "eventName": "Core.Customer.Phone.Added",
  "status": "Pending",
  "partnerId": "1e5d3f04-ae24-4af6-9e30-aecf012b99dd",
  "createdAt": "2025-05-19T13:52:06.383+00:00",
  "resources": [
    "core/v1/cm/customers/61dc4d4c-3b2c-47f3-b935-b2e201266e06/phones/c46bd68",
  ],
  "details": [
    {
      "phoneId": "c46bd68a-8c07-4548-b229-b2e201266e1e",
      "customerId": "61dc4d4c-3b2c-47f3-b935-b2e201266e06"
    }
  ]
}
```

## Core.Customer.Identification.Added



Sample Event



```
{
  "id": "431a3c49-429f-4001-a1fc-b2e201267683",
  "eventName": "Core.Customer.Identification.Added",
  "status": "Pending",
  "partnerId": "1e5d3f04-ae24-4af6-9e30-aecf012b99dd",
  "createdAt": "2025-05-19T13:52:06.483+00:00",
  "resources": [
    "core/v1/cm/customers/61dc4d4c-3b2c-47f3-b935-b2e201266e06/identification",
  ],
  "details": [
    {
      "identificationId": "6420d9e4-3abe-4be5-ae19-b2e201266e14",
      "customerId": "61dc4d4c-3b2c-47f3-b935-b2e201266e06"
    }
  ]
}
```

# Core.Customer.Email.Added



Sample Event



```
{
  "id": "3d862cdc-5fc7-45c4-b40d-b2e20126769b",
  "eventName": "Core.Customer.Email.Added",
  "status": "Pending",
  "partnerId": "1e5d3f04-ae24-4af6-9e30-aecf012b99dd",
  "createdAt": "2025-05-19T13:52:06.483+00:00",
  "resources": [
    "core/v1/cm/customers/61dc4d4c-3b2c-47f3-b935-b2e201266e06/emails/7ba8f19d-f224-4e83-926e-b2e201266e0f",
  ],
  "details": [
    {
      "emailId": "7ba8f19d-f224-4e83-926e-b2e201266e0f",
      "customerId": "61dc4d4c-3b2c-47f3-b935-b2e201266e06"
    }
  ]
}
```

### 3.3. Response and request codes

---

## Customer entity types

Entity Type
Limited Liability Company
Sole Proprietorship
Corporation
Not for Profit
Investment/Hedge/Private Equity Fund
Trust
Special Purpose Vehicle/Special Purpose Entity
Private Investment Company/Private Holding Company

## 3.4. Error codes

---

Code	Description
1000	General exception
1200	DateFormed required
1201	DateFormed cannot be in the future
1202	BirthDate required
1203	BirthDate cannot be in the future
1204	EntityName required
1205	FirstName not allowed for classification
1206	MiddleName not allowed for classification
1207	LastName not allowed for classification
1208	Prefix not allowed for classification
1209	Suffix not allowed for classification
1210	PreferredName not allowed for classification
1211	FirstName required
1212	LastName required
1213	EntityName not allowed for classification
1214	Name required
1215	PoliticallyExposedPerson required
1216	PoliticallyExposedPerson not allowed for classification
1217	BirthDate not allowed for classification
1218	DateFormed not allowed for classification
1219	RegO required
1220	RegO not allowed for classification
1221	PrimaryOwnerCustomerId required
1222	PrimaryOwnerCustomerId not allowed for classification
1223	EntityType not allowed for classification



Code	Description
1224	OwnershipType not allowed for classification
1225	TaxIdType not allowed for classification
2000	General exception
2001	Account not active
2002	Status cannot be set to closed manually
2003	Account not open
2004	Account is closed
2005	Account is not closed
2006	Account cannot be closed until active holds removed
2007	Account cannot be closed until active memo posts removed
2008	Account cannot be closed until all completed transactions have been applied and balance is zero
2009	Account cannot be closed until sub accounts are closed
2010	Max total accounts exceeded for product
2011	Max active accounts exceeded for product
2012	Max daily accounts exceeded for product
2013	Account already exists
2014	Account number cannot be generated
2015	Account generation attempts exceeded
2016	Invalid account number prefix
2017	Invalid account number length
2018	Customer not assigned to same partner as product
2019	Customer classification not allowed for product
2020	Customer must have an address associated
2021	Customer must have a phone number associated

Code	Description
2022	Customer must be cleared for OFAC prior to account opening
2100	Sub account not found
2101	Sub account not active
2102	Cannot modify implicit sub account
2103	Sub account already exists
2104	Sub account is closed
2105	Sub account is not closed
2106	Master account required
2107	Master account not open
2108	Master account not active
2109	Master account not found
2110	Beneficiary required
2111	Sub account total limit has been reached for master account
2112	Sub accounts are not enabled for product
2113	Sub account number cannot be generated
2114	Sub account generation attempts exceeded
2115	Invalid sub account number prefix
2116	Invalid sub account number length
2117	Sub account cannot be closed until all completed transactions have been applied and balance is zero
2118	Sub account daily limit has been reached for master account
2119	Sub account active limit has been reached for master account
2200	Customer not found
2201	Customer not active
2202	Entity not active

Code	Description
2203	Address not active
2204	Cannot remove primary address
2205	DateFormed required
2206	DateFormed cannot be in the future
2207	BirthDate required
2208	BirthDate cannot be in the future
2209	Primary owner customer not found
2210	Primary owner customer not active
2211	Primary owner customer must be a person
2212	Entity and primary owner must be under same partner
2213	Entity and primary owner must be different customers
2214	Entity customer must be a business
2215	Owner customer not found
2216	Owner customer not active
2217	Owner customer must be a person
2218	Entity and owner customers must be under same partner
2219	Entity and owner must be different customers
2220	Owner relationship already exists for entity customer
2221	Unable to deactivate customer due to established account relationship(s)
2222	Partner not found or is inactive
2223	An active customer with this tax ID already exists for partner
2224	Cannot remove primary identification
2225	Cannot remove primary phone
2226	Invalid customer entity type
2350	Hold not found

Code	Description
2351	Hold is inactive
2352	Invalid hold amount
2353	Expiration date cannot be in the past
2375	Restriction not found
2376	Restriction not active
2377	Restriction policy in request must match restriction policy associated with change
2378	Restriction policy approval requires dual control
2379	Restriction policy not found
2380	Restriction policy not active
2381	Restriction policy max restrictions exceeded
2382	Restriction policy name already exists
2383	Pending restriction policy change with name already exists
2384	Restriction already inactive
2385	Restriction policy change not approved
2386	Restriction policy change not pending
2425	Interest accrual not found
2426	Start date must be before end date
2427	Amount must be greater than or equal to zero
2428	Payments not equal to zero must be posted to core
2450	Memo post not found
2451	Memo post not active
2452	TransactionCode is invalid
2475	Product not found
2476	Product not active

Code	Description
2477	Product change not approved
2478	Product change not pending
2479	Product in request must match product associated with change
2480	Classification not allowed for product type
2481	Product type invalid
2482	Product name already exists
2483	Unknown account type for product
2484	Pending product change with name already exists
2485	Interest Accrual not allowed for account type
2487	Statements not allowed for account type
2488	Product change approval requires dual control
2525	Relationship not found
2526	Relationship not active
2527	Primary relationship cannot be removed
2528	Relationship not allowed for account type
2529	RelationshipType not allowed for account classification
2530	IsTaxReportingOwner cannot be enabled for relationship type
2531	Customer must be classified as a person
2532	Customer relationship already exists
2533	Customer must be under same partner as account
2534	Customer must be cleared for OFAC prior to creating relationship
2550	Settlement not found
2551	Daily settlement must be pending
2552	Daily settlement must not be complete
2553	Daily settlement must be pre-settling and all interest must be processed

Code	Description
2554	All accounts must be settled before daily settlement can be completed
2575	Snapshot not found
2576	Snapshot account number does not match
2577	Snapshot partner does not match
2600	Statement not found
2601	Statement not completed
2602	Statement is closed
2603	AccountNumber must match statement
2604	Unable to generate statement for account type
2625	Transaction not found
2626	Debit account not found
2627	Debit account not active
2628	Credit account not found
2629	Credit account not active
2630	Debit and credit accounts must be different
2631	Debit account type not allowed
2632	Credit account type not allowed
2633	General ledger transfers require auto approval permission
2634	Transaction must be pending
2635	Transaction not effective
2636	Transaction must be under review
2637	Transaction authorization not found
2638	Transaction approval requires dual control
2639	Invalid transaction status
2640	PartnerId not accessible

Code	Description
2641	PartnerId is required for user type
2642	TransactionCode is invalid
2643	Max transaction flags exceeded
2644	Invalid transaction flag
2645	Transaction flags not allowed
2646	Transaction filter required
2647	Transaction trace number required
2648	Transaction proposal required
2649	Transaction amount must be greater than zero
2650	Transaction auth status invalid
2700	Accounting export is not processing

## 4. Accounts

---

### Accounts concepts

Cross River offers a set of APIs for account management in our banking core.

**Get started** with Cross River APIs.

Once you have access to our **sandbox** you can access these API endpoints in the `core` module, `DDA` section, through **Swagger**.

### GET calls

- Add query parameters to filter the response the API returns to general GET calls.
- Use **pagination** to control presentation of your results.

## Account management

This table presents general account management APIs that we describe in detail.

### IMPORTANT

Only pull tax documents as needed. If you want to pull a large number of documents, speak to your Relationship Manager.



Action	API Call	Description
<a href="#"><u>Open account</u></a>	POST /core/v1/dda/accounts	Creates an account based on the product ID for that type of account. This includes CD (time deposit) accounts.
<a href="#"><u>Tax document list</u></a>	GET /core/v1/dda/accounts/{accountNumber}/tax-documents/	Gets a list of tax documents for a given account, including IDs for each
<a href="#"><u>Tax document</u></a>	GET /core/v1/dda/accounts/{accountNumber}/tax-documents/{id}	Retrieves a PDF of a specific tax document

## Time deposit (CD)

This table presents time deposit (CD) APIs that we describe in detail.

Action	API Call	Description
<a href="#"><u>Early funds withdrawal</u></a>	POST /core/v1/dda/accounts/{accountNumber}/time-deposit/early-withdraw	Requests to withdraw the funds from a time deposit account before the account reaches maturity
<a href="#"><u>CD activity</u></a>	GET /core/v1/dda/accounts/{accountNumber}/time-deposit	Returns information about the account activity of a specific time deposit account
<a href="#"><u>CD penalty fee</u></a>	GET /core/v1/dda/accounts/{accountNumber}/time-deposit/penalty-fee	Returns the projected penalty fee if you make an early withdrawal to an active time deposit account

## Deposit sweeps

Once you have access to our [sandbox](#) you can access these API endpoints in the `SweepsIntraFi` module through [Swagger](#).

This table presents APIs we describe in detail.

Action	API Call	Description
<a href="#">Account details</a>	<code>GET/SweepsIntraFi/v1/accounts/{accountNumber}</code>	GET call to get account details of a COS account enrolled in Sweeps
<a href="#">Exclude bank/account</a>	<code>PUT/SweepsIntraFi/v1/accounts/{accountNumber}</code>	PUT call to exclude specific bank or opt out specific account from Sweeps.
<a href="#">Identify bank</a>	<code>GET/SweepsIntraFi/v1/banks/{bankNumber}</code>	GET call to get bank name for a specific BankID
<a href="#">Bank list</a>	<code>GET/SweepsIntraFi/v1/banks</code>	GET call to get a list of all banks

## Subledgers

In subledger API endpoints, subledgers are referred to as `subaccounts` , For example, `POST /core/v1/dda/subaccounts` creates a new subledger.

This table presents the subledger management APIs we describe in detail.

Action	API Call	Description
<a href="#">Create subledger</a>	<code>POST /core/v1/dda/subaccounts</code>	Creates a new subledger
<a href="#">Subledger info</a>	<code>GET /core/v1/dda/subaccounts/{accountNumber}</code>	Returns information on a specific subledger account

## Related topics

- Webhook events

# Tutorials

- Open an account: Create a deposit account.
- Open a subledger: Create a subledger (virtual account).
- Withdraw CD funds early: Withdraw funds from a certificate of deposit before it reaches maturity.

## 4.1. **Account mgmt APIs**

### 4.1.1. Open account

---

## Endpoint: /v1/dda/accounts

Creates a deposit account based on the product ID for that type of account.

**POST**

https://sandbox.crbcos.com/core/v1/dda/accounts



**Request**

Response

### BODY PARAMETERS

**customerId**

String

**required**

The unique ID assigned to a customer when the customer record is created. You need this ID to take action for a specific customer. For example, when opening an account or adding an address to a customer record.

The ID is in GUID format.

**productId**

String

**required**

ID in GUID format of your specific product type on which the account is based. Provided by Cross River.

**title**

String

**required**

The name on the account. Usually this is the account holder name, but not always. 255 characters maximum.

**statementAddress** ▶

Object

**required**

Mailing address for paper statement delivery

**clientIdentifier**

String

**optional**

Use this attribute to add your own unique identifying string to a payment call or COS record. This attribute is useful for idempotency purposes.

**rateCard**

String

optional

ID of the account's assigned, pre-configured interest rate profile, in GUID format



Curl



Node.js



Python



Ruby



Go



```
curl --location 'https://sandbox.crbcos.com/core/v1/dda/accounts' \
--header 'Idempotency-key: 025aaf3e-87e5-46e6-ba60-2071c3a69be8' \
--header 'Content-Type: application/json' \
--header 'Authorization: Bearer <token>' \
--data '{
  "customerId": "53c34cf0-4cd2-4768-937b-b30000dffdca",
  "productId": "83bed086-8182-4151-a1e3-af5b01362783",
  "title": "Jerry Penn",
  "statementAddress": {
    "street1": "12 Main St",
    "city": "New York",
    "state": "NY",
    "postalCode": "10001",
    "countryCode": "US"
  }
}
```

## Responses

● 200



```
{
  "accountNumber": "2819503463",
  "status": "Active",
  "accountType": "Deposit",
  "ledgerType": "Passthrough",
  "classification": "Personal",
  "productType": "Checking",
  "productId": "83bed086-8182-4151-a1e3-af5b01362783",
  "title": "Jerry Penn",
  "currentBalance": 0,
  "holdAmount": 0,
  "transactionCount": 0,
  "availableBalance": 0,
  "creditLimit": 0,
  "overdraftFundingEnabled": false,
  "overdraftFundingThreshold": 0,
  "currency": "usd",
  "customerId": "53c34cf0-4cd2-4768-937b-b30000dffdca",
  "statementAddress": {
    "street1": "12 Main St",
    "city": "New York",
    "state": "NY",
    "postalCode": "10001",
    "countryCode": "US"
  },
  "openedAt": "2025-06-18T09:41:00.9073289-04:00",
  "lastMaintenanceAt": "2025-06-18T09:41:00.9073289-04:00",
  "partnerId": "cd9c12f4-7691-424a-b38b-af5b0134c611",
  "lastModifiedAt": "2025-06-18T09:41:00.9073289-04:00"
}
```



## 4.1.2. Tax document list

---

## Endpoint: /core/v1/dda/accounts/{accountNumber}/tax-documents/

Use a specific account number to returns a list of the tax documents associated with the specified account based on query criteria. The list includes each document's unique ID, which you need to pull a specific document. The response values below are repeated for each document in the list.

Once you have the document ID use GET

/core/v1/dda/accounts/{accountNumber}/tax-documents/{id} to pull an individual document.

GET

<https://sandbox.crbcos.com/core/v1/dda/accounts/{accountNum> 

Request

Response

### QUERY PARAMETERS

**accountNumber**

String

required

Number of the customer's account that earns tax-reportable income

**status**

String

optional

Status of the document in the Cross River system:

- Importing
- Active
- Recalled

**documentType**

String

optional

Document type the query returns information about:

- Form1099Int
- Unknown

**year**

Integer

optional

Year the document covers, in the format YYYY

 Curl Node.js Python Ruby Go

```
curl --location 'https://sandbox.crbcos.com/core/v1/dda/accounts/220325085'
--header 'Authorization: Bearer <token>'
```

## Responses

● 200



```
{  "results":
  [

  ],
  "pageNumber": 1,
  "pageSize": 0,
  "hasPreviousPage": false,
  "hasNextPage": false
}
```

### IMPORTANT

Only pull documents as needed. If you want to pull a large number of documents, speak to your Relationship Manager.

The IRS requires Cross River to provide account holders with an IRS Form 1099 for earnings equal to or greater than \$10.00, per taxpayer identification number (TIN). Should the customer have more than one account, the system generates a 1099 for *each* account if the earnings of *all* accounts total \$10.00 or more.

### 4.1.3. Tax document

---

## Endpoint: /core/v1/dda/accounts/{accountNumber}/tax-documents/{id}

Gets a specific tax document associated with the account. This call returns a PDF of the specified tax document.

GET

<https://sandbox.crbcos.com/core/v1/dda/accounts/{accountNumber}/tax-documents/{id}> 

### Request

#### PATH PARAMS

**accountNumber** String **required**

Number of the customer's account that earns tax-reportable income

**id** String **required**

The document id returned in a GET tax documents query for the specific desired document.



Curl



Node.js



Python



Ruby



Go



```
package main

import (
    "fmt"
    "net/http"
    "io"
)

func main() {

    url := "https://sandbox.crbcos.com/core/v1/dda/accounts/2203250853/tax-d
    method := "GET"

    client := &http.Client {
    }
    req, err := http.NewRequest(method, url, nil)

    if err != nil {
        fmt.Println(err)
        return
    }
    req.Header.Add("Authorization", "Bearer <token>")

    res, err := client.Do(req)
    if err != nil {
        fmt.Println(err)
        return
    }
    defer res.Body.Close()

    body, err := io.ReadAll(res.Body)
    if err != nil {
        fmt.Println(err)
        return
    }
    fmt.Println(string(body))
}
```

## Responses

● 200



RETURNS A PDF

### Response

The system responds with a PDF of the specific tax document for display in your app.

#### IMPORTANT

Only pull documents as needed. If you want to pull a large number of documents, speak to your Relationship Manager.

The IRS requires Cross River to provide account holders with an IRS Form 1099 for earnings equal to or greater than \$10.00, per taxpayer identification number (TIN). Should the customer have more than one account, the system generates a 1099 *foreach* account if the earnings of *all* accounts total \$10.00 or more.

## 4.2. Subledgers APIs

### 4.2.1. Create subledger

---



## Endpoint: /core/v1/dda/subaccounts

Creates a subledger for the specified Master Account.

POST

https://sandbox.crbcos.com/core/v1/dda/subaccounts



Request

Response

### BODY PARAMETERS

**masterAccountNumber** String **required**

Master account to open the subledger under

**title** String **required**

The name on the account. Usually this is the account holder name, but not always. 255 characters maximum.

**customerId** String **optional**

The unique ID assigned to a customer when the customer record is created. You need this ID to take action for a specific customer. For example, when opening an account or adding an address to a customer record.

The ID is in GUID format.

**beneficiary ▶** Object **optional**

Beneficiary profile for this subledger

**clientIdIdentifier** String **optional**

Use this attribute to add your own unique identifying string to a payment call or COS record. This attribute is useful for idempotency purposes.



Curl



Node.js



Python



Ruby



Go



```
curl --location 'https://sandbox.crbcos.com/core/v1/dda/subaccounts' \
--header 'Idempotency-key: 3392de17-6e92-4999-b943-cad5f19b1629' \
--data '{
  "masterAccountNumber": "2819503463",
  "title": "Jerry Penn",
  "customerId": "53c34cf0-4cd2-4768-937b-b30000dffdca",
  "beneficiary": {
    "firstName": "Jerry" ,
    "lastName": "Penn",
    "streetAddress1": "12 Main St",
    "city": "New York",
    "state": "NY",
    "postalCode": "10001",
    "countryCode": "US"
  }
}
```

## Responses

● 200

● 400



```
{
  "accountNumber": "302022615264",
  "accountType": "Deposit",
  "status": "Active",
  "ledgerType": "Passthrough",
  "productType": "Checking",
  "title": "Jerry Penn",
  "currentBalance": 0,
  "availableBalance": 0,
  "holdAmount": 0,
  "transactionCount": 0,
  "creditLimit": 0,
  "currency": "usd",
  "openedAt": "2025-06-24T03:05:04.7952946-04:00",
  "masterAccountNumber": "2819503463",
  "customerId": "53c34cf0-4cd2-4768-937b-b30000dffdca",
  "beneficiary": {
    "firstName": "Jerry",
    "lastName": "Penn",
    "streetAddress1": "12 Main St",
    "city": "New York",
    "state": "NY",
    "postalCode": "10001",
    "countryCode": "US"
  },
  "partnerId": "cd9c12f4-7691-424a-b38b-af5b0134c611",
  "lastMaintenanceAt": "2025-06-24T03:05:04.7952946-04:00",
  "productId": "83bed086-8182-4151-a1e3-af5b01362783",
  "lastModifiedAt": "2025-06-24T03:05:04.8043224-04:00"
}
```

## 4.2.2. Subledger info

---

## Endpoint: /core/v1/dda/subaccounts/{accountNumber}

Returns information on a specific subledger account

GET

<https://sandbox.crbcos.com/core/v1/dda/subaccounts/{accountNumber}> 

Request

Response

### PATH PARAMS

**accountNumber**

String

**required**

Subaccount number to retrieve information about



Curl



Node.js



Python



Ruby



Go



```
curl --location 'https://sandbox.crbcos.com/core/v1/dda/subaccounts/302022'
--header 'Authorization: Bearer <token>'
```

## Responses

● 200



```
{
  "accountNumber": "302022615264",
  "accountType": "Deposit",
  "status": "Active",
  "ledgerType": "Passthrough",
  "productType": "Checking",
  "title": "Jerry Penn",
  "currentBalance": 0,
  "availableBalance": 0,
  "holdAmount": 0,
  "transactionCount": 0,
  "creditLimit": 0,
  "currency": "usd",
  "openedAt": "2025-06-24T03:05:04.797-04:00",
  "masterAccountNumber": "2819503463",
  "customerId": "53c34cf0-4cd2-4768-937b-b30000dffdca",
  "beneficiary": {
    "firstName": "Jerry",
    "lastName": "Penn",
    "streetAddress1": "12 Main St",
    "city": "New York",
    "state": "NY",
    "postalCode": "10001",
    "countryCode": "US"
  },
  "partnerId": "cd9c12f4-7691-424a-b38b-af5b0134c611",
  "lastMaintenanceAt": "2025-06-24T03:05:04.797-04:00",
  "productId": "83bed086-8182-4151-a1e3-af5b01362783",
  "lastModifiedAt": "2025-06-24T03:05:04.8043224-04:00"
}
```

## 4.3. Time deposit (CD) APIs

### 4.3.1. CD activity

---

## Endpoint: /core/v1/dda/accounts/{accountNumber}/time-deposit

Returns information about the account activity of a specific time deposit account.

GET

<https://sandbox.crbcos.com/core/v1/dda/accounts/{accountNumber}/time-deposit> 

Request

Response

### PATH PARAMS

**accountNumber**

String

**required**

Account number of the time deposit account





Curl



Node.js



Python



Ruby



Go



```
curl --location 'https://sandbox.crbcos.com/core/v1/dda/accounts/190432130'
```

Responses

● 200





## 4.3.2. CD penalty fee

### Endpoint: /core/v1/dda/accounts/{accountNumber}/time-deposit/pena

Returns the penalty fee if you make an early withdrawal to an active time deposit account. The response will define the number of months during which a penalty will usually be applied on early withdrawal (months), and the number of days of interest to be paid in penalty (feeDays).

GET

<https://sandbox.crbcos.com/core/v1/dda/accounts/{accountNuml> 

Request

Response

#### PATH PARAMS

**accountNumber** String **required**

Account number of the time deposit account



Curl



Node.js



Python



Ruby



Go



```
curl --location 'https://sandbox.crbcos.com/core/v1/dda/accounts/190432130'
```

#### Responses

● 200



```
{
  "months": 0,
  "feeDays": 0,
  "feeAmount": 0
}
```

### 4.3.3. Early funds withdrawal

---

## Endpoint: /core/v1/dda/accounts/{accountNumber}/time-deposit/early

Requests to withdraw the funds from a time deposit (CD) account before the account reaches maturity. Depending on the setting of the waivePenaltyFee attribute, the penalty fee is either waived or not. If the penalty fee for early withdrawal is waived, it is as if the account owner performs a partial withdrawal of funds.

Actual withdrawal of the funds requires a transaction call.

**POST**

<https://sandbox.crbcos.com/core/v1/dda/accounts/{accountNumber}/time-deposit/early> 

**Request**

Response

### PATH PARAMS

**accountNumber** String **required**

Account number of the time deposit account

### BODY PARAMETERS

**waivePenaltyFee** Boolean **required**

If true, a request for early withdrawal will be processed without a penalty fee.  
Otherwise, false.



Curl



Node.js



Python



Ruby



Go



```
curl --location 'https://sandbox.crbcos.com/core/v1/dda/accounts/190432130' \
--header 'Content-Type: application/json' \
--header 'Authorization: Bearer <token>' \
--data '{
  "waivePenaltyFee": true
}'
```

Responses

● 200







## 4.4. Deposit sweeps APIs

### 4.4.1. Account details

---

## Endpoint: /SweepsIntrafi/v1/accounts/{accountNumber}

Returns specific details of an account enabled for sweeps

GET

<https://sandbox.crbcos.com/SweepsIntrafi/v1/accounts/{accountNumber}> 

Request

Response

### PATH PARAMS

**accountNumber**

String

**required**

Account number of account you want to look up



Curl



Node.js



Python



Ruby



Go



```
curl --location --globoff 'SweepsIntrafi/v1/accounts/2666129453}' \  
--header 'Accept: application/json' \  
--header 'Content-Type: application/json'
```

Responses

200 404



```
{  
  "accountNumber": "2666129453",  
  "status": "Active"
```

## 4.4.2. Exclude bank/account

---

## Endpoint: SweepsIntrafi/v1/accounts/{accountNumber}

Use this call to:

Exclude banks from the sweep network

Your customers may already have funds at a certain bank and this is a way to exclude that bank.

Opt out of sweeps - When a product is enabled for sweeps, all accounts created from that product are enrolled into the sweeps program. Use this endpoint to mark an account inactive for sweeps, effectively opting out for sweeps.

**PUT**

<https://sandbox.crbcos.com/SweepsIntrafi/v1/accounts/{accountNumber}> 

**Request**

Response

### PATH PARAMS

**accountNumber** String **required**

Account number of account you want to make changes to

### BODY PARAMETERS

**status** String **required**

Set to "Active" to enroll this account in the sweeps program.

Set to "Inactive" to opt this account out of the sweeps program.

**bankExclusions** String **optional**

Bank IDs of banks to excluded from the network of banks where your funds may be swept to.





Curl



Node.js



Python



Ruby



Go



```
curl --location --request PUT 'https://sandbox.crbcos.com/SweepsIntrafi/v1
--header 'Content-Type: application/json'
--header 'Authorization: Bearer <token>'
--data '{
  "status": "Active",
  "bankExclusions": [
    "30895",
    "10015"
  ]
}'
```

## Responses

● 200● 404

```
{
  "accountNumber": "2666129453",
  "status": "Active",
  "title": "Sweeps Test 2",
  "bankExclusions": [
    "30895",
    "10015"
  ],
  "coreBalance": 125627321,
  "coreBalanceAsOf": "2024-09-25T16:17:20.1-04:00",
  "custodialBalance": 25627321,
  "networkBalance": 100000000,
  "networkBalanceAsOf": "2024-09-24T16:33:43.613-04:00",
  "maxSweepAmount": 0,
  "productId": "e4404005-03c1-4710-b759-b13700ebfac4",
  "partnerId": "6e6769ec-99e0-4c4f-8ae8-b13700eab0fc",
  "openedAt": "2024-05-14T16:26:10.227-04:00",
  "lastModifiedAt": "2024-09-25T16:18:48.5446146-04:00",
  "productName": "ibs Checking Product",
  "partnerName": "Ibrahim Saeed Partner"
}
```

## 4.4.3. Identify bank

### Endpoint: /SweepsIntrafi/v1/Banks/{bankNumber}

The previous endpoints have shared strings of Bank IDs when sharing exclusions. This endpoint returns the bank name.

GET

<https://sandbox.crbcos.com/SweepsIntrafi/v1/Banks/{bankNumber}>

Request

Response

#### PATH PARAMS

**bankNumber** String **required**

The bank number for the bank



Curl



Node.js



Python



Ruby



Go



```
curl --location 'https://sandbox.crbcos.com/SweepsIntrafi/v1/banks/10004'
```

#### Responses

200

400

404



```
{
  "bankNumber": "10004",
  "name": "Farmers State Bank",
  "abaNumber": "075903763",
  "effectiveDate": "2025-06-20T00:00:00-04:00",
  "lastModifiedAt": "2025-06-20T16:35:45.5739859-04:00"
}
```

#### 4.4.4. Bank list

---

## Endpoint: SweepsIntrafi/v1/Banks

Returns a list of all banks.

GET

<https://sandbox.crbcos.com/SweepsIntrafi/v1/Banks>



Request

Response



Curl



Node.js



Python



Ruby



Go



```
curl --location 'SweepsIntrafi/v1/Banks' \  
--header 'Accept: application/json' \  
--header 'Content-Type: application/json'
```

Responses

● 200





## 4.5. Webhook events

---

When you work with Cross River accounts, cards and payments, you register for specific **webhooks** which have a specific **event format**.

This table presents common `Accounts` and `Subledger` webhook events that are described in detail below.



Event Name	Description
<u>Core.Account.Opened</u>	A new account was opened
<u>Core.SubAccount.Opened</u>	Subledger (subaccount) was opened
<u>Core.SubAccount.Closed</u>	Subledger (subaccount) was closed
<u>Core.Restriction.Placed</u>	Restriction was placed on the account
<u>Core.Restriction Removed</u>	Restriction was removed for the account
<u>Core.Transaction.Completed (Incoming ACH)</u>	Transaction completed successfully for and incoming ACH
<u>Core.Transaction.Completed (Incoming Checks)</u>	Transaction completed successfully for an incoming check
<u>Core.Transaction.Completed (Interest Payments)</u>	Transaction completed successfully for interest payments
<u>Core.Transaction.Completed (Outbound ACH Aggregate)</u>	Transaction completed successfully for outbound ACH aggregate
<u>Core.Transaction.Completed (Outbound ACH Individual)</u>	Transaction completed successfully for outbound ACH individual
<u>Core.Transaction.Rejected</u>	Transaction was rejected on account
<u>Core.Activity.Posted (Inbound ACH)</u>	Activity was posted to the account for and inbound ACH
<u>Core.Activity.Posted (Internal Transfer)</u>	Activity was posted to the account for an internal transfer
<u>Core.Activity.Posted (Outbound ACH)</u>	Activity was posted to the account for an outbound ACH

## Event examples

### Core.Account.Opened



Sample Event



```
{
  "id": "259cdbca-6a89-4af8-a50e-ada3010fb13f",
  "eventName": "Core.Account.Opened",
  "status": "Pending",
  "partnerId": "e6c3824a-377f-44d5-a2f6-a9a600c9b37e",
  "createdAt": "2021-01-26T09:48:10.1011462-05:00",
  "resources": [
    "core/v1/dda/accounts/2235223803"
  ],
  "details": []
}
```

## Core.SubAccount.Opened



Sample Event



```
{
  "id": "7dffc503-e24b-4fdd-96c3-b2e20129aa6d",
  "eventName": "Core.SubAccount.Opened",
  "status": "Pending",
  "partnerId": "1e5d3f04-ae24-4af6-9e30-aecf012b99dd",
  "createdAt": "2025-05-19T14:03:46.053+00:00",
  "resources": [
    "core/v1/dda/subaccounts/327631054035"
  ],
  "details": [
    {
      "subAccountNumber": "327631054035",
      "masterAccountNumber": "2254575653",
      "accountType": "Deposit",
      "status": "Active",
      "title": "Robert Jordan's Sub",
      "clientIdentifier": null
    }
  ]
}
```

## Core.SubAccount.Closed



```
{
  "id": "d08ebe9e-c024-44bb-ba3b-b2e2012a371f",
  "eventName": "Core.SubAccount.Closed",
  "status": "Pending",
  "partnerId": "1e5d3f04-ae24-4af6-9e30-aecf012b99dd",
  "createdAt": "2025-05-19T14:05:46.117+00:00",
  "resources": [
    "core/v1/dda/subaccounts/327631054035"
  ],
  "details": [
    {
      "subAccountNumber": "327631054035",
      "masterAccountNumber": "2254575653",
      "accountType": "Deposit",
      "status": "Closed",
      "title": "Robert Jordan's Sub",
      "clientIdentifier": null
    }
  ]
}
```

## Core.Restriction.Placed



```
{
  "id": "9ef09f23-f14d-4634-b1b5-b2e201288d95",
  "eventName": "Core.Restriction.Placed",
  "status": "Pending",
  "partnerId": "1e5d3f04-ae24-4af6-9e30-aecf012b99dd",
  "createdAt": "2025-05-19T13:59:42.99+00:00",
  "resources": [
    "core/v1/dda/accounts/2161759366/restrictions/906ed893-bb6f-4a84-a3ef-b2e2012870b7",
  ],
  "details": [
    {
      "restrictionId": "906ed893-bb6f-4a84-a3ef-b2e2012870b7",
      "subAccountNumber": "2161759366",
      "masterAccountNumber": "2161759366",
      "accountType": "Deposit",
      "status": "Active",
      "amountThreshold": "",
      "currency": "usd",
      "rail": "",
      "transactionCode": null,
      "transactionType": ""
    }
  ]
}
```

## Core.Restriction.Removed



```
{
  "id": "2c693879-6ce4-48eb-9def-b2e20128d3f1",
  "eventName": "Core.Restriction.Removed",
  "status": "Pending",
  "partnerId": "1e5d3f04-ae24-4af6-9e30-aecf012b99dd",
  "createdAt": "2025-05-19T14:00:43.027+00:00",
  "resources": [
    "core/v1/dda/accounts/2161759366/restrictions/906ed893-bb6f-4a84-a3ef-b2e2012870b7",
  ],
  "details": [
    {
      "restrictionId": "906ed893-bb6f-4a84-a3ef-b2e2012870b7",
      "subAccountNumber": "2161759366",
      "masterAccountNumber": "2161759366",
      "accountType": "Deposit",
      "status": "Inactive",
      "amountThreshold": "",
      "currency": "usd",
      "rail": "",
      "transactionCode": null,
      "transactionType": ""
    }
  ]
}
```

## Core.Transaction.Completed (Incoming ACH)



```
{
  "id": "3804c582-b2ca-4c18-a3a8-b1ca00de6003",
  "eventName": "Core.Transaction.Completed",
  "status": "Pending",
  "partnerId": "1e5d3f04-ae24-4af6-9e30-aecf012b99dd",
  "createdAt": "2024-08-12T09:29:38.567-04:00",
  "resources": [
    "core/v1/transactions/35fe9374-5bd5-4aa6-8e68-b1ca00de0ecc"
  ],
  "details": [
    {
      "transactionId": "35fe9374-5bd5-4aa6-8e68-b1ca00de0ecc",
      "transactionCode": "Incoming ACH",
      "debitSubAccount": "00000271282",
      "debitMasterAccount": "00000271282",
      "debitResult": "OK",
      "creditSubAccount": "2151546989",
      "creditMasterAccount": "2151546989",
      "creditResult": "OK",
      "rail": "Ach",
      "railId": "payment/5836a14f-34a7-4a47-a9fe-b1ca00de0ebe",
      "amount": "75000",
      "currency": "usd",
      "clientIdentifier": null,
      "description": "IRS REFUND ACH PPD ID: 3333385474 REF: A2254L9KA06X",
      "proposedAt": "08/12/2024 09:29:29",
      "executedAt": "08/12/2024 09:29:29",
      "schedule": ""
    }
  ]
}
```

## Core.Transaction.Completed (Incoming Check)

```
{
  "id": "ae52b166-8651-4d44-b772-b23000e47d9a",
  "eventName": "Core.Transaction.Completed",
  "status": "Pending",
  "partnerId": "ba9d0234-26e9-43a2-9070-aedd0108eadb",
  "createdAt": "2024-11-22T08:51:54.533-05:00",
  "resources": [
    "core/v1/transactions/4de4f18a-d3bd-4327-832a-b23000e45ab2"
  ],
  "details": [
    {
      "transactionId": "4de4f18a-d3bd-4327-832a-b23000e45ab2",
      "transactionCode": "Check",
      "debitSubAccount": "199560644304",
      "debitMasterAccount": "199560644304",
      "debitResult": "OK",
      "creditSubAccount": "26250000095",
      "creditMasterAccount": "26250000095",
      "creditResult": "OK",
      "rail": "Checks",
      "railId": "payment/5cc1369e-bd27-4bd9-8f15-b23000e45aad",
      "amount": "10000",
      "currency": "usd",
      "clientIdentifier": null,
      "description": "CHECK #12345 REF: C327GJUOK7Q5",
      "proposedAt": "11/22/2024 08:51:29",
      "executedAt": "11/22/2024 08:51:30",
      "schedule": ""
    }
  ]
}
```

## Core.Transaction.Completed (Interest Payment)



```
{
  "id": "6d7da4d8-9ee6-46f9-a218-b22700f56b58",
  "eventName": "Core.Transaction.Completed",
  "status": "Pending",
  "partnerId": "ba9d0234-26e9-43a2-9070-aedd0108eadb",
  "createdAt": "2024-11-13T09:53:32.663-05:00",
  "resources": [
    "core/v1/transactions/a89e9bf2-c3fb-4e28-a20e-b22700ef4ed2"
  ],
  "details": [
    {
      "transactionId": "a89e9bf2-c3fb-4e28-a20e-b22700ef4ed2",
      "transactionCode": "Interest",
      "debitSubAccount": "40050060011",
      "debitMasterAccount": "40050060011",
      "debitResult": "OK",
      "creditSubAccount": "2151546989",
      "creditMasterAccount": "2151546989",
      "creditResult": "OK",
      "rail": "Internal",
      "railId": "3253942f-97b7-4e44-a60b-b22700ef4ed2",
      "amount": "337",
      "currency": "usd",
      "clientIdentifier": null,
      "description": "Interest Payment",
      "proposedAt": "11/13/2024 09:53:16",
      "executedAt": "11/13/2024 09:53:16",
      "schedule": ""
    }
  ]
}
```

## Core.Transaction.Completed (Outbound ACH Aggregate)







```
{
  "id": "ef5b6453-a4c1-48c5-a22e-b22700ebcd00",
  "eventName": "Core.Transaction.Completed",
  "status": "Pending",
  "partnerId": "ba9d0234-26e9-43a2-9070-aedd0108eadb",
  "createdAt": "2024-11-13T09:18:31.47-05:00",
  "resources": [
    "core/v1/transactions/4421921c-c5eb-4cbf-a0f8-b22700ebab83",
    "core/v1/transactions/fb6fd7db-fdba-4e3d-b3a5-b22700ebab87"
  ],
  "details": [
    {
      "transactionId": "4421921c-c5eb-4cbf-a0f8-b22700ebab83",
      "transactionCode": "Outgoing ACH",
      "debitSubAccount": "2905551814",
      "debitMasterAccount": "2905551814",
      "debitResult": "OK",
      "creditSubAccount": "00000016431",
      "creditMasterAccount": "00000016431",
      "creditResult": "OK",
      "rail": "Ach",
      "railId": "distribution/b8d9621e-ab11-4349-9e93-b22700eba7f4",
      "amount": "2500",
      "currency": "usd",
      "clientIdentifier": null,
      "description": "ACH DIS REF: ADIS31894FU PUSH ORIGINATION FUNDING 24111",
      "proposedAt": "11/13/2024 09:18:03",
      "executedAt": "11/13/2024 09:18:04",
      "schedule": ""
    },
    {
      "transactionId": "fb6fd7db-fdba-4e3d-b3a5-b22700ebab87",
      "transactionCode": "Outgoing ACH",
      "debitSubAccount": "00000016431",
      "debitMasterAccount": "00000016431",
      "debitResult": "OK",
      "creditSubAccount": "2905551814",
      "creditMasterAccount": "2905551814",
      "creditResult": "OK",
      "rail": "Ach",
      "railId": "distribution/b8d9621e-ab11-4349-9e93-b22700eba7f4",
      "amount": "4000",
      "currency": "usd",
      "clientIdentifier": null,
      "description": "ACH DIS REF: ADIS31894FU PULL ORIGINATION FUNDING 24111"
    }
  ]
}
```

```

    "proposedAt": "11/13/2024 09:18:19",
    "executedAt": "11/13/2024 09:18:19",
    "schedule": ""
  }
]
}

```

## Core.Transaction.Completed (Outbound ACH Individual)



Sample Event



```

{
  "id": "6d7da4d8-9ee6-46f9-a218-b22700f56b58",
  "eventName": "Core.Transaction.Completed",
  "status": "Pending",
  "partnerId": "ba9d0234-26e9-43a2-9070-aedd0108eadb",
  "createdAt": "2024-11-13T09:53:32.663-05:00",
  "resources": [
    "core/v1/transactions/a89e9bf2-c3fb-4e28-a20e-b22700ef4ed2"
  ],
  "details": [
    {
      "transactionId": "a89e9bf2-c3fb-4e28-a20e-b22700ef4ed2",
      "transactionCode": "Outgoing ACH",
      "debitSubAccount": "00000016431",
      "debitMasterAccount": "00000016431",
      "debitResult": "OK",
      "creditSubAccount": "2405952710",
      "creditMasterAccount": "2405952710",
      "creditResult": "OK",
      "rail": "Ach",
      "railId": "payment/3253942f-97b7-4e44-a60b-b22700ef4ed2",
      "amount": "17500",
      "currency": "usd",
      "clientIdentifier": null,
      "description": "PETER GRIFFIN DIRDEP ACH PPD ID: REF: A318W1P0MK7I",
      "proposedAt": "11/13/2024 09:53:16",
      "executedAt": "11/13/2024 09:53:16",
      "schedule": "0,0,17500"
    }
  ]
}

```

## Core.Transaction.Rejected



Sample Event



```
{
  "id": "638dc050-ab02-4d11-93f7-b1cd01308285",
  "eventName": "Core.Transaction.Rejected",
  "status": "Pending",
  "partnerId": "ba9d0234-26e9-43a2-9070-aedd0108eadb",
  "createdAt": "2024-08-15T14:28:41.19-04:00",
  "resources": [
    "core/v1/transactions/113d5b53-a1a8-41ca-bbf1-b1cd013076a1"
  ],
  "details": [
    {
      "transactionId": "113d5b53-a1a8-41ca-bbf1-b1cd013076a1",
      "transactionCode": "Account Transfer",
      "debitSubAccount": "199560644304",
      "debitMasterAccount": "199560644304",
      "debitResult": "NSF",
      "creditSubAccount": "20020020022",
      "creditMasterAccount": "20020020022",
      "creditResult": "OK",
      "rail": "Internal",
      "railId": "0",
      "amount": "1500000",
      "currency": "usd",
      "clientIdentifier": null,
      "description": "Account Transfer",
      "proposedAt": "08/15/2024 14:28:31",
      "executedAt": "08/15/2024 14:28:31",
      "schedule": ""
    }
  ]
}
```

## Core.Activity.Posted (Inbound ACH)



```
{
  "id": "62fa2f65-a7b7-4741-85ee-b2db01558daa",
  "eventName": "Core.Activity.Posted",
  "status": "Pending",
  "partnerId": "1e5d3f04-ae24-4af6-9e30-aecf012b99dd",
  "createdAt": "2025-05-12T16:43:33.457+00:00",
  "resources": [
    "core/v1/dda/accounts/2254575653/activity/d1fe8741-e885-4340-82e7-b2db01558daa",
  ],
  "details": [
    {
      "activityId": "d1fe8741-e885-4340-82e7-b2db01558490",
      "transactionId": "9d1f39e4-4c78-48ac-89ce-b2db01554583",
      "transactionType": "Credit",
      "transactionCode": "Incoming ACH",
      "subAccount": "2254575653",
      "masterAccount": "2254575653",
      "rail": "Ach",
      "railId": "payment/8e72c7df-c52c-48d4-aa5b-b2db01554571",
      "amount": "15000",
      "currency": "usd",
      "postingDate": "5/12/2025",
      "description": "SUBTESTING REFUND ACH PPD ID: 3333385474 REF: A13229562",
      "createdAt": "05/12/2025 16:43:25",
      "executedAt": "05/12/2025 16:43:25",
      "schedule": ""
    }
  ]
}
```

## Core.Activity.Posted (Internal Transfer)







```
{
  "id": "76e7b8a6-bbee-46f5-852a-b1ca00dc96bb",
  "eventName": "Core.Activity.Posted",
  "status": "Pending",
  "partnerId": "1e5d3f04-ae24-4af6-9e30-aecf012b99dd",
  "createdAt": "2024-08-12T09:23:08.35-04:00",
  "resources": [
    "core/v1/dda/accounts/2151546989/activity/d0c3c26d-3816-4c89-850b-b1ca00d",
    "core/v1/dda/accounts/2207975570/activity/811d94d0-d1b3-4feb-9bdd-b1ca00d",
  ],
  "details": [
    {
      "activityId": "d0c3c26d-3816-4c89-850b-b1ca00dc89f7",
      "transactionId": "27abd733-7d34-4823-90cc-b1ca00dc89f7",
      "transactionType": "Debit",
      "transactionCode": "Account Transfer",
      "subAccount": "2151546989",
      "masterAccount": "2151546989",
      "rail": "Internal",
      "railId": "0",
      "amount": "213",
      "currency": "usd",
      "postingDate": "8/12/2024",
      "description": "transfer",
      "createdAt": "08/12/2024 09:22:57",
      "executedAt": "08/12/2024 09:22:57",
      "schedule": ""
    },
    {
      "activityId": "811d94d0-d1b3-4feb-9bdd-b1ca00dc89f7",
      "transactionId": "27abd733-7d34-4823-90cc-b1ca00dc89f7",
      "transactionType": "Credit",
      "transactionCode": "Account Transfer",
      "subAccount": "2207975570",
      "masterAccount": "2207975570",
      "rail": "Internal",
      "railId": "0",
      "amount": "213",
      "currency": "usd",
      "postingDate": "8/12/2024",
      "description": "transfer",
      "createdAt": "08/12/2024 09:22:57",
      "executedAt": "08/12/2024 09:22:57",
      "schedule": ""
    }
  ]
}
```

```
]
}
```

## Core.Activity.Posted (Outbound ACH)



Sample Event



```
{
  "id": "c37fc243-9b8e-4d9c-9487-b19d010df95b",
  "eventName": "Core.Activity.Posted",
  "status": "Pending",
  "partnerId": "1e5d3f04-ae24-4af6-9e30-aecf012b99dd",
  "createdAt": "2024-06-28T12:22:56.733-04:00",
  "resources": [
    "core/v1/dda/accounts/2234629257/activity/348981eb-1af7-4ea6-86ec-b19d010",
  ],
  "details": [
    {
      "activityId": "348981eb-1af7-4ea6-86ec-b19d010ddab0",
      "transactionId": "33132892-398d-47f6-8d42-b19d010d8b9e",
      "transactionType": "Debit",
      "transactionCode": "Outgoing ACH",
      "subAccount": "2234629257",
      "masterAccount": "2234629257",
      "rail": "Ach",
      "railId": "payment/61a015bc-5e36-4728-9f11-b19d010d8b9e",
      "amount": "25000",
      "currency": "usd",
      "postingDate": "6/28/2024",
      "description": "PETER GRIFFIN STRING ACH PPD ID: REF: A180202KKD42",
      "createdAt": "06/28/2024 12:22:30",
      "executedAt": "06/28/2024 12:22:30",
      "schedule": ""
    }
  ]
}
```

## 4.6. Request and response codes

---

### Account status

Values returned in the `status` attribute in the response to many account calls describes whether the account is active or not, and if not, why not.

Status	Description
Inactive	Dormant for 1 year
Active	Account is active
Dormant	Dormant for 2 years
Escheat	Account is in escheatment (dormant for 3 years)
Closed	Account is closed
ChargeOff	Account has been overdrawn for a period of greater than 30 days

### Account types

Values returned in the `accountType` attribute in the response to `POST /v1/dda/accounts` describes the type of account created.

Type	Description
GeneralLedger	General Ledger
Deposit	Demand Deposit Account (DDA). Checking and Savings accounts.
CreditCard	Credit Card
StoredValue	Gifts Cards and other stored value accounts
Wallet	Wallet account
Unknown	

# Closure reason

Values returned in the `closureReason` attribute in the response to many account calls describes why an account was closed.

Reason returned
CustomerInitiatedNoComplaint
CustomerComplaintAccountClosing
CustomerComplaintAddOnProducts
CustomerComplaintApplicationProcessing
CustomerComplaintInterestRate
CustomerComplaintBillingDispute
CustomerComplaintBillingStatement
CustomerComplaintCollectionPractices
CustomerComplaintCreditDetermination
CustomerComplaintCreditReporting
CustomerComplaintCustomerService
CustomerComplaintFraud
CustomerComplaintFundAvailability
CustomerComplaintLateFees
CustomerComplaintMerchantDispute
CustomerComplaintOnlineAccountManagement
CustomerComplaintOtherFees
CustomerComplaintBalanceIssue
CustomerComplaintPromoDispute
CustomerComplaintMarketing
CustomerComplaintSettlementHardshipPrograms
BankPartnerInitiatedNoBalance
PartnerProductClosure
FirstPartyFraud
ThirdPartyFraud

**Reason returned**

FinancialCrimeNonFraud

HighOverdraftAmount

ExceededTransactionLimits

AccountInactivityDormancyEscheatment

TermsOfServiceBreach

BulkClosureExcerciseOther

ChargeOff

## 4.7. Error codes

---

Code	Description
1000	General exception
1200	DateFormed required
1201	DateFormed cannot be in the future
1202	BirthDate required
1203	BirthDate cannot be in the future
1204	EntityName required
1205	FirstName not allowed for classification
1206	MiddleName not allowed for classification
1207	LastName not allowed for classification
1208	Prefix not allowed for classification
1209	Suffix not allowed for classification
1210	PreferredName not allowed for classification
1211	FirstName required
1212	LastName required
1213	EntityName not allowed for classification
1214	Name required
1215	PoliticallyExposedPerson required
1216	PoliticallyExposedPerson not allowed for classification
1217	BirthDate not allowed for classification
1218	DateFormed not allowed for classification
1219	RegO required
1220	RegO not allowed for classification
1221	PrimaryOwnerCustomerId required
1222	PrimaryOwnerCustomerId not allowed for classification
1223	EntityType not allowed for classification



Code	Description
1224	OwnershipType not allowed for classification
1225	TaxIdType not allowed for classification
2000	General exception
2001	Account not active
2002	Status cannot be set to closed manually
2003	Account not open
2004	Account is closed
2005	Account is not closed
2006	Account cannot be closed until active holds removed
2007	Account cannot be closed until active memo posts removed
2008	Account cannot be closed until all completed transactions have been applied and balance is zero
2009	Account cannot be closed until sub accounts are closed
2010	Max total accounts exceeded for product
2011	Max active accounts exceeded for product
2012	Max daily accounts exceeded for product
2013	Account already exists
2014	Account number cannot be generated
2015	Account generation attempts exceeded
2016	Invalid account number prefix
2017	Invalid account number length
2018	Customer not assigned to same partner as product
2019	Customer classification not allowed for product
2020	Customer must have an address associated
2021	Customer must have a phone number associated

Code	Description
2022	Customer must be cleared for OFAC prior to account opening
2100	Sub account not found
2101	Sub account not active
2102	Cannot modify implicit sub account
2103	Sub account already exists
2104	Sub account is closed
2105	Sub account is not closed
2106	Master account required
2107	Master account not open
2108	Master account not active
2109	Master account not found
2110	Beneficiary required
2111	Sub account total limit has been reached for master account
2112	Sub accounts are not enabled for product
2113	Sub account number cannot be generated
2114	Sub account generation attempts exceeded
2115	Invalid sub account number prefix
2116	Invalid sub account number length
2117	Sub account cannot be closed until all completed transactions have been applied and balance is zero
2118	Sub account daily limit has been reached for master account
2119	Sub account active limit has been reached for master account
2200	Customer not found
2201	Customer not active
2202	Entity not active

Code	Description
2203	Address not active
2204	Cannot remove primary address
2205	DateFormed required
2206	DateFormed cannot be in the future
2207	BirthDate required
2208	BirthDate cannot be in the future
2209	Primary owner customer not found
2210	Primary owner customer not active
2211	Primary owner customer must be a person
2212	Entity and primary owner must be under same partner
2213	Entity and primary owner must be different customers
2214	Entity customer must be a business
2215	Owner customer not found
2216	Owner customer not active
2217	Owner customer must be a person
2218	Entity and owner customers must be under same partner
2219	Entity and owner must be different customers
2220	Owner relationship already exists for entity customer
2221	Unable to deactivate customer due to established account relationship(s)
2222	Partner not found or is inactive
2223	An active customer with this tax ID already exists for partner
2224	Cannot remove primary identification
2225	Cannot remove primary phone
2226	Invalid customer entity type
2350	Hold not found

Code	Description
2351	Hold is inactive
2352	Invalid hold amount
2353	Expiration date cannot be in the past
2375	Restriction not found
2376	Restriction not active
2377	Restriction policy in request must match restriction policy associated with change
2378	Restriction policy approval requires dual control
2379	Restriction policy not found
2380	Restriction policy not active
2381	Restriction policy max restrictions exceeded
2382	Restriction policy name already exists
2383	Pending restriction policy change with name already exists
2384	Restriction already inactive
2385	Restriction policy change not approved
2386	Restriction policy change not pending
2425	Interest accrual not found
2426	Start date must be before end date
2427	Amount must be greater than or equal to zero
2428	Payments not equal to zero must be posted to core
2450	Memo post not found
2451	Memo post not active
2452	TransactionCode is invalid
2475	Product not found
2476	Product not active

Code	Description
2477	Product change not approved
2478	Product change not pending
2479	Product in request must match product associated with change
2480	Classification not allowed for product type
2481	Product type invalid
2482	Product name already exists
2483	Unknown account type for product
2484	Pending product change with name already exists
2485	Interest Accrual not allowed for account type
2487	Statements not allowed for account type
2488	Product change approval requires dual control
2525	Relationship not found
2526	Relationship not active
2527	Primary relationship cannot be removed
2528	Relationship not allowed for account type
2529	RelationshipType not allowed for account classification
2530	IsTaxReportingOwner cannot be enabled for relationship type
2531	Customer must be classified as a person
2532	Customer relationship already exists
2533	Customer must be under same partner as account
2534	Customer must be cleared for OFAC prior to creating relationship
2550	Settlement not found
2551	Daily settlement must be pending
2552	Daily settlement must not be complete
2553	Daily settlement must be pre-settling and all interest must be processed

Code	Description
2554	All accounts must be settled before daily settlement can be completed
2575	Snapshot not found
2576	Snapshot account number does not match
2577	Snapshot partner does not match
2600	Statement not found
2601	Statement not completed
2602	Statement is closed
2603	AccountNumber must match statement
2604	Unable to generate statement for account type
2625	Transaction not found
2626	Debit account not found
2627	Debit account not active
2628	Credit account not found
2629	Credit account not active
2630	Debit and credit accounts must be different
2631	Debit account type not allowed
2632	Credit account type not allowed
2633	General ledger transfers require auto approval permission
2634	Transaction must be pending
2635	Transaction not effective
2636	Transaction must be under review
2637	Transaction authorization not found
2638	Transaction approval requires dual control
2639	Invalid transaction status
2640	PartnerId not accessible

Code	Description
2641	PartnerId is required for user type
2642	TransactionCode is invalid
2643	Max transaction flags exceeded
2644	Invalid transaction flag
2645	Transaction flags not allowed
2646	Transaction filter required
2647	Transaction trace number required
2648	Transaction proposal required
2649	Transaction amount must be greater than zero
2650	Transaction auth status invalid
2700	Accounting export is not processing

## 5. Cards

---

### Card Concepts

Cross River card issuing lets you offer customers different types of debit card products. These products, known as **instrument types**, include contact-less and virtual cards.

**Get started** with Cross River APIs.

Once you have access to our **sandbox** you can access these API endpoints in the `cardmanagement` module through **Swagger**.

### GET calls

- Add query parameters to filter the response the API returns to general GET calls.
- Use **Pagination** to control presentation of your results.

This table presents APIs that we describe in detail.



Action	API Call	Description
<a href="#"><u>Create new card</u></a>	POST /cardmanagement/v1/cards	Requests creation of a new card.
<a href="#"><u>Activate a card by ID</u></a>	POST /cardmanagement/v1/cards/{id}/activate	Activates a card for a customer. This is the preferred way to activate a card.
<a href="#"><u>Set initial PIN</u></a>	PUT /cardmanagement/v1/cards/{id}/pin	Sets the customer PIN.
<a href="#"><u>Replace card by ID</u></a>	POST /cardmanagement/v1/cards/{id}/replace	Order a replacement for damaged, lost or stolen cards.
<a href="#"><u>Card details</u></a>	GET /cardmanagement/v1/cards	Returns details about all cards based on the input.
<a href="#"><u>Card details by ID</u></a>	GET /cardmanagement/v1/cards/{id}	Returns the details of a specific debit card based on its ID.
<a href="#"><u>Transaction details</u></a>	GET /cardmanagement/v1/transactions	Returns information about card transactions based on partner ID.
<a href="#"><u>Get profile</u></a>	PUT /cardmanagement/v1/cards/{id}/profile	Retrieves the profile associated to a card.
<a href="#"><u>Suspend card</u></a>	POST /cardmanagement/v1/cards/{id}/suspend	Suspends a card based on card ID.
<a href="#"><u>Unsuspend card</u></a>	POST /cardmanagement/v1/cards/{id}/unsuspend	Unsuspects a card based on card ID.

Action	API Call	Description
<a href="#"><u>Admin block</u></a>	POST /cardmanagement/v1/cards/{id}/admin-block	Card blocked by admin. Status returned is suspended.
<a href="#"><u>Close card</u></a>	POST cardmanagement/v1/cards/{id}/close	Permanently deactivates a card. Closed cards cannot be reactivated.
<a href="#"><u>Simulate card auth</u></a>	POST /cardmanagement/v1/simulate/authorization	Simulates merchant authorization.
<a href="#"><u>Simulate card clearing</u></a>	POST /cardmanagement/v1/simulate/clearing	Simulates replacement of a memo post with a core transaction.
<a href="#"><u>Simulate incremental</u></a>	POST /cardmanagement/v1/simulate/incremental	simulates an incremental transaction reflected in the deposit account activity.
<a href="#"><u>Simulate single-message-clearing</u></a>	POST /cardmanagement/v1/simulate/single-message-clearing	Simulates verification and requires the cardholder to enter their PIN to authorize and clear their transaction.
<a href="#"><u>Simulate single-message-reversal</u></a>	POST /cardmanagement/v1/simulate/single-message-reversal	Simulates a reversal transaction to a single message clearing transaction.

## Related topics

- [Webhook events](#)
- [Request and response codes](#)
- [Error codes](#)

## Tutorials

- **Create a card**: Create and order a new card.
- **Activate a card**: Change the card status to Active.
- **Simulate card management**: Test the system with our card simulation endpoints.

## 5.1. APIs

### 5.1.1. Create new card

---

## Endpoint: /cardmanagement/v1/cards

To create a card, you must supply a valid deposit account number along with other card-specific information. The initial card status of a newly submitted card is unactivated. When the card is successfully created the order status is OrderPending.

**POST**

<https://sandbox.crbcos.com/cardmanagement/v1/cards>



**Request**

Response

### BODY PARAMETERS

**accountNumber** String **required**

Cardholder's deposit account number

**customerId** String **required**

The unique Cross River ID assigned to a customer when the customer record is created. You need this ID to take action for a specific customer. For example, when opening an account or adding an address to a customer record.

The ID is in GUID format.

**configurationId** String **required**

Unique ID for the card configuration profile the card is based on. You get this ID from Cross River. This ID is in GUID format.

**firstName** String **required**

Cardholder first name  
60 characters maximum

**middleName** String **optional**

Cardholder middle name  
60 characters maximum

**lastName** String **required**

Cardholder last name  
60 characters maximum

**suffix** String **optional**

Cardholder name suffix  
60 characters maximum

**phone ▶** Object **optional**

Details of the cardholder phone contact information

**emailAddress** String **optional**

Cardholder email address  
60 characters maximum

**shippingAddress ▶** Object **required**

Address to which a physical card is shipped

**billingAddress ▶** Object **required**

Address to send credit card bills

**nameOnCard** String **required**

Cardholder name as it appears on the card  
  
23 characters maximum

**companyNameOnCard**      String      optional

If the card is connected to a business account, the company name  
23 characters maximum

**shippingType**      String      required

How the card is shipped:

- Normal
- Expedited

**clientIdentifier**      String      optional

Use this attribute to add your own unique identifying string to a payment call or COS  
record. This attribute is useful for idempotency purposes.

50 characters maximum.

**cardDesignId**      String      optional

Identifier associated to the card design to use when overriding a program's default  
card design.



Curl



Node.js



Python



Ruby



Go



```
curl --location 'https://sandbox.crbcos.com/cardmanagement/v1/cards' \
--header 'Idempotency-key: b06ab5cc-553a-4fe4-ad78-751b5ff81f0e' \
--data-raw '{
  "accountNumber": "158560897007",
  "customerId": "e2599a17-d1e2-476c-9672-b2ff008fa575",
  "configurationId": "b2251a28-c218-4e2e-a787-b2f700ef0ecd",
  "firstName": "Daly",
  "lastName": "Khol",
  "phone": {
    "phoneType": "Home",
    "phoneNumber": "7185551234"
  },
  "emailAddress": "dkhol@gmail.com",
  "shippingAddress": {
    "street1": "1 Roshar Ave",
    "city": "Roshar",
    "state": "NY",
    "postalCode": "10001",
    "countryCode": "US"
  },
  "billingAddress": {
    "street1": "1 Roshar Ave",
    "city": "Roshar",
    "state": "NY",
    "postalCode": "10001",
    "countryCode": "US"
  },
  "nameOnCard": "Daly Khol",
  "shippingType": "Normal",
  "clientIdentifier": "b06ab5cc-553a-4fe4-ad78-751b5ff81f00"
}
```



Responses

● 200





## 5.1.2. Activate card by ID

---

## Endpoint: /cardmanagement/v1/cards/{id}/activate

Cards can only be used once they are activated. After creation, the card status is Unactivated. Once you are ready to activate the card for use, call the Activate Card ID endpoint. This is the preferred way to activate a card.

**POST**

<https://sandbox.crbcos.com/cardmanagement/v1/cards/{id}/activ> 

**Request**

Response

### PATH PARAMS

**id** String **required**

GUID format card ID in the Cross River system, which is the id attribute value of the response to Create a card. This is not the number that appears on the card.



Curl



Node.js



Python



Ruby



Go



```
curl --location --globoff --request POST 'https://sandbox.crbcos.com/cardm  
--header 'Idempotency-key: a5b1b129-421e-441b-9661-ec1c0396965e' \  
--data ''
```

Responses

● 200





### 5.1.3. Set initial PIN

---



## Endpoint: /cardmanagement/v1/cards/{id}/pin

Set an initial debit card personal identification number (PIN) number for withdrawing funds from an ATM and other uses. To set a card PIN number, the card cannot already have a PIN number set. This means that the card's isPinSet must have a value of false.

**PUT**

<https://sandbox.crbcos.com/cardmanagement/v1/cards/{id}/pin> 

**Request**

Response

### PATH PARAMS

**id** String **required**

GUID format card ID in the Cross River system, which is the id attribute value of the response to Create a card. This is not the number that appears on the card.

### BODY PARAMETERS

**pin** String **required**

4-digit personal identification number associated with the card for verification purposes during purchase or cash withdrawal transactions.



Curl



Node.js



Python



Ruby



Go



```
curl --location --globoff --request PUT 'https://sandbox.crbcos.com/cardma
--data '{
  "pin": "1534"
}
```

Responses

● 200





## 5.1.4. Replace card by ID

---

## Endpoint: /cardmanagement/v1/cards/{id}/replace

Order a replacement for damaged, lost or stolen cards using this endpoint. Damaged cards remain active until the cardholder activates the replacement card. Lost or stolen cards are immediately closed and a replacement card is issued.

**POST**

<https://sandbox.crbcos.com/cardmanagement/v1/cards/{id}/repla> 

**Request**

Response

### PATH PARAMS

**id** String **required**

GUID format card ID in the Cross River system, which is the id attribute value of the response to Create a card. This is not the number that appears on the card.

**request** ▶ Object **required**

Array containing details about the request to replace the card.



Curl



Node.js



Python



Ruby



Go



```
curl --location --globoff 'https://sandbox.crbcos.com/cardmanagement/v1/ca
--header 'id: 8c6a53e0-83ad-4b76-b446-b300006a3e6a' \
--data '{
  "replaceReason": "Damaged",
  "nameOnCard": "Daly Khol",
  "shippingType": "Normal",
  "clientIdentifier": "b06ab5cc-553a-4fe4-ad78-751b5ff81f00"
}'
```

Responses

● 200







## 5.1.5. Card details

---

## Endpoint: /cardmanagement/v1/cards

The request can contain any of the query parameters, must contain at least one. The response values below are repeated for each card. This example's input users Account Number.

GET

https://sandbox.crbcos.com/cardmanagement/v1/cards



Request

Response

### QUERY PARAMETERS

**from** String optional

Provide date and time

**to** String optional

Provide date and time

**firstName** String optional

First name of the cardholder

**lastName** String optional

Cardholder last name  
60 characters maximum

**productId** String optional

Product ID of the cardholder deposit account. This is in the GUID format.

**accountNumber** String optional

Deposit account number associated with the card.

**status** String optional

Select from:

- Unactivated
- Active
- Suspended
- Closed

**adminBlocked** Boolean optional

Select from:

- True
- False

**fraudSuspect** Boolean optional

Select from:

- True
- False

**statusReasonCode** String optional

Values:

- NotSet
- Damaged
- Expired
- Lost
- FraudCompromised
- Return
- NewEnrollment
- Stolen
- AdminBlocked
- Fraud Blocked

**orderStatus**                      String              optional

Values:

- OrderPending
- Completed
- Failed
- Authorizing
- AuthorizationFailed
- AuthorizationCompleted

**pageNumber**                      Integer              optional

Page number in a paginated API response.

**pageSize**                      Integer              optional

Number of items per page in a paginated response.



Curl



Node.js



Python



Ruby



Go



```
curl --location 'https://sandbox.crbcos.com/cardmanagement/v1/transactions' --data ''
```

Responses

● 200



```
{
  "results": [
    {
      "id": "ec2639b3-7524-4c1e-8b34-b30000c93225",
      "productId": "57146944-b145-4326-884d-b2f700ecf688",
      "partnerId": "19222b81-0e1e-452d-a842-b2f1011c16f3",
      "accountNumber": "158560897007",
      "createdAt": "2025-06-18T08:12:31.91-04:00",
      "description": "644371423186 StreamFlix           Anywhere",
      "isReversal": false,
      "traceNumber": 357609,
      "transmissionDate": "0618121231",
      "panLastFour": "4558",
      "isDebit": true,
      "isPin": false,
      "transactionAmount": 1999,
      "billingAmount": 1999,
      "settlementAmount": 0,
      "panEntryMode": "PANAutoEntryViaContactless",
      "pinCaptureCapability": "Unspecified",
      "retrievalReferenceNumber": "644371423186",
      "networkType": "Visa",
      "acquirerNetworkId": "7476",
      "merchantNameLocation": "StreamFlix           Anywhere",
      "merchantType": "1111",
      "merchantPostalCode": "10001",
      "transactionCurrencyCode": "840",
      "billingCurrencyCode": "840",
      "settlementCurrencyCode": "840",
      "authorizationCode": "750884",
      "transactionIdentifier": "1-10001719746199",
      "transactionUniqueIdentifier": "750884",
      "accountIdentification": "827277613081398",
      "partialAuthorizationIndicator": false,
      "isChargeBack": false,
      "outboundResponseCode": "00",
      "messageType": "0100",
      "processingCode": "BillPayment",
      "expirationDate": "2025-06-21T08:12:31.91-04:00",
      "billingConversionRate": 0,
      "settlementConversionRate": 0,
      "bankNetReferenceNumber": "747771579",
      "transactionFeeAmount": 0,
      "acquiringInstitutionCode": "52377112883",
      "forwardingInstitutionCode": "50185636050",
    }
  ]
}
```



```

    "processor": "i2c",
    "memoPostId": "10dbd093-a37a-4121-885a-b30000c9327b",
    "transactionKey": "283D2B2AC4BE5E3F1F34A939E3E96D9478DD2BBE",
    "transactionCurrency": "usd",
    "billingCurrency": "usd",
    "settlementCurrency": "usd",
    "postClosureAccountUsed": false,
    "relatedTransactionId": "ec2639b3-7524-4c1e-8b34-b30000c93225"
  },
  {
    "id": "71e83301-766d-46d0-b7a0-b30000ca0bc6",
    "productId": "57146944-b145-4326-884d-b2f700ecf688",
    "partnerId": "19222b81-0e1e-452d-a842-b2f1011c16f3",
    "accountNumber": "158560897007",
    "createdAt": "2025-06-18T08:15:37.6166667-04:00",
    "description": "644371423186 StreamFlix                               Anywhere",
    "isReversal": false,
    "traceNumber": 357609,
    "transmissionDate": "0618121537",
    "panLastFour": "4558",
    "isDebit": true,
    "isPin": false,
    "transactionAmount": 1999,
    "billingAmount": 1999,
    "settlementAmount": 0,
    "panEntryMode": "PANAutoEntryViaContactless",
    "pinCaptureCapability": "Unspecified",
    "retrievalReferenceNumber": "644371423186",
    "networkType": "Visa",
    "acquirerNetworkId": "9193",
    "merchantNameLocation": "StreamFlix                               Anywhere",
    "merchantType": "1111",
    "merchantPostalCode": "10001",
    "transactionCurrencyCode": "840",
    "billingCurrencyCode": "840",
    "settlementCurrencyCode": "840",
    "authorizationCode": "920441",
    "transactionIdentifier": "1-10001719746199",
    "transactionUniqueIdentifier": "920441",
    "accountIdentification": "827277613081398",
    "partialAuthorizationIndicator": false,
    "isChargeBack": false,
    "outboundResponseCode": "00",
    "messageType": "0200",
    "processingCode": "BillPayment",
    "expirationDate": "2025-06-21T08:15:37.6166667-04:00",
  }

```

```

    "billingConversionRate": 0,
    "settlementConversionRate": 0,
    "bankNetReferenceNumber": "747771579",
    "transactionFeeAmount": 0,
    "acquiringInstitutionCode": "58143509848",
    "forwardingInstitutionCode": "23817195170",
    "processor": "i2c",
    "memoPostId": "10dbd093-a37a-4121-885a-b30000c9327b",
    "coreTransactionId": "71e83301-766d-46d0-b7a0-b30000ca0bc6",
    "transactionKey": "7125F4B9F48F549B247346C855A40A10788B0B2C",
    "transactionCurrency": "usd",
    "billingCurrency": "usd",
    "settlementCurrency": "usd",
    "postClosureAccountUsed": false,
    "relatedTransactionId": "ec2639b3-7524-4c1e-8b34-b30000c93225"
  },
  {
    "id": "b031aafb-0e5a-490e-8fa6-b30000cc8b20",
    "productId": "57146944-b145-4326-884d-b2f700ecf688",
    "partnerId": "19222b81-0e1e-452d-a842-b2f1011c16f3",
    "accountNumber": "158560897007",
    "createdAt": "2025-06-18T08:24:43.1966667-04:00",
    "description": "654492282074 StreamFlix                               Anywhere",
    "isReversal": false,
    "traceNumber": 385910,
    "transmissionDate": "0618122443",
    "panLastFour": "4558",
    "isDebit": true,
    "isPin": false,
    "transactionAmount": 1999,
    "billingAmount": 1999,
    "settlementAmount": 0,
    "panEntryMode": "PANAUTOENTRYVIACONTACTLESS",
    "pinCaptureCapability": "UNSPECIFIED",
    "retrievalReferenceNumber": "654492282074",
    "networkType": "VISA",
    "acquirerNetworkId": "4832",
    "merchantNameLocation": "StreamFlix                               Anywhere",
    "merchantType": "1111",
    "merchantPostalCode": "10001",
    "transactionCurrencyCode": "840",
    "billingCurrencyCode": "840",
    "settlementCurrencyCode": "840",
    "authorizationCode": "489716",
    "transactionIdentifier": "1-10001425931742",
    "transactionUniqueIdentifier": "489716",
  }

```

```

"accountIdentification": "827277613081398",
"partialAuthorizationIndicator": false,
"isChargeBack": false,
"outboundResponseCode": "00",
"messageType": "0100",
"processingCode": "BillPayment",
"expirationDate": "2025-06-21T08:24:43.1966667-04:00",
"billingConversionRate": 0,
"settlementConversionRate": 0,
"bankNetReferenceNumber": "483338568",
"transactionFeeAmount": 0,
"acquiringInstitutionCode": "24341234788",
"forwardingInstitutionCode": "21312234800",
"processor": "i2c",
"memoPostId": "10a2ff3c-ba4c-4947-9876-b30000cc8b31",
"transactionKey": "993D5770CBFB6A66E104AA8EE7E9EF1FE01D2E8D",
"transactionCurrency": "usd",
"billingCurrency": "usd",
"settlementCurrency": "usd",
"postClosureAccountUsed": false,
"relatedTransactionId": "b031aafb-0e5a-490e-8fa6-b30000cc8b20"
},
{
  "id": "d14c88a9-0ade-498c-9329-b30000cca8bf",
  "productId": "57146944-b145-4326-884d-b2f700ecf688",
  "partnerId": "19222b81-0e1e-452d-a842-b2f1011c16f3",
  "accountNumber": "158560897007",
  "createdAt": "2025-06-18T08:25:08.4733333-04:00",
  "description": "(Reversal) 654492282074 StreamFlix
  "isReversal": true,
  "traceNumber": 385910,
  "transmissionDate": "0618122508",
  "panLastFour": "4558",
  "isDebit": false,
  "isPin": false,
  "transactionAmount": 123,
  "billingAmount": 123,
  "settlementAmount": 0,
  "panEntryMode": "PANAUTOENTRYVIACONTACTLESS",
  "pinCaptureCapability": "UNSPECIFIED",
  "retrievalReferenceNumber": "654492282074",
  "networkType": "Visa",
  "acquirerNetworkId": "8384",
  "merchantNameLocation": "StreamFlix
  "merchantType": "1111",
  "merchantPostalCode": "10001",

```

Anywhere

```

    "transactionCurrencyCode": "840",
    "billingCurrencyCode": "840",
    "settlementCurrencyCode": "840",
    "authorizationCode": "840479",
    "transactionIdentifier": "1-10001425931742",
    "transactionUniqueIdentifier": "840479",
    "accountIdentification": "827277613081398",
    "partialAuthorizationIndicator": false,
    "isChargeBack": false,
    "outboundResponseCode": "00",
    "messageType": "0420",
    "processingCode": "BillPayment",
    "expirationDate": "2025-06-21T08:25:08.4733333-04:00",
    "billingConversionRate": 0,
    "settlementConversionRate": 0,
    "bankNetReferenceNumber": "483338568",
    "transactionFeeAmount": 0,
    "acquiringInstitutionCode": "29167599701",
    "forwardingInstitutionCode": "56144299975",
    "processor": "i2c",
    "memoPostId": "10a2ff3c-ba4c-4947-9876-b30000cc8b31",
    "transactionKey": "991705EC38A5523A13B2B63A888B7E9A19554DE8",
    "transactionCurrency": "usd",
    "billingCurrency": "usd",
    "settlementCurrency": "usd",
    "reversalType": "Authorization",
    "postClosureAccountUsed": false,
    "relatedTransactionId": "b031aafb-0e5a-490e-8fa6-b30000cc8b20"
  },
  {
    "id": "e1cdd82c-72ae-4056-b8c2-b30000cdb9e9a",
    "productId": "57146944-b145-4326-884d-b2f700ecf688",
    "partnerId": "19222b81-0e1e-452d-a842-b2f1011c16f3",
    "accountNumber": "158560897007",
    "createdAt": "2025-06-18T08:29:05.5766667-04:00",
    "description": "654492282074 StreamFlix",
    "isReversal": false,
    "traceNumber": 385910,
    "transmissionDate": "0618122905",
    "panLastFour": "4558",
    "isDebit": true,
    "isPin": false,
    "transactionAmount": 123,
    "billingAmount": 123,
    "settlementAmount": 0,
    "panEntryMode": "PANAUTOENTRYVIACONTACTLESS",
    "Anywhere": true
  }
]

```

```

    "pinCaptureCapability": "Unspecified",
    "retrievalReferenceNumber": "654492282074",
    "networkType": "Visa",
    "acquirerNetworkId": "2138",
    "merchantNameLocation": "StreamFlix                Anywhere",
    "merchantPostalCode": "10001",
    "transactionCurrencyCode": "840",
    "billingCurrencyCode": "840",
    "settlementCurrencyCode": "840",
    "authorizationCode": "223597",
    "transactionIdentifier": "1-10001425931742",
    "transactionUniqueIdentifier": "223597",
    "accountIdentification": "827277613081398",
    "partialAuthorizationIndicator": true,
    "isChargeBack": false,
    "outboundResponseCode": "00",
    "messageType": "0100",
    "processingCode": "BillPayment",
    "expirationDate": "2025-06-21T08:29:05.5766667-04:00",
    "billingConversionRate": 0,
    "settlementConversionRate": 0,
    "bankNetReferenceNumber": "483338568",
    "transactionFeeAmount": 0,
    "acquiringInstitutionCode": "86912943724",
    "forwardingInstitutionCode": "28290432101",
    "processor": "i2c",
    "memoPostId": "10a2ff3c-ba4c-4947-9876-b30000cc8b31",
    "transactionKey": "4B698DA7B5B5B1E253C29FFAA09B481E5A5DF197",
    "transactionCurrency": "usd",
    "billingCurrency": "usd",
    "settlementCurrency": "usd",
    "postClosureAccountUsed": false,
    "relatedTransactionId": "b031aafb-0e5a-490e-8fa6-b30000cc8b20"
  },
  {
    "id": "170a6467-c547-4f71-bd94-b30000d52f55",
    "productId": "57146944-b145-4326-884d-b2f700ecf688",
    "partnerId": "19222b81-0e1e-452d-a842-b2f1011c16f3",
    "accountNumber": "158560897007",
    "createdAt": "2025-06-18T08:56:10.9466667-04:00",
    "description": "674583031323 StreamFlix                Anywhere",
    "isReversal": false,
    "traceNumber": 562452,
    "transmissionDate": "0618125610",
    "panLastFour": "4558",
    "isDebit": true,

```

```

    "isPin": false,
    "transactionAmount": 22,
    "billingAmount": 22,
    "settlementAmount": 0,
    "panEntryMode": "PANAutoEntryViaContactless",
    "pinCaptureCapability": "Unspecified",
    "retrievalReferenceNumber": "674583031323",
    "networkType": "Visa",
    "acquirerNetworkId": "1498",
    "merchantNameLocation": "StreamFlix                Anywhere",
    "merchantType": "1111",
    "merchantPostalCode": "10001",
    "transactionCurrencyCode": "840",
    "billingCurrencyCode": "840",
    "settlementCurrencyCode": "840",
    "authorizationCode": "160375",
    "transactionIdentifier": "1-10001055423125",
    "transactionUniqueIdentifier": "160375",
    "accountIdentification": "827277613081398",
    "partialAuthorizationIndicator": false,
    "isChargeBack": false,
    "outboundResponseCode": "00",
    "messageType": "0200",
    "processingCode": "Purchase",
    "expirationDate": "2025-06-21T08:56:10.9466667-04:00",
    "billingConversionRate": 0,
    "settlementConversionRate": 0,
    "bankNetReferenceNumber": "564514688",
    "transactionFeeAmount": 0,
    "acquiringInstitutionCode": "77848948385",
    "forwardingInstitutionCode": "19835472422",
    "processor": "i2c",
    "memoPostId": "a3a9a3b4-16ce-4eb0-875d-b30000d52f69",
    "coreTransactionId": "170a6467-c547-4f71-bd94-b30000d52f55",
    "transactionKey": "B40847BB1F5179AD605895146328FBBF205CDADDD",
    "transactionCurrency": "usd",
    "billingCurrency": "usd",
    "settlementCurrency": "usd",
    "postClosureAccountUsed": false,
    "relatedTransactionId": "170a6467-c547-4f71-bd94-b30000d52f55"
  },
  {
    "id": "baecdf9c-dcfa-402f-9062-b30000d710b8",
    "productId": "57146944-b145-4326-884d-b2f700ecf688",
    "partnerId": "19222b81-0e1e-452d-a842-b2f1011c16f3",
    "accountNumber": "158560897007",

```

"createdAt": "2025-06-18T09:03:01.73-04:00".

## 5.1.6. Card details by ID

---



## Endpoint: /cardmanagement/v1/cards/{id}

Returns the details of a specific debit card based on its ID.

GET

https://sandbox.crbcos.com/cardmanagement/v1/cards/{id}



Request

Response

### PATH PARAMS

**id**

String

**required**

GUID format card ID in the Cross River system, which is the id attribute value of the response to Create a card. This is not the number that appears on the card.



Curl



Node.js



Python



Ruby



Go



```
curl --location --globoff 'https://sandbox.crbcos.com/cardmanagement/v1/ca  
--data ''
```

Responses

● 200





## 5.1.7. Transaction details

---

## Endpoint: /cardmanagement/v1/transactions

Returns details about card transactions. The response values below are repeated for each transaction. This example uses Partner ID as the input.

GET

<https://sandbox.crbcos.com/cardmanagement/v1/transactions>



Request

Response

### QUERY PARAMETERS

**from** String optional

Provide date and time

**to** String optional

Provide date and time

**partnerId** String optional

Your ID in the Cross River system. This ID is in GUID format.

**productId** String optional

Product ID of the cardholder deposit account. This is in the GUID format.

**accountNumber** String optional

Deposit account number associated with the card

**minAmount** String optional

Minimum amount of the transaction

**maxAmount** String optional

Maximum amount of the transaction

**retrievalReferenceNumber** String optional

Card network generated identifier used to uniquely track a transaction across systems.

**outboundResponseCode** String optional

Code describing outbound response.

**isDebit** Boolean optional

Indicator whether a debit card was used in the transaction.

Values:

- True (if the card is a debit card)
- False

**isPin** Boolean optional

Indicator whether a PIN was used in the transaction.

Values:

- True
- False

**postClosureAccountUsed** Boolean optional

Indicates if a transaction was attempted on a closed account DDA and therefore had to be posted on the program's post closure account.

Values:

- True
- False

**pageNumber** Integer optional

Page number in a paginated API response.

**pageSize**

Integer

optional

Number of items per page in a paginated response.





Curl



Node.js



Python



Ruby



Go



```
curl --location 'https://sandbox.crbcos.com/cardmanagement/v1/transactions' --data ''
```

Responses

● 200



```
{
  "results": [
    {
      "id": "ec2639b3-7524-4c1e-8b34-b30000c93225",
      "productId": "57146944-b145-4326-884d-b2f700ecf688",
      "partnerId": "19222b81-0e1e-452d-a842-b2f1011c16f3",
      "accountNumber": "158560897007",
      "createdAt": "2025-06-18T08:12:31.91-04:00",
      "description": "644371423186 StreamFlix           Anywhere",
      "isReversal": false,
      "traceNumber": 357609,
      "transmissionDate": "0618121231",
      "panLastFour": "4558",
      "isDebit": true,
      "isPin": false,
      "transactionAmount": 1999,
      "billingAmount": 1999,
      "settlementAmount": 0,
      "panEntryMode": "PANAutoEntryViaContactless",
      "pinCaptureCapability": "Unspecified",
      "retrievalReferenceNumber": "644371423186",
      "networkType": "Visa",
      "acquirerNetworkId": "7476",
      "merchantNameLocation": "StreamFlix           Anywhere",
      "merchantType": "1111",
      "merchantPostalCode": "10001",
      "transactionCurrencyCode": "840",
      "billingCurrencyCode": "840",
      "settlementCurrencyCode": "840",
      "authorizationCode": "750884",
      "transactionIdentifier": "1-10001719746199",
      "transactionUniqueIdentifier": "750884",
      "accountIdentification": "827277613081398",
      "partialAuthorizationIndicator": false,
      "isChargeBack": false,
      "outboundResponseCode": "00",
      "messageType": "0100",
      "processingCode": "BillPayment",
      "expirationDate": "2025-06-21T08:12:31.91-04:00",
      "billingConversionRate": 0,
      "settlementConversionRate": 0,
      "bankNetReferenceNumber": "747771579",
      "transactionFeeAmount": 0,
      "acquiringInstitutionCode": "52377112883",
      "forwardingInstitutionCode": "50185636050",
    }
  ]
}
```

```
    "processor": "i2c",
    "memoPostId": "10dbd093-a37a-4121-885a-b30000c9327b",
    "transactionKey": "283D2B2AC4BE5E3F1F34A939E3E96D9478DD2BBE",
    "transactionCurrency": "usd",
    "billingCurrency": "usd",
    "settlementCurrency": "usd",
    "postClosureAccountUsed": false,
    "relatedTransactionId": "ec2639b3-7524-4c1e-8b34-b30000c93225"
  },
  {
    "id": "71e83301-766d-46d0-b7a0-b30000ca0bc6",
    "productId": "57146944-b145-4326-884d-b2f700ecf688",
    "partnerId": "19222b81-0e1e-452d-a842-b2f1011c16f3",
    "accountNumber": "158560897007",
    "createdAt": "2025-06-18T08:15:37.6166667-04:00",
    "description": "644371423186 StreamFlix                               Anywhere",
    "isReversal": false,
    "traceNumber": 357609,
    "transmissionDate": "0618121537",
    "panLastFour": "4558",
    "isDebit": true,
    "isPin": false,
    "transactionAmount": 1999,
    "billingAmount": 1999,
    "settlementAmount": 0,
    "panEntryMode": "PANAutoEntryViaContactless",
    "pinCaptureCapability": "Unspecified",
    "retrievalReferenceNumber": "644371423186",
    "networkType": "Visa",
    "acquirerNetworkId": "9193",
    "merchantNameLocation": "StreamFlix                               Anywhere",
    "merchantType": "1111",
    "merchantPostalCode": "10001",
    "transactionCurrencyCode": "840",
    "billingCurrencyCode": "840",
    "settlementCurrencyCode": "840",
    "authorizationCode": "920441",
    "transactionIdentifier": "1-10001719746199",
    "transactionUniqueIdentifier": "920441",
    "accountIdentification": "827277613081398",
    "partialAuthorizationIndicator": false,
    "isChargeBack": false,
    "outboundResponseCode": "00",
    "messageType": "0200",
    "processingCode": "BillPayment",
    "expirationDate": "2025-06-21T08:15:37.6166667-04:00",
```

```

    "billingConversionRate": 0,
    "settlementConversionRate": 0,
    "bankNetReferenceNumber": "747771579",
    "transactionFeeAmount": 0,
    "acquiringInstitutionCode": "58143509848",
    "forwardingInstitutionCode": "23817195170",
    "processor": "i2c",
    "memoPostId": "10dbd093-a37a-4121-885a-b30000c9327b",
    "coreTransactionId": "71e83301-766d-46d0-b7a0-b30000ca0bc6",
    "transactionKey": "7125F4B9F48F549B247346C855A40A10788B0B2C",
    "transactionCurrency": "usd",
    "billingCurrency": "usd",
    "settlementCurrency": "usd",
    "postClosureAccountUsed": false,
    "relatedTransactionId": "ec2639b3-7524-4c1e-8b34-b30000c93225"
  },
  {
    "id": "b031aafb-0e5a-490e-8fa6-b30000cc8b20",
    "productId": "57146944-b145-4326-884d-b2f700ecf688",
    "partnerId": "19222b81-0e1e-452d-a842-b2f1011c16f3",
    "accountNumber": "158560897007",
    "createdAt": "2025-06-18T08:24:43.1966667-04:00",
    "description": "654492282074 StreamFlix                               Anywhere",
    "isReversal": false,
    "traceNumber": 385910,
    "transmissionDate": "0618122443",
    "panLastFour": "4558",
    "isDebit": true,
    "isPin": false,
    "transactionAmount": 1999,
    "billingAmount": 1999,
    "settlementAmount": 0,
    "panEntryMode": "PANAutoEntryViaContactless",
    "pinCaptureCapability": "Unspecified",
    "retrievalReferenceNumber": "654492282074",
    "networkType": "Visa",
    "acquirerNetworkId": "4832",
    "merchantNameLocation": "StreamFlix                               Anywhere",
    "merchantType": "1111",
    "merchantPostalCode": "10001",
    "transactionCurrencyCode": "840",
    "billingCurrencyCode": "840",
    "settlementCurrencyCode": "840",
    "authorizationCode": "489716",
    "transactionIdentifier": "1-10001425931742",
    "transactionUniqueIdentifier": "489716",
  }

```

```

"accountIdentification": "827277613081398",
"partialAuthorizationIndicator": false,
"isChargeBack": false,
"outboundResponseCode": "00",
"messageType": "0100",
"processingCode": "BillPayment",
"expirationDate": "2025-06-21T08:24:43.1966667-04:00",
"billingConversionRate": 0,
"settlementConversionRate": 0,
"bankNetReferenceNumber": "483338568",
"transactionFeeAmount": 0,
"acquiringInstitutionCode": "24341234788",
"forwardingInstitutionCode": "21312234800",
"processor": "i2c",
"memoPostId": "10a2ff3c-ba4c-4947-9876-b30000cc8b31",
"transactionKey": "993D5770CBFB6A66E104AA8EE7E9EF1FE01D2E8D",
"transactionCurrency": "usd",
"billingCurrency": "usd",
"settlementCurrency": "usd",
"postClosureAccountUsed": false,
"relatedTransactionId": "b031aafb-0e5a-490e-8fa6-b30000cc8b20"
},
{
  "id": "d14c88a9-0ade-498c-9329-b30000cca8bf",
  "productId": "57146944-b145-4326-884d-b2f700ecf688",
  "partnerId": "19222b81-0e1e-452d-a842-b2f1011c16f3",
  "accountNumber": "158560897007",
  "createdAt": "2025-06-18T08:25:08.4733333-04:00",
  "description": "(Reversal) 654492282074 StreamFlix
  "isReversal": true,
  "traceNumber": 385910,
  "transmissionDate": "0618122508",
  "panLastFour": "4558",
  "isDebit": false,
  "isPin": false,
  "transactionAmount": 123,
  "billingAmount": 123,
  "settlementAmount": 0,
  "panEntryMode": "PANAUTOENTRYVIACONTACTLESS",
  "pinCaptureCapability": "UNSPECIFIED",
  "retrievalReferenceNumber": "654492282074",
  "networkType": "Visa",
  "acquirerNetworkId": "8384",
  "merchantNameLocation": "StreamFlix
  "merchantType": "1111",
  "merchantPostalCode": "10001",

```

Anywhere

```

    "transactionCurrencyCode": "840",
    "billingCurrencyCode": "840",
    "settlementCurrencyCode": "840",
    "authorizationCode": "840479",
    "transactionIdentifier": "1-10001425931742",
    "transactionUniqueIdentifier": "840479",
    "accountIdentification": "827277613081398",
    "partialAuthorizationIndicator": false,
    "isChargeBack": false,
    "outboundResponseCode": "00",
    "messageType": "0420",
    "processingCode": "BillPayment",
    "expirationDate": "2025-06-21T08:25:08.4733333-04:00",
    "billingConversionRate": 0,
    "settlementConversionRate": 0,
    "bankNetReferenceNumber": "483338568",
    "transactionFeeAmount": 0,
    "acquiringInstitutionCode": "29167599701",
    "forwardingInstitutionCode": "56144299975",
    "processor": "i2c",
    "memoPostId": "10a2ff3c-ba4c-4947-9876-b30000cc8b31",
    "transactionKey": "991705EC38A5523A13B2B63A888B7E9A19554DE8",
    "transactionCurrency": "usd",
    "billingCurrency": "usd",
    "settlementCurrency": "usd",
    "reversalType": "Authorization",
    "postClosureAccountUsed": false,
    "relatedTransactionId": "b031aafb-0e5a-490e-8fa6-b30000cc8b20"
  },
  {
    "id": "e1cdd82c-72ae-4056-b8c2-b30000cdb9e9a",
    "productId": "57146944-b145-4326-884d-b2f700ecf688",
    "partnerId": "19222b81-0e1e-452d-a842-b2f1011c16f3",
    "accountNumber": "158560897007",
    "createdAt": "2025-06-18T08:29:05.5766667-04:00",
    "description": "654492282074 StreamFlix                               Anywhere",
    "isReversal": false,
    "traceNumber": 385910,
    "transmissionDate": "0618122905",
    "panLastFour": "4558",
    "isDebit": true,
    "isPin": false,
    "transactionAmount": 123,
    "billingAmount": 123,
    "settlementAmount": 0,
    "panEntryMode": "PANAUTOENTRYVIACONTACTLESS",

```

```
"pinCaptureCapability": "Unspecified",
"retrievalReferenceNumber": "654492282074",
"networkType": "Visa",
"acquirerNetworkId": "2138",
"merchantNameLocation": "StreamFlix                Anywhere",
"merchantPostalCode": "10001",
"transactionCurrencyCode": "840",
"billingCurrencyCode": "840",
"settlementCurrencyCode": "840",
"authorizationCode": "223597",
"transactionIdentifier": "1-10001425931742",
"transactionUniqueIdentifier": "223597",
"accountIdentification": "827277613081398",
"partialAuthorizationIndicator": true,
"isChargeBack": false,
"outboundResponseCode": "00",
"messageType": "0100",
"processingCode": "BillPayment",
"expirationDate": "2025-06-21T08:29:05.5766667-04:00",
"billingConversionRate": 0,
"settlementConversionRate": 0,
"bankNetReferenceNumber": "483338568",
"transactionFeeAmount": 0,
"acquiringInstitutionCode": "86912943724",
"forwardingInstitutionCode": "28290432101",
"processor": "i2c",
"memoPostId": "10a2ff3c-ba4c-4947-9876-b30000cc8b31",
"transactionKey": "4B698DA7B5B5B1E253C29FFAA09B481E5A5DF197",
"transactionCurrency": "usd",
"billingCurrency": "usd",
"settlementCurrency": "usd",
"postClosureAccountUsed": false,
"relatedTransactionId": "b031aafb-0e5a-490e-8fa6-b30000cc8b20"
},
{
  "id": "170a6467-c547-4f71-bd94-b30000d52f55",
  "productId": "57146944-b145-4326-884d-b2f700ecf688",
  "partnerId": "19222b81-0e1e-452d-a842-b2f1011c16f3",
  "accountNumber": "158560897007",
  "createdAt": "2025-06-18T08:56:10.9466667-04:00",
  "description": "674583031323 StreamFlix                Anywhere",
  "isReversal": false,
  "traceNumber": 562452,
  "transmissionDate": "0618125610",
  "panLastFour": "4558",
  "isDebit": true,
```



```

    "isPin": false,
    "transactionAmount": 22,
    "billingAmount": 22,
    "settlementAmount": 0,
    "panEntryMode": "PANAutoEntryViaContactless",
    "pinCaptureCapability": "Unspecified",
    "retrievalReferenceNumber": "674583031323",
    "networkType": "Visa",
    "acquirerNetworkId": "1498",
    "merchantNameLocation": "StreamFlix                Anywhere",
    "merchantType": "1111",
    "merchantPostalCode": "10001",
    "transactionCurrencyCode": "840",
    "billingCurrencyCode": "840",
    "settlementCurrencyCode": "840",
    "authorizationCode": "160375",
    "transactionIdentifier": "1-10001055423125",
    "transactionUniqueIdentifier": "160375",
    "accountIdentification": "827277613081398",
    "partialAuthorizationIndicator": false,
    "isChargeBack": false,
    "outboundResponseCode": "00",
    "messageType": "0200",
    "processingCode": "Purchase",
    "expirationDate": "2025-06-21T08:56:10.9466667-04:00",
    "billingConversionRate": 0,
    "settlementConversionRate": 0,
    "bankNetReferenceNumber": "564514688",
    "transactionFeeAmount": 0,
    "acquiringInstitutionCode": "77848948385",
    "forwardingInstitutionCode": "19835472422",
    "processor": "i2c",
    "memoPostId": "a3a9a3b4-16ce-4eb0-875d-b30000d52f69",
    "coreTransactionId": "170a6467-c547-4f71-bd94-b30000d52f55",
    "transactionKey": "B40847BB1F5179AD605895146328FBBF205CDADDD",
    "transactionCurrency": "usd",
    "billingCurrency": "usd",
    "settlementCurrency": "usd",
    "postClosureAccountUsed": false,
    "relatedTransactionId": "170a6467-c547-4f71-bd94-b30000d52f55"
  },
  {
    "id": "baecdf9c-dcfa-402f-9062-b30000d710b8",
    "productId": "57146944-b145-4326-884d-b2f700ecf688",
    "partnerId": "19222b81-0e1e-452d-a842-b2f1011c16f3",
    "accountNumber": "158560897007",

```

"createdAt": "2025-06-18T09:03:01.73-04:00".

## 5.1.8. Get profile

---

## Endpoint: /cardmanagement/v1/cards/{id}/profile

Retrieves the profile associated to a card.

PUT

<https://sandbox.crbcos.com/cardmanagement/v1/cards/{id}/profile> 

Request

Response

### PATH PARAMS

**id** String **required**

GUID format card ID in the Cross River system, which is the id attribute value of the response to Create a card. This is not the number that appears on the card.

### BODY PARAMETERS

**firstName** String **required**

Cardholder first name  
60 characters maximum

**middleName** String **required**

Cardholder middle name  
60 characters maximum

**lastName** String **required**

Cardholder last name  
60 characters maximum

**suffix** String **optional**

Cardholder name suffix  
60 characters maximum

**phone ▶**

Object

**required**

Details of the cardholder phone contact information

**emailAddress**

String

**required**

Cardholder email address  
60 characters maximum

**billingAddress ▶**

Object

**required**

Address to send credit card bills

**shippingAddress ▶**

Object

**required**

Address to which a physical card is shipped



Curl



Node.js



Python



Ruby



Go



```
curl --location --globoff --request PUT 'https://sandbox.crbcos.com/cardma
--data-raw '{
  "firstName": "Daly",
  "lastName": "Khol",
  "phone": {
    "phoneType": "Home",
    "phoneNumber": "7185551234"
  },
  "emailAddress": "dkhol@gmail.com",
  "billingAddress": {
    "street1": "1 Roshar Ave",
    "city": "Roshar",
    "state": "NY",
    "postalCode": "10001",
    "countryCode": "US"
  },
  "shippingAddress": {
    "street1": "1 Roshar Ave",
    "city": "Roshar",
    "state": "NY",
    "postalCode": "10001",
    "countryCode": "US"
  }
}'
```

Responses

● 200







## 5.1.9. Suspend card

### Endpoint: /cardmanagement/v1/cards/{id}/suspend

Suspend a card

**POST**

<https://sandbox.crbcos.com/cardmanagement/v1/cards/{id}/susp> 

**Request**

Response

#### PATH PARAMS

**id** String **required**

GUID format card ID in the Cross River system, which is the id attribute value of the response to Create a card. This is not the number that appears on the card.



Curl



Node.js



Python



Ruby



Go



```
curl --location --globoff --request POST 'https://sandbox.crbcos.com/cardm
--header 'Idempotency-key: a5b1b129-421e-441b-9661-ec1c0396965e' \
--header 'Authorization: Bearer eyJhbGciOiJSUzI1NiIsImtpZCI6IjdQREJ4dGVldl
--data ''
```

#### Responses

● 200

● 400



```
{
  "errors": [
    {
      "code": 2011,
      "message": "Card is closed"
    }
  ]
}
```



## 5.1.10. Unsuspend card

---

## Endpoint: /cardmanagement/v1/cards/{id}/unsuspend

Unsuspend a card

**POST**

<https://sandbox.crbcos.com/cardmanagement/v1/cards/{id}/unsu> 

**Request**

Response

### PATH PARAMS

**id** String **required**

GUID format card ID in the Cross River system, which is the id attribute value of the response to Create a card. This is not the number that appears on the card.

### BODY PARAMETERS

**notes** String **optional**

Maximum length is 60 characters



Curl



Node.js



Python



Ruby



Go



```
curl --location --globoff 'https://sandbox.crbcos.com/cardmanagement/v1/ca
--header 'Idempotency-key: a5b1b129-421e-441b-9661-ec1c0396965e' \
--data '{
  "notes": "card unsuspended"
}'
```

## Responses

 200 400

```
{
  "errors": [
    {
      "code": 2005,
      "message": "Card is not suspended"
    }
  ]
}
```

## 5.1.11. Admin block

---

## Endpoint: /cardmanagement/v1/cards/{id}/admin-block

An administrative block means that the card is both suspended and has been blocked using this endpoint. Only an admin can remove this block. Status returned is suspended.

**POST**

<https://sandbox.crbcos.com/cardmanagement/v1/cards/{id}/admin-block> 

**Request**

Response

### PATH PARAMS

**id** String **required**

GUID format card ID in the Cross River system, which is the id attribute value of the response to Create a card. This is not the number that appears on the card.

### BODY PARAMETERS

**notes** String **optional**

Maximum length is 60 characters



Curl



Node.js



Python



Ruby



Go



```
curl --location --globoff 'https://sandbox.crbcos.com/cardmanagement/v1/ca
--header 'Idempotency-key: a5b1b129-421e-441b-9661-ec1c0396965e' \
--data '{
  "notes": "blocked pending review"
}'
```



Responses

● 200





## 5.1.12. Close card

---

## Endpoint: /cardmanagement/v1/cards/{id}/close

Permanently deactivates a card. Closed cards cannot be reactivated.

**POST**

<https://sandbox.crbcos.com/cardmanagement/v1/cards/{id}/close> 

**Request**

Response

### PATH PARAMS

**id** String **required**

GUID format card ID in the Cross River system, which is the id attribute value of the response to Create a card. This is not the number that appears on the card.

### BODY PARAMETERS

**notes** String **optional**

Maximum length is 60 characters



Curl



Node.js



Python



Ruby



Go



```
curl --location --globoff 'https://sandbox.crbcos.com/cardmanagement/v1/ca
--header 'Idempotency-key: a5b1b129-421e-441b-9661-ec1c0396965e' \
--data '{
  "notes": "new card ordered"
}'
```

Responses

● 200





### 5.1.13. Simulate card auth

---



## Endpoint: /cardmanagement/v1/simulate/authorization

Simulates merchant authorization that affects the available balance on the deposit account. To simulate a declined transaction ensure the amount of the authorization is greater than the COS account balance.

**POST**

<https://sandbox.crbcos.com/cardmanagement/v1/simulate/author> 

**Request**

Response

### BODY PARAMETERS

**amount** Integer **required**

Monetary value of the transaction in the transaction currency

**cardId** String **required**

GUID format card ID in the Cross River system, which is the id attribute value of the response to Create a card. This is not the number that appears on the card.

**merchantCountryCode** String **required**

Country code of the merchant

**merchantName** String **required**

Name of the merchant

**merchantStreet** String **required**

Street address of the merchant

**merchantCity** String **required**

City where the merchant is located

**merchantState** String required

State where the merchant is located

**merchantPostalCode** String required

Postal code of the merchant

**cardNetwork** String optional

Payment network used to process the transaction.

Select from:

- Visa
- Mastercard
- i2c

**additionalAmounts** ▶ Object optional

Array with ammounts other than the transaction amount that are relevant to this transactions (e.g. fees).

**processingCode**

String

optional

Code indicating the transaction type and account types involved.

Select from:

- Purchase
- Withdrawal
- DebitAdjustment
- ConversionGuarantee
- ConversionVerification
- TravelerCheck
- CashbackPurchase
- AccountFunding
- QuasiCash
- FundsWithdrawal
- CashDisbursement
- DeferredGoods
- DebitFee
- CreditReturn
- Deposit
- CreditAdjustment
- CheckDepositGuarantee
- CheckDeposit
- CashDeposit
- OriginalCredit
- PaymentTransaction
- CreditDisbursement
- AvailableFundsInquiry
- BalanceInquiry
- AccountVerification
- AtmMiniStatement
- EligibilityInquiry
- CardholderTransfer
- BillPayment
- Payment
- P2PPayment
- PaymentFromThirdParty
- P2PPaymentCredit
- LoanPayment
- PaymentEnclosed
- PrepaidActivation
- PinUnblock
- PinChange
- PV
- PD
- CB
- Q2



Curl



Node.js



Python



Ruby



Go



```
curl --location 'https://sandbox.crbcos.com/cardmanagement/v1/simulate/aut
--data '{
  "amount": 1999,
  "cardId": "8c6a53e0-83ad-4b76-b446-b300006a3e6a",
  "merchantCountryCode": "US",
  "merchantName": "StreamFlix",
  "merchantStreet": "123 Any St",
  "merchantCity": "Anywhere",
  "merchantState": "NY",
  "merchantPostalCode": "10001",
  "cardNetwork": "Visa",
  "additionalAmounts": [],
  "processingCode": "BillPayment"
}'
```

## Responses

● 200



```
{
  "retrievalReferenceNumber": "644371423186",
  "cardId": "8c6a53e0-83ad-4b76-b446-b300006a3e6a"
}
```

## 5.1.14. Simulate card clearing

---

## Endpoint: /cardmanagement/v1/simulate/clearing

Simulates replacement of a memo post with a core transaction.

**POST**

<https://sandbox.crbcos.com/cardmanagement/v1/simulate/clearin> 

### Request

#### BODY PARAMETERS

**cardId** String **required**

GUID format card ID in the Cross River system, which is the id attribute value of the response to Create a card. This is not the number that appears on the card.

**retrievalReferenceNumber** String **optional**

Automatically generated reference (alphanumeric). Card network generated identifier used to uniquely track a transaction across systems.

**systemTraceAuditNumber** String **optional**

Unqie transaction ID assigned by the acquirer of the transaction for audit purposes.

**processingCode**

String

optional

Code indicating the transaction type and account types involved.

Select from:

- Purchase
- Withdrawal
- DebitAdjustment
- ConversionGuarantee
- ConversionVerification
- TravelerCheck
- CashbackPurchase
- AccountFunding
- QuasiCash
- FundsWithdrawal
- CashDisbursement
- DeferredGoods
- DebitFee
- CreditReturn
- Deposit
- CreditAdjustment
- CheckDepositGuarantee
- CheckDeposit
- CashDeposit
- OriginalCredit
- PaymentTransaction
- CreditDisbursement
- AvailableFundsInquiry
- BalanceInquiry
- AccountVerification
- AtmMiniStatement
- EligibilityInquiry
- CardholderTransfer
- BillPayment
- Payment
- P2PPayment
- PaymentFromThirdParty
- P2PPaymentCredit
- LoanPayment
- PaymentEnclosed
- PrepaidActivation
- PinUnblock
- PinChange
- PV
- PD
- CB
- Q2



Curl



Node.js



Python



Ruby



Go



```
curl --location 'https://sandbox.crbcos.com/cardmanagement/v1/simulate/cle
--data '{
  "cardId": "8c6a53e0-83ad-4b76-b446-b300006a3e6a",
  "retrievalReferenceNumber": "644371423186",
  "processingCode": "BillPayment"
}'
```

## Responses



200





## 5.1.15. Simulate card reversal

---

## Endpoint: /cardmanagement/v1/simulate/reversal

This endpoint is used to reverse an existing authorization using the retrievalReferenceNumber returned in the authorization endpoint. This reverses the original authorization placed on the account and removes it from the deposit account activity. The amount in this request can be any value and does not have to be the same amount as the original clearing transaction, which allows for a partial amount reversal.

**POST**

<https://sandbox.crbcos.com/cardmanagement/v1/simulate/reversal> 

### Request

#### BODY PARAMETERS

**cardId** String **required**

GUID format card ID in the Cross River system, which is the id attribute value of the response to Create a card. This is not the number that appears on the card.

**retrievalReferenceNumber** String **optional**

Card network generated identifier used to uniquely track a transaction across systems.

**systemTraceAuditNumber** String **optional**

Unique transaction ID assigned by the acquirer of the transaction for audit purposes.

**amount** Integer **optional**

Monetary value of the transaction in the transaction currency.



Curl



Node.js



Python



Ruby



Go



```
curl --location 'https://sandbox.crbcos.com/cardmanagement/v1/simulate/rev'
--data '{
  "cardId": "8c6a53e0-83ad-4b76-b446-b300006a3e6a",
  "retrievalReferenceNumber": "654492282074",
  "amount": "123"
}'
```

## Responses

● 400



```
{
  "errors": [
    {
      "code": 2350,
      "message": "Authorization already been cleared"
    }
  ]
}
```

## 5.1.16. Simulate incremental

---

## Endpoint: /cardmanagement/v1/simulate/incremental

This endpoint, used with the retrievalReferenceNumber returned by the authorization endpoint, creates an incremental transaction reflected in the deposit account activity. An example of an incremental transaction is when a hotel adds an additional charge for an item charged to your room.

**POST**

<https://sandbox.crbcos.com/cardmanagement/v1/simulate/incremental> 

### Request

#### BODY PARAMETERS

**cardId** String **required**

GUID format card ID in the Cross River system, which is the id attribute value of the response to Create a card. This is not the number that appears on the card.

**retrievalReferenceNumber** String **optional**

Card network generated identifier used to uniquely track a transaction across systems.

**systemTraceAuditNumber** String **optional**

Unique transaction ID assigned by the acquirer of the transaction for audit purposes.

**amount** Integer **optional**

Monetary value of the transaction type and account types involved.



Curl



Node.js



Python



Ruby



Go



```
curl --location 'https://sandbox.crbcos.com/cardmanagement/v1/simulate/inc
--data '{
  "cardId": "8c6a53e0-83ad-4b76-b446-b300006a3e6a",
  "retrievalReferenceNumber": "654492282074",
  "amount": "123"
}'
```

## Responses

● 200



## 5.1.17. Simulate clearing

---

## Endpoint: /cardmanagement/v1/simulate/single-message-clearing

A single message is a message type that uses real-time verification and requires the cardholder to enter their PIN to authorize and clear their transaction. An example of a transaction that uses a single message is an ATM withdrawal.

**POST**

<https://sandbox.crbcos.com/cardmanagement/v1/simulate/single-> 

**Request**

Response



## BODY PARAMETERS

**processingCode** String optional

Code indicating the transaction type and account types involved.

Select from:

- Purchase
- Withdrawal
- DebitAdjustment
- ConversionGuarantee
- ConversionVerification
- TravelerCheck
- CashbackPurchase
- AccountFunding
- QuasiCash
- FundsWithdrawal
- CashDisbursement
- DeferredGoods
- DebitFee
- CreditReturn
- Deposit
- CreditAdjustment
- CheckDepositGuarantee
- CheckDeposit
- CashDeposit
- OriginalCredit
- PaymentTransaction
- CreditDisbursement
- AvailableFundsInquiry
- BalanceInquiry
- AccountVerification
- AtmMiniStatement
- EligibilityInquiry
- CardholderTransfer
- BillPayment
- Payment
- P2PPayment
- PaymentFromThirdParty
- P2PPaymentCredit
- LoanPayment
- PaymentEnclosed
- PrepaidActivation
- PinUnblock
- PinChange
- PV
- PD

- CB
- Q2

**cardId** String required

GUID format card ID in the Cross River system, which is the id attribute value of the response to Create a card. This is not the number that appears on the card.

**amount** Integer required

Monetary value of the transaction in the transaction currency

**merchantCountryCode** String required

County code of the merchant.

**merchantName** String required

Name of the merchant.

**merchantStreet** String required

Street address of the merchant.

**merchantCity** String required

City of the merchant.

**merchantState** String required

State where the merchant is located - 2 letter code format.

**merchantPostalCode** String required

Postal code of the merchant.

**cardNetwork**

String

optional

Payment network used to process the transaction.

Select from:

- Visa
- Mastercard
- i2c

**additionalAmounts** ►

Object

optional

Array with amounts other than the transaction amount that are relevant to this transaction (e.g. fees).



Curl



Node.js



Python



Ruby



Go



```
curl --location 'https://sandbox.crbcos.com/cardmanagement/v1/simulate/sin
--data '{
  "processingCode": "Purchase",
  "cardId": "8c6a53e0-83ad-4b76-b446-b300006a3e6a",
  "amount": "22",
  "merchantCountryCode": "US",
  "merchantName": "StreamFlix",
  "merchantStreet": "123 Any St",
  "merchantCity": "Anywhere",
  "merchantState": "NY",
  "merchantPostalCode": "10001",
  "cardNetwork": "Visa",
  "additionalAmounts": [
    {
      "amountType": "Unknown",
      "accountType": "NotSpecified",
      "amount": "10"
    }
  ]
}
```

## Responses

● 200



```
{
  "retrievalReferenceNumber": "674583031323",
  "cardId": "8c6a53e0-83ad-4b76-b446-b300006a3e6a"
}
```

## 5.1.18. Simulate reversal

---

## Endpoint: /cardmanagement/v1/simulate/single-message-reversal

This endpoint allows you to create a reversal transaction to a single message clearing transaction. For example, use the single-message-clearing endpoint to simulate a purchase and then simulate a return with the single-message-reversal.

**POST**

<https://sandbox.crbcos.com/cardmanagement/v1/simulate/single-> 

**Request**

Response

### BODY PARAMETERS

**amount**

Integer

**required**

Monetary value of the transaction in the transaction currency.

**processingCode**

String

optional

Code indicating the transaction type and account types involved.

Select from:

- Purchase
- Withdrawal
- DebitAdjustment
- ConversionGuarantee
- ConversionVerification
- TravelerCheck
- CashbackPurchase
- AccountFunding
- QuasiCash
- FundsWithdrawal
- CashDisbursement
- DeferredGoods
- DebitFee
- CreditReturn
- Deposit
- CreditAdjustment
- CheckDepositGuarantee
- CheckDeposit
- CashDeposit
- OriginalCredit
- PaymentTransaction
- CreditDisbursement
- AvailableFundsInquiry
- BalanceInquiry
- AccountVerification
- AtmMiniStatement
- EligibilityInquiry
- CardholderTransfer
- BillPayment
- Payment
- P2PPayment
- PaymentFromThirdParty
- P2PPaymentCredit
- LoanPayment
- PaymentEnclosed
- PrepaidActivation
- PinUnblock
- PinChange
- PV
- PD
- CB
- Q2

**cardId** String required

GUID format card ID in the Cross River system, which is the id attribute value of the response to Create a card. This is not the number that appears on the card.

**merchantCountryCode** String required

County code of the merchant.

**merchantName** String required

Name of the merchant.

**merchantStreet** String required

Street address of the merchant.

**merchantCity** String required

City of the merchant.

**merchantState** String required

State where the merchant is located - 2 letter code format.

**merchantPostalCode** String required

Postal code of the merchant.

**cardNetwork** String optional

Payment network used to process the transaction.

Select from:

- Visa
- Mastercard
- i2c



**additionalAmounts** ▶

Object

required

Array with amounts other than the transaction amount that are relevant to this transaction (e.g. fees).



Curl



Node.js



Python



Ruby



Go



```
curl --location 'https://sandbox.crbcos.com/cardmanagement/v1/simulate/sin
--data '{
  "amount": "22",
  "processingCode": "Purchase",
  "cardId": "8c6a53e0-83ad-4b76-b446-b300006a3e6a",
  "merchantCountryCode": "US",
  "merchantName": "StreamFlix",
  "merchantStreet": "123 Any St",
  "merchantCity": "Anywhere",
  "merchantState": "NY",
  "merchantPostalCode": "10001",
  "cardNetwork": "Visa",
  "additionalAmounts": [
    {
      "amountType": "Unknown",
      "accountType": "NotSpecified",
      "amount": "10"
    }
  ]
}
```

**Responses**

● 200



```
{
  "retrievalReferenceNumber": "150323318008",
  "cardId": "8c6a53e0-83ad-4b76-b446-b300006a3e6a"
}
```

## 5.2. Webhook events

When you work with Cross River accounts, cards and payments, you register for specific **webhooks** which have a specific **format**.

This table presents 

cards

 webhook events that are described in detail below.

Event Name	Description
<u>Cards.Card.Ordered</u>	Card has been ordered
<u>Cards.Card.Created</u>	Card created
<u>Cards.Card.Activated</u>	Card has been activated
<u>Cards.Card.Status.Changed</u>	Card state has changed
<u>Cards.Card.Pin.Set</u>	Set PIN for card
<u>Cards.Card.Suspended</u>	Card use temporarily suspended
<u>Cards.Card.Unsuspended</u>	Suspension removed from card
<u>Cards.Card.Profile.Changed</u>	Card profile has been updated
<u>Cards.Card Replaced</u>	Card replaced
<u>Cards.Card.Admin.Suspended</u>	Card suspended by admin
<u>Cards.Card.Closed</u>	Card closed
<u>Cards.Transaction.Authorized</u>	Card transaction approved
<u>Cards.Transaction.Cleared</u>	Card transaction successfully completed
<u>Cards.Transaction.Declined</u>	Card transaction not approved
<u>Cards.Transaction.Reversed</u>	Payment credited to the originating account

## Card Issuing

### Order a card



```
{
  "id": "1a175de4-5e0a-47af-8e56-b2f200d6cf7a",
  "eventName": "Cards.Card.Ordered",
  "status": "Pending",
  "partnerId": "1e5d3f04-ae24-4af6-9e30-aecf012b99dd",
  "createdAt": "2025-06-04T09:02:06.05+00:00",
  "resources": [
    "cardmanagement/v1/cards/e29b514d-a71d-494d-86da-b2f200d6be43"
  ],
  "details": [
    {
      "cardId": "e29b514d-a71d-494d-86da-b2f200d6be43",
      "status": "Unactivated",
      "statusReasonCode": "NotSet",
      "fraudSuspect": "False",
      "adminBlocked": "False"
    }
  ]
}
```

## Create a card



```
{
  "id": "b757dedd-1156-4e7e-9178-b2f200d6cf7b",
  "eventName": "Cards.Card.Created",
  "status": "Pending",
  "partnerId": "1e5d3f04-ae24-4af6-9e30-aecf012b99dd",
  "createdAt": "2025-06-04T09:02:06.067+00:00",
  "resources": [
    "cardmanagement/v1/cards/e29b514d-a71d-494d-86da-b2f200d6be43"
  ],
  "details": [
    {
      "cardId": "e29b514d-a71d-494d-86da-b2f200d6be43",
      "status": "Unactivated",
      "statusReasonCode": "NotSet",
      "fraudSuspect": "False",
      "adminBlocked": "False"
    }
  ]
}
```

## Activate a card



```
{
  "id": "ab1e566b-de09-4004-ba19-b2f200d78196",
  "eventName": "Cards.Card.Activated",
  "status": "Pending",
  "partnerId": "1e5d3f04-ae24-4af6-9e30-aecf012b99dd",
  "createdAt": "2025-06-04T09:04:38.043+00:00",
  "resources": [
    "cardmanagement/v1/cards/e29b514d-a71d-494d-86da-b2f200d6be43"
  ],
  "details": [
    {
      "cardId": "e29b514d-a71d-494d-86da-b2f200d6be43",
      "status": "Active",
      "statusReasonCode": "NotSet",
      "fraudSuspect": "False",
      "adminBlocked": "False",
      "previousStatus": "Unactivated",
      "previousStatusReasonCode": "NotSet",
      "previousFraudSuspect": "False",
      "previousAdminBlocked": "False"
    }
  ]
}
```

**Card status has changed**



```
{
  "id": "07d815aa-696d-4643-8ab9-b2f200d78195",
  "eventName": "Cards.Card.Status.Changed",
  "status": "Pending",
  "partnerId": "1e5d3f04-ae24-4af6-9e30-aecf012b99dd",
  "createdAt": "2025-06-04T09:04:38.043+00:00",
  "resources": [
    "cardmanagement/v1/cards/e29b514d-a71d-494d-86da-b2f200d6be43"
  ],
  "details": [
    {
      "cardId": "e29b514d-a71d-494d-86da-b2f200d6be43",
      "status": "Active",
      "statusReasonCode": "NotSet",
      "fraudSuspect": "False",
      "adminBlocked": "False",
      "previousStatus": "Unactivated",
      "previousStatusReasonCode": "NotSet",
      "previousFraudSuspect": "False",
      "previousAdminBlocked": "False"
    }
  ]
}
```

## Set a card PIN



```
{
  "id": "60207330-ac6d-408d-a71f-b2f200d80d2b",
  "eventName": "Cards.Card.Pin.Set",
  "status": "Pending",
  "partnerId": "1e5d3f04-ae24-4af6-9e30-aecf012b99dd",
  "createdAt": "2025-06-04T09:06:37.157+00:00",
  "resources": [
    "cardmanagement/v1/cards/e29b514d-a71d-494d-86da-b2f200d6be43"
  ],
  "details": [
    {
      "cardId": "e29b514d-a71d-494d-86da-b2f200d6be43",
      "status": "Active",
      "statusReasonCode": "NotSet",
      "fraudSuspect": "False",
      "adminBlocked": "False"
    }
  ]
}
```

**Card has been suspended**



```
{
  "id": "11c8368c-6b2f-42a5-9bb4-b2f200d89d3d",
  "eventName": "Cards.Card.Suspended",
  "status": "Pending",
  "partnerId": "1e5d3f04-ae24-4af6-9e30-aecf012b99dd",
  "createdAt": "2025-06-04T09:08:40.093+00:00",
  "resources": [
    "cardmanagement/v1/cards/e29b514d-a71d-494d-86da-b2f200d6be43"
  ],
  "details": [
    {
      "cardId": "e29b514d-a71d-494d-86da-b2f200d6be43",
      "status": "Suspended",
      "statusReasonCode": "NotSet",
      "fraudSuspect": "False",
      "adminBlocked": "False",
      "previousStatus": "Active",
      "previousStatusReasonCode": "NotSet",
      "previousFraudSuspect": "False",
      "previousAdminBlocked": "False"
    }
  ]
}
```

**Card has been unsuspended**





```
{
  "id": "53979cb2-a5fc-431a-b068-b2f200d92ba9",
  "eventName": "Cards.Card.Unsuspended",
  "status": "Pending",
  "partnerId": "1e5d3f04-ae24-4af6-9e30-aecf012b99dd",
  "createdAt": "2025-06-04T09:10:41.623+00:00",
  "resources": [
    "cardmanagement/v1/cards/e29b514d-a71d-494d-86da-b2f200d6be43"
  ],
  "details": [
    {
      "cardId": "e29b514d-a71d-494d-86da-b2f200d6be43",
      "status": "Active",
      "statusReasonCode": "NotSet",
      "fraudSuspect": "False",
      "adminBlocked": "False",
      "previousStatus": "Suspended",
      "previousStatusReasonCode": "NotSet",
      "previousFraudSuspect": "False",
      "previousAdminBlocked": "False"
    }
  ]
}
```

## Change card profile



```
{
  "id": "9d6bc42e-aa79-4bc7-8d74-b2f200d9ff46",
  "eventName": "Cards.Card.Profile.Changed",
  "status": "Pending",
  "partnerId": "1e5d3f04-ae24-4af6-9e30-aecf012b99dd",
  "createdAt": "2025-06-04T09:13:42.203+00:00",
  "resources": [
    "cardmanagement/v1/cards/e29b514d-a71d-494d-86da-b2f200d6be43"
  ],
  "details": [
    {
      "cardId": "e29b514d-a71d-494d-86da-b2f200d6be43",
      "status": "Active",
      "statusReasonCode": "NotSet",
      "fraudSuspect": "False",
      "adminBlocked": "False"
    }
  ]
}
```

## Card replaced



```
{
  "id": "5fcae666-d3f3-45c3-b67c-b2f200da6c3e",
  "eventName": "Cards.Card.Replaced",
  "status": "Pending",
  "partnerId": "1e5d3f04-ae24-4af6-9e30-aecf012b99dd",
  "createdAt": "2025-06-04T09:15:15.19+00:00",
  "resources": [
    "cardmanagement/v1/cards/7484bc54-47d4-441c-a409-b2f200da5aff"
  ],
  "details": [
    {
      "cardId": "7484bc54-47d4-441c-a409-b2f200da5aff",
      "previousCardId": "e29b514d-a71d-494d-86da-b2f200d6be43",
      "status": "Unactivated",
      "statusReasonCode": "NotSet"
    }
  ]
}
```

## Card suspended by admin



```
{
  "id": "7054e36b-2c8f-401b-8853-b2f200db1c1f",
  "eventName": "Cards.Card.Suspended",
  "status": "Pending",
  "partnerId": "1e5d3f04-ae24-4af6-9e30-aecf012b99dd",
  "createdAt": "2025-06-04T09:17:45.277+00:00",
  "resources": [
    "cardmanagement/v1/cards/7484bc54-47d4-441c-a409-b2f200da5aff"
  ],
  "details": [
    {
      "cardId": "7484bc54-47d4-441c-a409-b2f200da5aff",
      "status": "Suspended",
      "statusReasonCode": "NotSet",
      "fraudSuspect": "False",
      "adminBlocked": "True",
      "previousStatus": "Active",
      "previousStatusReasonCode": "NotSet",
      "previousFraudSuspect": "False",
      "previousAdminBlocked": "False"
    }
  ]
}
```

**Card closed**



```
{
  "id": "ba915833-4fe3-4e41-84c3-b2f200db64be",
  "eventName": "Cards.Card.Closed",
  "status": "Pending",
  "partnerId": "1e5d3f04-ae24-4af6-9e30-aecf012b99dd",
  "createdAt": "2025-06-04T09:18:47.247+00:00",
  "resources": [
    "cardmanagement/v1/cards/7484bc54-47d4-441c-a409-b2f200da5aff"
  ],
  "details": [
    {
      "cardId": "7484bc54-47d4-441c-a409-b2f200da5aff",
      "status": "Closed",
      "statusReasonCode": "NotSet",
      "fraudSuspect": "False",
      "adminBlocked": "False",
      "previousStatus": "Suspended",
      "previousStatusReasonCode": "NotSet",
      "previousFraudSuspect": "False",
      "previousAdminBlocked": "True"
    }
  ]
}
```

# Card Transactions

## Cards.Transaction.Authorized



```
{
  "id": "af816e87-15c4-4bd3-a7cd-b2f3012e950f",
  "eventName": "Cards.Transaction.Authorized",
  "status": "Pending",
  "partnerId": "1e5d3f04-ae24-4af6-9e30-aecf012b99dd",
  "createdAt": "2025-06-05T14:21:40.103+00:00",
  "resources": [
    "cardmanagement/v1/transactions/49e8a49f-6750-49dd-aaff-b2f3012e72d9"
  ],
  "details": [
    {
      "transactionId": "49e8a49f-6750-49dd-aaff-b2f3012e72d9",
      "outboundResponseCode": "00",
      "accountNumber": "2434801953",
      "memoPostId": "fcea0e4d-ff12-415d-9b1b-b2f3012e72ef",
      "coreTransactionId": null,
      "description": "015963889974 StreamFlix          Anywhere",
      "processingCode": "BillPayment",
      "amount": "1999",
      "processor": "i2c",
      "messageType": "0100"
    }
  ]
}
```

## Cards.Transaction.Cleared



```
{
  "id": "c3dd1a56-f1e4-48ed-8dcf-b2f3012f21c5",
  "eventName": "Cards.Transaction.Cleared",
  "status": "Pending",
  "partnerId": "1e5d3f04-ae24-4af6-9e30-aecf012b99dd",
  "createdAt": "2025-06-05T14:23:40.163+00:00",
  "resources": [
    "cardmanagement/v1/transactions/ae6cf986-2cbf-4754-bc5e-b2f3012f050b"
  ],
  "details": [
    {
      "transactionId": "ae6cf986-2cbf-4754-bc5e-b2f3012f050b",
      "outboundResponseCode": "00",
      "accountNumber": "2434801953",
      "memoPostId": "fcea0e4d-ff12-415d-9b1b-b2f3012e72ef",
      "coreTransactionId": "ae6cf986-2cbf-4754-bc5e-b2f3012f050b",
      "description": "015963889974 StreamFlix          Anywhere",
      "processingCode": "BillPayment",
      "amount": "1999",
      "processor": "i2c",
      "messageType": "0200"
    }
  ]
}
```

## Cards.Transaction.Declined



```
{
  "id": "fdc69fac-4e37-436c-9f2e-b2f3012ffa8e",
  "eventName": "Cards.Transaction.Declined",
  "status": "Pending",
  "partnerId": "1e5d3f04-ae24-4af6-9e30-aecf012b99dd",
  "createdAt": "2025-06-05T14:26:45.153+00:00",
  "resources": [
    "cardmanagement/v1/transactions/a47edf85-3417-4fb3-a071-b2f3012fd6e7"
  ],
  "details": [
    {
      "transactionId": "a47edf85-3417-4fb3-a071-b2f3012fd6e7",
      "outboundResponseCode": "51",
      "accountNumber": "2434801953",
      "memoPostId": null,
      "coreTransactionId": null,
      "description": "976310542139 StreamFlix",
      "processingCode": "BillPayment",
      "amount": "700000",
      "processor": "i2c",
      "messageType": "0100"
    }
  ]
}
```

Anywhere

## Cards.Transaction.Reversed





```
{
  "id": "d85606f7-3584-4958-9266-b2f301334d0e",
  "eventName": "Cards.Transaction.Reversed",
  "status": "Pending",
  "partnerId": "1e5d3f04-ae24-4af6-9e30-aecf012b99dd",
  "createdAt": "2025-06-05T14:38:50.913+00:00",
  "resources": [
    "cardmanagement/v1/transactions/94d112f2-7c94-4178-a832-b2f3013343bc"
  ],
  "details": [
    {
      "transactionId": "94d112f2-7c94-4178-a832-b2f3013343bc",
      "outboundResponseCode": "00",
      "accountNumber": "2434801953",
      "memoPostId": "8e29c5d1-627d-4c25-9181-b2f30133342c",
      "coreTransactionId": null,
      "description": "(Reversal) 873023855855 Coffee Xpress",
      "processingCode": "Purchase",
      "amount": "742",
      "processor": "i2c",
      "messageType": "0420"
    }
  ]
}
```

## 5.3. Request and response codes

### Card state and status

This table explains each card state and status. The status appears as a value in the `status` field of the response.

Card state	Card status	Other related fields and their values
Card has never been Activated	Unactivated	
Card Activated	Active	
Card Closed	Closed	
Card Lost	Suspended	<code>StatusReasonCode</code> = <b>Lost</b>
Card Stolen	Suspended	<code>StatusReasonCode</code> = <b>Stolen</b>
Admin Blocked	Suspended	<code>AdminBlocked</code> = <b>true</b>
Fraud Blocked	Suspended	<code>FraudSuspect</code> = <b>true</b> <code>StatusReasonCode</code> = <b>FraudBlocked</b>
Card Suspended by User	Suspended	<code>StatusReasonCode</code> = <b>NotSet</b>
Card Unsuspended by User or Admin	Active	<code>StatusReasonCode</code> = <b>NotSet</b> <code>FraudSuspect</code> = <b>false</b> <code>AdminBlocked</code> = <b>false</b>

### Card category

In response to many card issuing calls, the `category` field indicates the card category.

Category	Description
Debit	Debit card
Credit	Credit card
Prepaid	Prepaid card

## Card status

In response to many card issuing calls, the `status` field indicates the activation status of the card.

## Card status reason

In response to many card issuing calls, the `statusReasonCode` field indicates the reason for the activation status supplied in the status field.

Reason	Description
Not Set	
Damaged	Card damaged
Expired	Card expired
Lost Stolen	Card reported lost or stolen
Fraud Compromised	Card was compromised and closed because of fraud
Return	
New Enrollment	

## Order status

In response to many card issuing calls, the `orderStatus` field indicates the status of the card order.

Value	Status	Description
0	OrderPending	The order is pending fulfillment by the processor
1	Completed	The order has been fulfilled by the processor. For physical cards, the card is being printed and mailed to the cardholder.
2	Failed	The order failed at the processor
3	Authorizing	
4	Authorization Failed	
5	Authorization Completed	

## Instrument type

In response to many card issuing calls, the `paymentInstrument` field indicates the type of card.

Type	Description
Physical MSR	Physical card with a magnetic strip
Physical CC	Physical card with a chip
Physical Contactless	Physical card using RFID
Physical Combo	Physical card with a chip that supports contactless payments
Virtual Pan	Virtual card

## Processing code

In the response to `GET /cardmanagement/v1/transactions`, the `processingCode` field indicates what process took place for a given transaction.

Code	Definition
00	Purchase
01	Withdrawal
02	Debit Adjustment
03	Guarantee with Conversion (POS Check Service) (Future Use)
04	Verification with Conversion (POS Check Service) (Future Use)
06	Traveler Check
09	Purchase with Cash Back
10	Account Funding
11	Quasi-Cash Transaction–Debit or Internet Gambling Transaction
13	Funds Withdrawal for Electronic Purse
17	Cash Disbursement
18	Deferred Goods and Services/Scrip Issue/Conversion Only (POS Check Service) (Future Use)
19	Debit Fee Collection/Deferred Goods and Services With Cash Disbursement
20	Credit Return (of goods)/Credit Transaction/Credit Voucher or Merchandise/Return Authorization (U.S. Only)/Purchase Return/Refund
21	Deposit
22	Credit Adjustment
23	Check Deposit Guarantee
24	Check Deposit
25	Envelope-less Cash Deposit
26	Original Credit
28	Prepaid Activation and Load Prepaid Load/Payment Transaction
29	Credit Funds Disbursement/Primary Credit
30	Available Funds Inquiry/Commercial Deposit

Code	Definition
31	Balance Inquiry
33	Account Updater Code/Account Verification (Future Use)
34	ATM Mini Statement
49	Eligibility Inquiry/Generic Balance Inquiry (Future Use)
40	Cardholder Account Transfer
50	Bill Payment/Payment to Another Party
53	Payment (U.S. only)
54	Payment Debit (P2P)
55	Payment from Third Party
56	Payment Credit (P2P)
58	Payment from Account to Credit/Loan
59	Payment Enclosed
72	Prepaid Activation
91	PIN Unblock
92	PIN Change

## Transaction message type

In the response to `GET /cardmanagement/v1/transactions`, the `messageType` field indicates the message type related to the transaction

Type
Authorization
Authorization Advice
Financial
Financial Advice
Reversal Request
Network Management Request

## Transaction response code

In the response to `GET /cardmanagement/v1/transactions`, the `outboundResponseCode` field indicates the outbound response code related to the transaction

Code	Definition
00	Approved
01	Refer to Card Issuer
02	Refer to Card Issuer Special Condition
03	Invalid Merchant
04	Lost or Stolen Card
05	Do not Honor
06	Error
07	Pickup Card, Special Condition
08	Honor with Identification
10	Approved Partial Amount
12	Invalid Transaction
13	Invalid Amount
14	Invalid Card Number
15	Invalid Issuer
17	Customer Cancellation Reversal
27	Issuer File Update Field Edit Error
30	Format Error
31	Bank Not Supported by Switch - Future Use
32	Partial Reversal
33	Expired Card Pickup
34	Suspect Fraud
39	No Credit Account - Future Use
40	Requested Function Not Supported
41	Lost Card Not Captured
42	No universal account



Code	Definition
43	Stolen Card Pickup
51	Insufficient Funds
52	No Checking Account - Future Use
53	No Savings Account - Future Use
54	Expired Card
55	Invalid PIN
56	No Card Account
57	Transaction not Permitted to Cardholder
58	Transaction Not Permitted to Acquirer
59	Suspected Fraud
61	Exceeds Withdrawal Amount Limit
62	Restricted Card
63	Security Violation
64	Original Amount Incorrect Reversal
65	Exceeds Withdrawal Frequency Limit
68	Response Received Late
70	Invalid Transaction Contact Card Issuer
71	PIN Change Decline
75	Allowed Number of PIN Tries Exceeded
76	Invalid Nonexistent To Account Specified - Future Use
77	Invalid Nonexistent From Account Specified - Future Use
78	Invalid Nonexistent Account Specified General - Future Use
79	Key Exchange Validation Failed
80	System Not Available - Future Use
81	Invalid Transaction PIN Block Format Error

Code	Definition
82	Time Out Issuer
84	Invalid Authorization LifeCycle
85	Approved Account Verification
86	PIN Validation Not Possible
87	Approved Purchase Only
88	Invalid Transaction CVV2 Format Error
89	Bad Expiry Date
91	Issuer Switch Inoperative
92	Route Unable Transaction
93	Transaction Cannot be Completed
94	Duplicate Transmission
96	System Error
97	Already Activated Card
99	Approve Transaction Super Green Path
AT	Auth Host Timed Out
CD	Cryptogram Decline
CN	Invalid Currency
DN	Auth Host Down
E7	Bad CVV2 Expiry Date
GA	General AVS Decline
GC	General Card Decline
GV	General CVV2 Decline
PA	Pre Active Card
PF	Invalid Status Purse
PI	PIN Change Fail Invalid Data

Code	Definition
PL	Bad PIN Invalid PIN Block Length
SA	Inactive Card
SD	Account Closed
TM	Card Technology Mismatch
1A	Strong Customer Authentication Required
AI	ATC Validation Failed

## 5.4. Error codes

---

Code	Description
1000	General exception
2000	General exception
2001	CustomerId was not found or not associated with Card
2002	Card can not be activated
2003	Card was already activated
2004	Card can not be suspended
2005	Card is not suspended
2006	Card is already closed
2007	PIN cannot be set on closed cards
2008	Account relationship type is not eligible for debit cards
2009	Account does not belong to customer
2010	Card Order has not completed processing
2011	Card is closed
2012	Request has not completed
2015	Primary card not found
2020	Card not found
2021	Processor Card Id is invalid
2022	Customer already has a card for this account
2023	Card Order must be complete
2024	Card has not been activated
2025	Card is already suspended
2026	Card is already AdminBlocked
2027	Card Configuration not found
2028	Card Provider not found
2029	Card not found at processor

Code	Description
2030	Card order is not complete
2031	Company Name can only be used on cards associated with business accounts
2032	Card Replacement has not completed
2033	Pin must be four digits
2034	Card has already been replaced
2100	Cards Not Enabled On Product
2101	Card Config Not Found
2102	Card Config Change Not Pending
2103	Card Config Not Associated With Product
2104	Card Config Not Enabled
2105	ShippingType not supported by card configuration
2106	CardConfigStatus must be provided on updates
2150	Change not approved
2151	Product change not pending
2152	Product config change approval requires dual control
2153	Product config required
2200	TCP Error
2301	Account not found
2302	Account is closed
2350	Authorization already been cleared
2351	Auth was not found
2352	Error occurred generating referenceid
2353	Transaction was not found
2354	Transaction has already been reversed

Code	Description
2355	Transaction Category is not valid
2356	Transaction Type is not valid with this Category
2357	Authorization was reversed
2375	PrimaryCustomerId was not found
2376	ParentCustomerId was not found
2377	Bin Sponsorship Card Already Exists
2378	Primary Customer and Parent Customer Must Have The Same PartnerId
2380	Bin Sponsorship Card was not found
3000	General exception
3001	Access denied
3100	Card is locked by admin

## 6. Payments

---

Cross River cannot determine how a receiving financial institution displays payment details to its customers. For information on how fields appear in their banking portal or statements, please contact the receiving FI directly.

### Instant payments

Instant payments enable individuals and businesses to transfer funds that clear and settle within seconds between US bank accounts at different US-based financial institutions. Cross River (CR) facilitates instant payments through three networks: CRNow, RTP® via The Clearing House (TCH), and FedNow.

#### Instant payments

### Card payments

Use Cross River card management APIs to sign up a new payee card, find out details about a payee card, and to activate or deactivate a card.

#### Card payments

### International payments

Cross River offers a set of APIs to enable cross border money movement for International Payments.

#### International payments

### ACH

The Cross River ACH API allows you to both originate and receive payments through the Federal Reserve ACH network.



ACH

## Wires

Use wire transfers for both domestic and international payments. You or your customer can transfer money electronically to a recipient and a recipient can access the money, typically the same day.

Wires

## Checks

Manage both deposit and withdrawal checking for your customers using Cross River APIs.

Checks

## 6.1. Instant payments

---

### Instant payments concepts

Instant payments enable individuals and businesses to transfer funds that clear and settle within seconds between US bank accounts at different US-based financial institutions. Cross River (CR) facilitates instant payments through three networks: CRNow, RTP® via The Clearing House (TCH), and FedNow.

### Requirements for Credit Transfers via API

To initiate a credit transfer via API, you must have the following information:

- The account number funding the transfer.
- The transfer amount.
- The creditor's account number and routing number at the receiving bank.

**Get started** with Cross River APIs.

Once you have access to our **sandbox** you can access these API endpoints in the `rtp` module through **Swagger**.

### GET calls

- Add query parameters to filter the response the API returns to general GET calls.
- Use **pagination** to control presentation of your results.

This table provides an overview of the Instant Payments APIs, described in detail below.

Action	API Call	Description
<a href="#"><u>Credit transfer</u></a>	POST /v1/payments	Initiates a single Instant Payment
<a href="#"><u>Payment request</u></a>	POST /v1/payments/payment-request	Initiates a request for a single Instant Payment
<a href="#"><u>Cancel request</u></a>	POST /v1/payments/{id}/payment-request/cancel	Initiate a cancellation of a requested payment
<a href="#"><u>Service info</u></a>	GET /v1/directory	Retrieves a list of instant payment-related services supported by a specific financial institution

Monetary amounts in API calls and responses are formatted as whole numbers, with no decimal point separating dollars and cents.

## Related topics

- [Webhook events](#)
- [Monitor limit utilization](#)
- [Request and response codes](#)
- [Error codes](#)

## Tutorials

- [Send an instant payment](#): Originate a payment using RTP, FedNow, CRNow or network interoperability.
- [Get service info](#): Find out the instant payment services a specific financial institution supports, per network (RTP or FedNow).
- [Set payment expiration](#): Set your own custom expiration time for retrying a payment when a receiving financial institution is down.

- **Fraud reporting**: Each instant payment network has specific requirements for reporting suspected fraud, with its own criteria for what qualifies as reportable fraud

## 6.1.1. **APIs**

### 6.1.1.1. Credit transfer

---

## Endpoint: /rtp/v1/payments

Post a payment by ID. Initiates an instant payment credit transfer. Enables real-time fund transfers from your account to the recipient's account. Transfers can originate from:

- Master account
- Deposit account (DDA)
- Associated subledger

**POST**

<https://sandbox.crbcos.com/rtp/v1/payments>



**Request**

Response

### BODY PARAMETERS

**accountNumber** String **required**

Account that funds the payment

**amount** Integer **required**

Dollar amount of transaction in positive integral cents. For example, write \$1.00 as 100.

**debtor ▶** Object **optional**

Debtor Business/Individual Information. Depending on the configuration, this might be populated automatically.

**ultimateDebtor ▶** Object **optional**

Ultimate debtor business/individual information

**creditor ▶** Object **required**

Creditor business/individual information

**ultimateCreditor ▶**

Object

optional

Ultimate creditor business/individual information

**purpose**

String

optional

Internal CR field used to include a short description of the purpose of the payment. The RDFI does not see this field. 50 characters maximum.

**remittanceData**

String

optional

A description of the payment, meaning, the remittance advice. Use this field for any remittance information that doesn't have its own field. 140 characters maximum.

**remittanceLocation ▶**

Object

optional

Remittance location information

**clientIdentifier**

String

optional

Use this attribute to add your own unique identifying string to a payment call or COS record. This attribute is useful for idempotency purposes. 50 characters maximum.

**networkPlatform**

String

optional

The payment network platform used to process the payment. Possible values are: TCH (for RTP via The Clearing House), FedNow, or CRNow.

**queuedPaymentExpiresAt**

String

optional

Date and time that a queued payment will expire (payment canceled) if the receiving FI is offline, in this format: yyyy-mm-ddThh:mm:ss[.mmm]. Not sent to TCH/Fed.

**queuedPaymentExpiresAfterInSeconds** optional

The time interval (in seconds) after which a queued payment will expire if the receiving FI remains offline. This is not sent to TCH or Fed.

If both queuedPaymentExpiresAt and this attribute are provided, this attribute will take precedence. [preferred by Cross River]





Curl



Node.js



Python



Ruby



Go



```
curl -L 'https://sandbox.crbcos.com/rtp/v1/payments' ^
-d '{
  "accountNumber": "2955057589",
  "amount": 10000,
  "creditor": {
    "routingNumber": "011000138",
    "accountNumber": "456789000",
    "name": "Richard Robert",
    "addressStreetName": "Speedway Blvd",
    "addressBuildingNumber": "3366",
    "addressCity": "Talladega",
    "addressState": "AL",
    "addressPostalCode": "35096",
    "addressCountry": "US"
  },
  "purpose": "Sponsor donation",
  "clientIdentifier": "3e4ac46c-fa9e-4654-80b8-83f971622dfd"
}'
```

Responses

200 400





## 6.1.1.2. Payment request

---

## Endpoint: /rtp/v1/payments/payment-request

Initiates a request for payment (RfP)

Allows the recipient to authorize and trigger an instant payment credit transfer. RfPs can be initiated from:

- Master account
- Deposit account (DDA)
- Associated subledger

**POST**

<https://sandbox.crbcos.com/rtp/v1/payments/payment-request>



**Request**

Response

### BODY PARAMETERS

**accountNumber** String **required**

Account from which the RfP is originated

**amount** Integer **required**

Requested payment amount in positive integral cents. For example, represent \$1.00 as 100.

**purpose** String **optional**

Internal CR field used to include a short description or purpose for the payment request. The RDFI does not see this field. 50 characters maximum.

**requestedExecutionDate** String **required**

Date when the payment request needs to be processed by the Debtor. Formatted as yyyy-mm-ddThh:mm:ss[.mmm].

**expiryDate** String **required**

Expiration date and time of the RfP. If no time is specified, it expires at 12:00 AM on the given date. If a time is provided, it expires within one hour after the specified time on the same date. Formatted as yyyy-mm-ddThh:mm:ss[.mmm].

**clientIdentifier** String optional

Use this attribute to add your own unique identifying string to a payment call or COS record. This attribute is useful for idempotency purposes. 50 characters maximum.

**remittanceData** String optional

A description of the payment request, meaning, the remittance advice. Use this field for any remittance information that doesn't have its own field. 140 characters maximum.

**remittanceLocation ▶** Object optional

Remittance location information

**debtor ▶** Object optional

Debtor Business/Individual Information. The recipient of the RfP who will initiate the credit transfer upon authorization. Depending on the configuration, this information may be populated automatically.

**ultimateDebtor ▶** Object optional

Ultimate Debtor Business/Individual Information. The final party receiving the RfP and responsible for initiating the credit transfer upon authorization.

**creditor ▶** Object optional

Creditor Business/Individual Information. The recipient of the payment, typically the party at CRB initiating the RfP. Depending on the configuration, this information may be populated automatically.

**ultimateCreditor ▶** Object optional

Ultimate Creditor Business/Individual Information. The final recipient of the payment. Depending on the configuration, this information may be populated automatically.

**useCase** String **required**

Use one of the following values for the useCase field:

- BusinessToBusiness
- ConsumerBillPay
- AccountToAccount
- DownPaymentOrFinalPayment
- ConsumerToGovernment

**industryCategory** String **optional**

Use one of the following values for the industryCategory field (optional when useCase is BusinessToBusiness or ConsumerToGovernment):

- HomeUtilities
- TelecomUtilities
- Mortgage
- Rent
- InsurancePremiums
- AutoLoans
- PersonalLoans
- CreditCardPayment
- MediaSubscriptions
- Memberships
- FinancialInstitution
- BrokerOrDealer
- Medical
- OnlineGaming
- RealEstate
- Government

**senderID** String **required**

Use the senderId field (optional when useCase is BusinessToBusiness or ConsumerToGovernment), provided by Cross River and generated by The Clearing House (TCH).

Valid senderID's for Sandbox Testing:

- 1000000000001
- 1000000000002
- 1000000000003
- 1000000000004
- 1000000000005



Curl



Node.js



Python



Ruby



Go



```
curl --location 'https://sandbox.crbcos.com/rtp/v1/payments/payment-reques'
--header 'Content-Type: application/json' \
--data '{
  "accountNumber": "2564791479",
  "amount": 20000,
  "discount": 1000,
  "debtor": {
    "routingNumber": "071212128",
    "accountNumber": "2758463448",
    "name": "Terry Tester",
    "birthCity": "Pine Brook",
    "birthCountry": "US",
    "birthDate": "1996-08-09",
    "addressStreetName": "Linwood Ave",
    "addressBuildingNumber": "2115",
    "addressCity": "Pine Brook",
    "addressState": "NJ",
    "addressPostalCode": "07024",
    "addressCountry": "US"
  },
  "creditor": {
    "accountNumber": "2564791479",
    "name": "the creditor"
  },
  "purpose": "Test Payment 2",
  "clientIdentifier": "123456",
  "requestedExecutionDate": "2025-08-10",
  "expiryDate": "2025-08-16",
  "senderId": "1000000001",
  "useCase": "BusinessToBusiness",
  "industryCategory": "OnlineGaming"
}'
```



Responses

200 400



```
{
  "id": "8ee74810-45fe-47ee-9569-b30500b8d6ae",
  "originalPaymentId": "8ee74810-45fe-47ee-9569-b30500b8d6ae",
  "referenceId": "R1740I808VYXJ",
  "accountNumber": "2955057589",
  "clientIdentifier": "123456",
  "amount": 19000,
  "direction": "Outbound",
  "status": "Created",
  "paymentType": "RequestForPayment",
  "source": "Api",
  "transactionAccountContext": "NotSubmitted",
  "debtor": {
    "routingNumber": "011000138",
    "accountNumber": "2758463448",
    "name": "Terry Tester",
    "birthCity": "Pine Brook",
    "birthCountry": "US",
    "birthDate": "1996-08-09",
    "addressStreetName": "Linwood Ave",
    "addressBuildingNumber": "2115",
    "addressCity": "Pine Brook",
    "addressState": "NJ",
    "addressPostalCode": "07024",
    "addressCountry": "US"
  },
  "creditor": {
    "routingNumber": "021214891",
    "accountNumber": "2955057589",
    "name": "Dim Mak",
    "addressStreetName": "400 Kelby St",
    "addressCity": "Fort Lee",
    "addressState": "NJ",
    "addressPostalCode": "07024",
    "addressCountry": "US"
  },
  "network": {
    "clearingSystemCode": "TCH",
    "currency": "USD",
    "endToEndId": "6909c29416f0453fbdfdb30500b8d6ae",
    "instructionId": "20250623021214273T1B19TS07136537534",
    "interbankSettlementAmount": 19000,
    "interbankSettlementDate": "2025-06-23",
    "messageId": "M20250623021214273T1BDQK15672530466",
    "numberOfTransactions": 1,
  }
}
```

```
"settlementMethod": "CLRG",  
"businessMessageId": "B20250623021214273T1BPRK46725920502",  
"toParticipantId": "990000001S1",
```

### 6.1.1.3. Cancel request

---

## Endpoint: /rtp/v1/payments/{id}/payment-request/cancel

Cancels a payment request if a credit transfer or payment request response has not been received.

**POST**

<https://sandbox.crbcos.com/rtp/v1/payments/{id}/payment-reque> 

**Request**

Response

### PATH PARAMS

**Id** String **required**

The payment ID. This first appears in the initial response to canceling a payment request. This ID is in GUID format.



### BODY PARAMETERS

**reasonCode** String **required**

Reason code for payment request cancellation. Specifies the justification for the cancellation. Please refer to the Request and Response Codes page for more details.

**clientIdentifier** String **optional**

A unique identifier assigned to the payment call or COS record. This value reflects the identifier provided in the original request and helps ensure idempotency.

 Curl Node.js JS Python Ruby Curl Node.js J 

```
curl --location --globoff '/v1/payments/{id}/payment-request/cance/{Id}' \  
--header 'Accept: application/json' \  
--header 'Content-Type: application/json' \  
--data '{"reasonCode":"String","clientIdentifier":"String"}
```

Responses

● 200



Refer to **Request and response codes** for a complete list of `reason code` values.



# 6.1.1.4. Service info

---

## Endpoint: /rtp/v1/directory

Retrieves a list of instant payment services supported by a specific financial institution. If the bank supports TCH and FedNow instant payments, service codes for both networks are returned. For FedNow, an FI must register each RTN separately. An FI can have some but not all RTNs registered for FedNow.

GET

<https://sandbox.crbcos.com/rtp/v1/directory>



Request

Response

### QUERY PARAMETERS

**name** String optional

The name of the financial institution.

**routingNumber** String optional

The routing number of the specific account at the financial institution receiving the payment, such as the number found on a check

**participantId** String optional

The 11-digit identification number assigned to a participant in the payment network

**institutionRoutingNumber** String optional

The routing number that uniquely identifies the financial institution

**networkPlatform** String optional

The payment network platform used to process the payment. Valid values are:

- TCH – RTP via The Clearing House
- FedNow – Federal Reserve's instant payment network



Curl



Node.js



Python



Ruby



Go



```
curl -L -X GET 'https://sandbox.crbcos.com/rtp/v1/directory' ^  
-H 'Accept: application/json' ^  
-H 'Content-Type: application/json' ^  
-d '{"reasonCode":"String","clientIdentifier":"String"}'
```

Responses

● 200



```
{  
  "routingNumber": "2000000009",  
  "participantId": "2000000020T1"
```

## 6.1.2. Webhook events

When you work with Cross River accounts, cards and payments, you register for specific webhooks which have a specific event format.

This table presents common `instant payments` webhook events that are described in detail below.

Event Name	Description
<u><a href="#">Rtp.Payment.Sent</a></u>	The RTP payment has been successfully transmitted to the receiving institution, and the funds are now available in the receiver's bank account
<u><a href="#">Rtp.Payment.Received</a></u>	An inbound payment has been received from another institution and successfully posted to an account in COS
<u><a href="#">Rtp.Payment.Rejected</a></u>	<ul style="list-style-type: none"><li>• <b>Payment Rejected</b> – The payment was rejected by the receiving institution or RTP Network.</li><li>• <b>Insufficient Funds</b> – The payment could not be processed because the originating account had insufficient funds.</li></ul>
<u><a href="#">Rtp.Payment.Canceled</a></u>	<ul style="list-style-type: none"><li>• <b>Partner Cancellation</b> – The payment was canceled at the request of the partner.<ul style="list-style-type: none"><li>◦ <b>Research Cancellation</b> – A payment with a status of <b>ResearchRequired</b> has been canceled.</li></ul></li></ul>
<u><a href="#">Rtp.Limits.Utilization.Changed</a></u>	Track your usage and remaining limit. <u><a href="#">Additional information.</a></u>

## Event examples

### Rtp.Payment.Sent



```
{
  "id": "6be43fe7-043b-43be-ba64-b1c6012ee9e5",
  "eventName": "Rtp.Payment.Sent",
  "status": "Pending",
  "partnerId": "1e5d3f04-ae24-4af6-9e30-aecf012b99dd",
  "createdAt": "2024-08-08T14:22:52.507-04:00",
  "resources": [
    "rtp/v1/payments/469ce76c-6d74-4dfc-9b8a-b1c6012edbf8"
  ],
  "details": [
    {
      "paymentId": "469ce76c-6d74-4dfc-9b8a-b1c6012edbf8",
      "paymentType": "CreditTransfer",
      "resultCode": "OK",
      "resultAdditionalInfo": null,
      "coreTransactionId": "d1ec6120-7151-4af9-b56d-b1c6012edc3e",
      "memoPostId": "469ce76c-6d74-4dfc-9b8a-b1c6012edbf8",
      "accountNumber": "2212135061",
      "postingCode": "OK",
      "rtpTransactionStatus": "ACTC",
      "purpose": "Sponsor donation",
      "awaitingResponse": "False",
      "referencedPaymentId": null,
      "networkPlatform": "TCH",
      "amount": "3500",
      "clientIdentifier": "41x30e25-a0a9-42e0-ac89-845520508c85"
    }
  ]
}
```

## Rtp.Payment.Received



```
{
  "id": "3f4968a3-1425-47bf-a97e-b2980109dee1",
  "eventName": "Rtp.Payment.Received",
  "status": "Pending",
  "partnerId": "ba9d0234-26e9-43a2-9070-aedd0108eadb",
  "createdAt": "2025-03-06T11:08:02.303+00:00",
  "resources": [
    "rtp/v1/payments/b06839a7-08c8-4918-8610-b2980109d7b2"
  ],
  "details": [
    {
      "paymentId": "b06839a7-08c8-4918-8610-b2980109d7b2",
      "paymentType": "CreditTransfer",
      "resultCode": "OK",
      "resultAdditionalInfo": null,
      "coreTransactionId": "a64f9eea-dce1-4809-aa0d-b2980109da6d",
      "accountNumber": "138394490138",
      "postingCode": "OK",
      "purpose": "Credit Transfer",
      "amount": "15000",
      "awaitingResponse": "False",
      "referencedPaymentId": null,
      "networkPlatform": "TCH",
      "clientIdentifier": null,
      "debtorName": "string",
      "debtorAccountNumber": "9877987977",
      "debtorRoutingNumber": "020202020",
      "remittanceData": null
    }
  ]
}
```

## Rtp.Payment.Rejected





```
{
  "id": "911ff705-e28d-4e3f-80a9-b1c6012ee9e5",
  "eventName": "Rtp.Payment.Rejected",
  "status": "Pending",
  "partnerId": "1e5d3f04-ae24-4af6-9e30-aecf012b99dd",
  "createdAt": "2024-08-08T14:22:52.507-04:00",
  "resources": [
    "rtp/v1/payments/9097797b-d731-4f60-8d07-b1c6012eca94"
  ],
  "details": [
    {
      "paymentId": "9097797b-d731-4f60-8d07-b1c6012eca94",
      "paymentType": "CreditTransfer",
      "resultCode": "1100",
      "resultAdditionalInfo": "Simulator - Reject Payment Test",
      "postingCode": "OK",
      "rtpTransactionStatus": "RJCT",
      "purpose": "Sponsor donation",
      "referencedPaymentId": null,
      "networkPlatform": "TCH",
      "amount": "3000",
      "clientIdentifier": "41x30e25-a0a9-42e0-ac89-845520508c85",
      "direction": "Outbound"
    }
  ]
}
```

## Rtp.Payment.Canceled



Sample Event



```
{
  "id": "dcfa23b6-0a58-49c0-8d7c-b1c6012e16da",
  "eventName": "Rtp.Payment.Canceled",
  "status": "Pending",
  "partnerId": "1e5d3f04-ae24-4af6-9e30-aecf012b99dd",
  "createdAt": "2024-08-08T14:19:52.42-04:00",
  "resources": [
    "rtp/v1/payments/38ad12ed-bbe8-4c3d-8e98-b1c6012dd420"
  ],
  "details": [
    {
      "paymentId": "38ad12ed-bbe8-4c3d-8e98-b1c6012dd420",
      "paymentType": "CreditTransfer",
      "resultCode": "OK",
      "resultAdditionalInfo": null,
      "postingCode": "OK",
      "rtpTransactionStatus": null,
      "purpose": "Sponsor donation",
      "referencedPaymentId": null,
      "networkPlatform": "TCH",
      "amount": "1500",
      "clientIdentifier": "41x30e25-a0a9-42e0-ac89-845520508c85",
      "direction": "Outbound"
    }
  ]
}
```

## Rtp.Limits.Utilization.Changed



Sample Event



```
"eventName": "Rtp.Limits.Utilization.Changed",
"metaFields":
[
  "productId",
  "utilizationPercent",
  "previousUtilizationPercent"
]
```



## 6.1.3. Monitor limit utilization

Cross River sets a **daily send limit** for Instant Payments per partner to ensure smooth operations and prevent disruptions. Tracking utilization helps manage limits and prevent payment disruptions. We provide a webhooks to monitor usage and be aware when you are approaching your daily send limit.

Use the `RTP.Limits.Utilization.Changed` webhook to track your usage and remaining limit.

- The webhook notifies you of:
  - **utilizationPercent** – The percentage of your limit that has been used.
  - **previousUtilizationPercent** – The percentage of your limit used before the most recent webhook event.
  - The webhook triggers when utilization reaches 5% and fires at every 5% increment (10%, 15%, 20%, etc.).

Regularly monitoring limit utilization helps prevent disruptions to your Instant Payment services.

`RTP.Limits.Utilization.Change`



```
"eventName": "Rtp.Limits.Utilization.Changed",
"metaFields":
[
  "productId",
  "utilizationPercent",
  "previousUtilizationPercent"
]
```

### Webhook Notifications

- [Register](#) for the `RTP.Limits.Utilization.Changed` webhook.

## 6.1.4. Request and response codes

---

### Payment status

The payment status is reflected in the `status` field of Instant Payment responses. Below is a breakdown of the possible statuses for instant payments:

Status	Description
Created	Initial status for both inbound and outbound payments.
Pending	Outbound – Payment is under validation before submission to the RTP network. Inbound – Payment has been received and is awaiting authorization.
Processing	Outbound – Payment has been submitted to the Instant Payments network. Inbound – Payment is authorized and awaiting confirmation of successful processing by the network.
Completed*	Outbound – Payment accepted and received by the receiving institution. Inbound – Payment has been received and accepted. <i>For credit transfers, this also indicates the payment has successfully posted.</i>
Rejected*	Outbound – Payment was rejected by the receiving institution or RTP network. Inbound – Payment was manually or automatically rejected. <i>Rejection reasons are found in the Result.Code field.</i>
Canceled*	Outbound payment canceled by an internal user while in Hold or ResearchRequired status. <i>Payments can only be canceled in these statuses.</i>
Hold	Payment is under review by the Operations Team.
TimedOut*	Payment was not acknowledged within the SLA window with The Clearing House.
Failed*	Payment failed due to technical issues.
Finalizing	Payment is in the process of posting to an account.
ResearchRequired	No response was received for an outbound payment. Manual review is required to determine if the payment should be canceled or completed.

\* Indicates a final status for a payment.

## Payment types

The `paymentType` field in Instant Payments responses specifies the nature of the transaction. This value provides details about the type of payment or request being processed, indicating whether it involves monetary movement or serves as a non-monetary notification or request.

Type	Description
CreditTransfer	Payment sent by a Debtor FI to a Creditor FI.
ReturnRequest	The originator of the original payment is requesting the funds be returned. This is a non-monetary transaction.
ReturnResponse	Response to a Return of Funds request. This is a non-monetary transaction. The actual money movement to return funds would be done in another payment as a credit transfer.
SystemTimeout	Notification to the Creditor FI that a Credit Transfer has timed out. This is a non-monetary transaction.
RemittanceAdvice*	Allows the Debtor to send stand-alone remittance information to the Creditor or the Creditor to send detailed invoice information to the Debtor (B2B or B2C). This is a non-monetary transaction.
RequestForInformation*	Used to request additional details related to a Credit Transfer or a Request for Payment that has been received. This is a non-monetary transaction.
PaymentAck*	Acknowledges that a Credit Transfer has been received and applied. A Creditor FI must use a Payment Acknowledgement to confirm the funds associated with a Credit Transfer (accepted without posting) have posted to the Creditor's account. This is a non-monetary transaction.
RequestForInformationResponse*	Provides requested additional information as an amendment to the original Request for Payment or Credit Transfer. This is a non-monetary transaction.
Unknown	Payment type was not recognized.

\* *This payment type is not currently supported.*

# Network platform

Network Platform	Description
TCH	The pacs.008 credit transfer is routed through the RTP® network via The Clearing House (TCH).
FedNow	The pacs.008 credit transfer is routed through the FedNow® network.
CRNow	The pacs.008 credit transfer is routed internally through Cross River's CRNow network for partner payments.
No value	The credit transfer is routed according to <a href="#">Network interoperability</a>

## Reject/Reason Codes

The `resultCode` in Instant Payments responses explains why a payment didn't complete. If successful, the `resultCode` will show `OK`.



Code	Description
650	Cannot parse the message
690	Signature mismatch or verification error
OK	Ok
BLKD	Payment has been blocked
AC02	Debtor account is invalid
AC03	Creditor account is invalid
AC04	Account closed
AC06	Account is blocked
AC07	Creditor account closed
AC10	Debtor account currency is invalid or missing
AC11	Creditor account currency is invalid or missing
AC13	Debtor account type missing or invalid
AC14	Creditor account type missing or invalid
AG01	Transaction forbidden on this type of account
AG03	Transaction type not supported/authorized on this account
AGNT	Incorrect Agent
AM02	Transaction amount exceeds allowed maximum
AM04	Insufficient funds to cover message amount
AM09	Amount received is not as agreed or expected
AM11	Transaction currency is invalid or missing
AM12	Amount is invalid or missing
AM13	Transaction amount exceeds clearing system limits
AM14	Transaction amount exceeds limits set by bank and client
BE04	Creditor's address missing or incorrect
BE06	End customer not known or no longer exists

Code	Description
BE07	Debtor's address missing or incorrect
BE10	Debtor country code is missing or invalid
BE11	Creditor country code is missing or invalid
BE13	Debtor's residence country code missing or invalid
BE14	Creditor's residence country code missing or invalid
BE16	Debtor identification code missing or invalid
BE17	Creditor identification code missing or invalid
DS24	Waiting time expired due to incomplete order
DT04	Future date not supported
DUPL	Duplicate payment detected
DS0H	Signer not authorized for this account
FF02	Syntax error in narrative information
FF03	Invalid Payment Type Information
FF08	End to End Id missing or invalid
MD07	End customer is deceased
NARR	Narrative information provided for reason
RC01	Bank identifier code format is incorrect
RC02	Bank identifier is invalid or missing
RC03	Debtor FI identifier is invalid or missing
RC04	Creditor FI identifier is invalid or missing
TM01	Invalid Cut Off Time
TK01	Invalid Token
TK02	Sender Token Not Found
TK03	Receiver Token Not Found
TK04	Token Expired

Code	Description
TK05	Token Counterparty Mismatch
TK06	Token Value Limit Rule Violation
TK07	Single Use Token Already Used
TK08	Token Suspended
INSF	Insufficient funds for outbound message
NOAT	Customer account does not support this message type
1100	Reason provided in additional information
9909	Central Switch (RTP) system malfunction
9910	Instructed Agent signed-off
9912	Recipient connection unavailable
9934	Instructing Agent signed-off
9946	Instructing Agent suspended
9947	Instructed Agent suspended
9948	Central Switch (RTP) service suspended

## Return Response Reject Codes

The `returnResponseRejectCodes` attribute specifies the reason for rejecting a return response. This applies only to RTP via TCH.

Code	Description
AC04	Closed Account Number
AM04	Insufficient Funds
ARDT	Already Returned
CUST	Customer Decision
LEGL	Legal Decision
NOAS	No Answer From Customer
NOOR	No Original Transaction Received

## Return Request Statuses

The `returnRequestStatuses` attribute indicates the status of a return request. This applies only to RTP via TCH.

Code	Description
IPAY	Payment will be refunded
RJCR	Cancellation request has been rejected
PECR	Payment will be partially refunded

## Return Request Codes

The `returnRequestCodes` attribute defines the reason for initiating a return request. This applies only to RTP via TCH.

Code	Description
AC03	Invalid Creditor Account Number - Requested by the customer due to mistake or error
AM09	Wrong Amount - Amount of the Credit Transfer or Request for Payment is incorrect
CUST	Requested By Customer - Cancellation requested by the Debtor due to mistake or error
DUPL	Duplicate Payment - Identical to another transaction
FRAD	Fraudulent Origin - Debtor claims the payment was unauthorized or fraudulently induced
FRTR	Final Response - Repeat attempt due to prior non-response
TECH	Technical Problem - Cancellation requested due to technical issues
UPAY	Undue Payment - Payment was made through another channel (for Request for Payment expiry)
WIAM	Wrong Amount with Indemnity
WICT	Requested By Customer with Indemnity
WIDP	Duplicate Payment with Indemnity
WIFD	Fraudulent Origin with Indemnity
WIFT	Final Response with Indemnity
WITH	Technical Problem with Indemnity

## FedNow Return Request Codes

The `fedNowReturnRequestCodes` attribute defines return request reasons for FedNow transactions.

Code	Description
AC03	Invalid Creditor Account Number - Requested by the customer due to mistake or error
AM09	Wrong Amount - Amount of the Credit Transfer or Request for Payment is incorrect
AGNT	Requested By Agent
CUST	Requested By Customer - Cancellation requested by the Debtor due to mistake or error
CUTA	Return Request Upon Information Request
DUPL	Duplicate Payment
FRAD	Fraudulent Origin - Debtor claims the payment was unauthorized or fraudulently induced
NARR	Narrative
SVNR	Service Not Rendered
TECH	Technical Problem - Cancellation requested due to technical issues
UPAY	Undue Payment - Payment was made through another channel (for Request for Payment expiry)

## Return Clearing Channels

The `returnClearingChannels` attribute defines the clearing channel used for processing returns.

Code	Description
MPNS	ACH Payment
RTGS	FedWire
RTNS	CHIPS

## Payment Acknowledgment Types

The `paymentAckTypes` attribute defines the types of acknowledgments in response to a payment transaction.

Code	Description
ACK	Acknowledgment from Creditor (end-user) to the sender of the Credit Transfer
ACWP	Acknowledgment from Creditor FI that funds have been posted to the Creditor Account following an ACWP transaction status

## Request for Payment Response Codes

The `requestForPaymentResponseCodes` attribute defines the possible responses to a request for payment. This applies only to RTP via TCH.

Code	Description
AC06	Account specified is blocked, prohibiting transactions
AG01	Transaction forbidden on this type of account
AG03	Transaction type not supported or authorized on this account
AM09	Amount received is incorrect
AM14	Transaction amount exceeds limits agreed between bank and client
BE04	Missing or incorrect creditor address required for payment
BE07	Missing or incorrect debtor address required for payment
CH11	Creditor Identifier Incorrect - Customer indicates creditor is unknown
CUST	Requested By Customer - Customer declines payment
DS04	Order Rejected - The bank rejected the order due to content issues
MD07	End customer is deceased
NARR	Narrative reason provided in additional information
1100	Other Reasons - Not covered by specific codes
SL12	Debtor does not wish to receive RFPs from this Creditor

# Request for Payment Cancellation Codes

The `requestForPaymentCancellationCodes` attribute defines the reasons for canceling a request for payment. This applies only to RTP via TCH.

Code	Description
AC03	Invalid Creditor Account Number
AM09	Incorrect Amount
CUST	Requested By Customer - Cancellation requested
DUPL	Duplicate Payment
FRAD	Fraudulent Payment - Unauthorized or fraudulently induced
TECH	Technical Problem - Cancellation due to system issues
UPAY	Undue Payment - Payment was made through another channel
AC14	Invalid or Missing Creditor Account Type
AM06	Amount Too Low
BE05	Unrecognized Initiating Party
FOCR	Following Refund Request
MS02	No Specified Reason - Customer
MS03	No Specified Reason - Agent
NARR	Narrative reason provided in additional information
RR04	Regulatory Reason
RUTA	Return Upon Unable To Apply

# FedNow Request for Payment Response Codes

The `fedNowRequestForPaymentResponseCodes` attribute defines responses for a FedNow request for payment.



Code	Description
AC02	Invalid Debtor Account
AC05	Closed Debtor Account
AC06	Blocked Account
AG01	Transaction Forbidden
AG03	Transaction Not Supported or Authorized on This Account
AM04	Insufficient Funds
AM09	Amount Received is Incorrect
AM14	Transaction Amount Exceeds Limits
BE01	Inconsistent End Customer
BE05	Unknown Initiating Party
CUST	Requested By Customer
DS02	Order Canceled
DT01	Incorrect Date
DUPL	Duplicate Request for Payment
NARR	Narrative reason provided in additional information

## Transaction account context

- `TransactionAccountContext` is an internal CR code that provides extra details about a payment's state within its current status.
- To understand the overall status of a payment in relation to the Instant Payments network, refer to `PaymentStatus`.

Context	Description
NotSubmitted	Payment has not been sent to the RTP Network
Pending	Payment is in a pending state
Processing	Payment is processing
Complete	Payment has completed
Reversal	Transaction is being reversed
MemoPost	Attempting to set a Memo Post
TimedOut	Action has timed-out
Canceled	Action has been canceled
Rejected	Payment has been rejected
AuthOnly	Authorizing a transaction
NotSet	Not Set

## Transaction Status

Network status of the payment ( `rtpTransactionStatus` ) as determined by the receiving institution, this status applies to Credit Transfers and Requests for Payment.

Status	Description	Network platform
ACTC	Payment has been accepted.	TCH, FedNow, CRNow
RJCT	Payment or payment-related message has been rejected.	TCH, FedNow, CRNow
RCVD	Payment-related message has been received by the receiving institution.	TCH
ACWP	Payment instruction within the credit transfer is accepted but not yet posted to the creditor's account.	TCH, FedNow
ACCC	Payment has been posted to the creditor customer's account.	FedNow
ACSC	Accepted settlement is completed.	FedNow
BLCK	A payment previously in ACWP status is now blocked. Funds will not be posted to the creditor's account or returned to the debtor's account. (FedNow specific)	FedNow
PDNG	Payment in ACWP status has not yet posted to the creditor's account. (FedNow specific)	FedNow

## ServiceLevelCode

**SDVA** is the only accepted code for RTP via TCH, indicating that payments must be executed with same-day value to the creditor (processed in seconds for RTP via TCH).

## LocalInstrumentProprietary

For RTP via TCH, this identifies the origination condition of the instruction, allowing the instructed agent to correctly process the transaction.

Code	Description
INTERMEDIARY	Payment sent through a domestic Payment Service Provider – either “Ultimate Debtor” or “Ultimate Creditor” must be present.
STANDARD	Standard RTP payment.

## Category Purpose

Identifies the Debtor/Sender as a business or consumer customer of the Debtor FI.

### RTP via TCH Category Purpose Codes

Code	Description
BUSINESS	Business initiated.
CONSUMER	Consumer initiated.

### FedNow Category Purpose Codes

Code	Description
BIZZ	Business initiated.
CONS	Consumer initiated.
GOVT	Government initiated.

# 6.1.5. Error codes

---

RTP via TCH and FedNow return their own specific error codes. See the error codes value in the `result.code` field of the response.

## RTP via TCH response reject codes

Code	Description
AC04	Closed Account Number
AM04	Insufficient Funds
ARDT	Already Returned
CUST	Customer Decision
LEGL	Legal Decision
NOAS	No Answer From Customer
NOOR	No Original Transaction Received

## FedNow response reject codes

Refer to the [ISO 20022 External Code Set](#) on the ISO website for additional information on FedNow response reject codes.

Error Code	Error Description
E301	Report request failed. Review request, update and resubmit as needed.
E400	ISO message does not exist, can resend with same message ID (or new message ID).
E401	Sender of the payment status request is not authorized to receive transaction status.
E402	Original ISO message referenced was not processed; resend with new unique message ID.
E411	Transaction is still processing; retry after timeout clock has elapsed.
E481	Invalid dates included in the reporting from date and/or to date fields.
E500	Internal processing error; contact FRB Service Support Center.
E600	Balance inquiry is not allowed for this account or by this Participant.
E760	Invalid response sender and receiver. Response sender and receiver must match original transfer request.
E890	Pending transaction status is not valid for this message.
E891	Finalized transaction status is not valid for this message.
E960	Internal error. Contact FRB Services Support Center for details.
E970	Invalid settlement relationship. Sender FI and Receiver FI must have a valid settlement relationship within the FedNow Service to indicate the appropriate Master Account to which transactions will be settled.
E974	Internal processing error; contact FRB Service Support Center.
E980	Invalid FedNow Service profile for the Sender FI RTN. As a result, the RTN is not eligible to send this message type.
E981	Invalid FedNow Service profile for the Receiver FI RTN. As a result, the Receiver FI RTN is not eligible to receive this message type.
E982	Sender FI or Receiver FI does not have a Participant Profile enabled to send and/or receive instant payments; or Receiver FI does not have a Participant Profile enabled to receive Request for Payment (RfP) messages.

Error Code	Error Description
E983	Liquidity Management Transfers (LMTs) were sent outside of supported hours. LMTs can be sent only during specified LMT operating hours.
E984	Sender FI or Receiver FI does not have a Participant Profile enabled for the indicated RTN to send and/or receive LMTs.
E990	Invalid transaction limit. Transaction exceeds the FedNow Service transaction limit, or a lower limit set by the Sender FI.
E996	Reserved response timeout. Insufficient time remains on the payment timeout clock to accommodate Receiver FI reserved response time.
E997	Timeout clock has expired. Processing of the transaction exceeded the payment timeout clock.
E998	<ul style="list-style-type: none"> <li>Receiver FI is not signed onto the FedNow Service and therefore cannot receive instant payments.</li> <li>Sender FI and/or Receiver FI is not yet active on the FedNow Service.</li> </ul>
E999	ISO message has a creation date or timestamp that is in the future from when it was received by the FedNow Service. This is not allowed.
T501	ISO message exceeds the maximum allowable size of 25,000 characters.
T504	Invalid clearing system member ID field. The ID in the "to" component should be the FedNow Service member ID (e.g., "021150706").
T505	ISO message does not meet the FedNow Service ISO 20022 message specifications for one or multiple fields.
T506	Invalid message definition identifier. This identifier should be associated with the underlying message and must be a version that FedNow Service supports.
T508	Business processing date/time element is not allowed in the ISO message. This element is used only by the FedNow Service.
T509	Copy duplicate element is not allowed in an ISO message sent by a Participant. This element is used only by the FedNow Service.
T510	Related element is not allowed in an ISO message sent by a Participant. This element is used only by the FedNow Service.

Error Code	Error Description
T512	Invalid message ID. A valid message ID must meet the following criteria: <ul style="list-style-type: none"> <li>Calendar date (CCYYMMDD) is the current date and aligned with the Business Application Header (BAH) date.</li> <li>Connection Party ID (nine alphanumerical characters).</li> <li>Reference sequence assigned by sender (up to 18 characters).</li> </ul>
T514	Structured remittance information field exceeded the maximum length of 4,000 characters.
T515	Message contains both structured and unstructured remittance elements. The FedNow Service only allows one or the other within a single message.
T516	Invalid sending party or sending party information. A valid sending party is a Participant or a Service Provider that is authorized to send messages on behalf of the Participant.
T517	ISO message ID is a duplicate. The message ID must be unique.
T518	ISO message does not contain an original transaction ID or original UETR. The ISO message must contain one or both.
T519	ISO message does not contain a phone number in the preferred contact Method (PHON) field. A phone number must be provided when phone is the preferred contact method.
T520	ISO message does not contain a mobile number in the preferred contact method (CELL) field. A mobile number must be provided when cell is the preferred contact method.
T521	ISO message does not contain an email address in the preferred contact method (MAIL) field. An email address must be provided when mail is the preferred contact method.
T522	ISO message does not contain the required status reason code for the rejected transaction.
T523	Invalid transaction status code. ISO messages must include one of the following eligible codes: ACTC ACWP



Error Code	Error Description
	ACCC BLCK PDNG RJCT
T524	Invalid original message name identification. A valid ISO message name as permitted in the ISO 2022 message specification rules is required.
T525	Invalid original interbank settlement amount format. This amount must be: <ul style="list-style-type: none"> <li>• Greater than \$0</li> <li>• Maximum of two decimal places</li> </ul>
T526	Insufficient information provided. When debtor of the return is a party, the debtor party, account and agent must be provided.
T527	Invalid local instrument. The local instrument specifies the FedNow Service product code under which the message is sent. The code should be FDNA.
T528	Invalid returned interbank settlement amount format. This amount must be: <ul style="list-style-type: none"> <li>• Greater than \$0.</li> <li>• Maximum of two decimal places.</li> </ul>
T529	Invalid message signature on ISO 20022 message. For the signature to be valid, it must be signed with the key owner's private key that corresponds to an active public key that was previously created and validated by the FedNow Service.
T530	Invalid country code. See the officially assigned code list on the ISO 20022 website ( <a href="http://www.iso20022.org">www.iso20022.org</a> ).
T532	Invalid interbank settlement amount format. This amount must be: <ul style="list-style-type: none"> <li>• Greater than \$0</li> <li>• Maximum of two decimal places</li> </ul>
T533	Invalid compensation amount. This amount must be: <ul style="list-style-type: none"> <li>• Greater than \$0</li> <li>• Maximum of two decimal places</li> </ul>
T534	ISO message does not contain the required transaction identification or

Error Code	Error Description
	UETR.
T535	Invalid category purpose field. This field must contain either CONS, BIZZ or GOVT.
T536	Insufficient information. If the ISO 20022 message used reason code is NARR, additional information or additional incorrect information must be present, as applicable.
T537	Key signature algorithm not supported.
T538	Account balance request cannot include a reporting period because the balance is as of a particular date and time.
T539	Account balance request must include the account type.
T540	If an account balance report (ABAR), account activity totals report (AATR or IATR) or an account activity details report (AADR) is requested, then the member identification field must be the FedNow Service Connection Party identifier of the requestor of the report(s) and must match the "From" field in BAH.
T542	Invalid Market Practice Identifier
T543	Invalid original message date. Original message date must be the current date or previous calendar date.
T544	Message referenced in admi.007 is not found
T545	Reporting Period and Account Type are not allowed for an intraday account activity totals report (IATR) or a correspondent intraday account activity totals report (CITR).
T546	Invalid Account ID for message type. A correspondent account activity totals report cannot be requested for a single RTN.
T547	Invalid status code. The status code must be TS02.
T548	Invalid charges amount format. This amount must be: <ul style="list-style-type: none"> <li>• Greater than \$0</li> <li>• Maximum of two decimal places</li> </ul>

Error Code	Error Description
T550	<p>Invalid status confirmation code for the return request response message. Must contain one of the following codes:</p> <ul style="list-style-type: none"> <li>• IPAY</li> <li>• RJCR</li> <li>• PDCR</li> <li>• PECR</li> </ul>
T551	<p>Cancellation Status Reason Information component is incorrect:</p> <ul style="list-style-type: none"> <li>• Reason code is required if status confirmation code is 'RJCR.'</li> <li>• Only one instance of the Cancellation Status Reason Information component is allowed</li> </ul>
T553	Proprietary under Reason cannot be populated.
T554	<p>Invalid status code for the information request response message. Must be one of the following codes:</p> <ul style="list-style-type: none"> <li>• IPAY</li> <li>• IDUP</li> <li>• INFO</li> <li>• NINF</li> <li>• PDNG</li> </ul>
T555	Incorrect format in fnclockstart field. Refer to the Technical Specification document for the correct format.
T556	Invalid Correction Transaction component in information request response message. If the correction transaction component is present, the status must be "IPAY."
T557	If an (end-of-day) account activity totals report (AATR), an (end-of-day) account activity details report (AADR), a (end-of-day) correspondent account activity totals report (CATR), or a (end-of-day) correspondent account activity details report (CADR) is requested, then Reporting Period is mandatory, and Account Type is not allowed.

Error Code	Error Description
T558	Invalid Requested Message Name Identification in the camt.060 message. A valid ISO message name as permitted in the ISO 20022 message specification rules is required.
T560	Invalid Transaction Status Code. Must be one of the following codes: <ul style="list-style-type: none"> <li>• ACTC</li> <li>• PRES</li> <li>• RJCT</li> <li>• RCVD</li> </ul>
T561	The Status Reason Information component can only occur once
T562	The Business Service element in the BAH must not be used by participants
T563	Invalid Status Confirmation code. Must be one of the following codes: <ul style="list-style-type: none"> <li>• CNCL</li> <li>• RJCR</li> <li>• PDCR</li> </ul>
T565	If the Resolution Related Information is populated, the status must be IDUP
T566	If the Resolution Related Information is populated, the status must be either IPAY or PECR
T570	Invalid key encoding. The encoding algorithm must be one supported by the FedNow Service.
T571	Invalid key algorithm. The key algorithm must be one supported by the FedNow Service.
T572	Invalid encoded key. Unable to parse the key using the given encoding and key algorithms.
T573	Invalid key length
T574	Invalid key fingerprint
T575	Invalid key expiration. Keys with an expiration date greater than one year are not accepted by the FedNow Service.

Error Code	Error Description
T576	Duplicate key. All keys added must be unique.
T580	Event Time future dated
T581	Error in event code field. If Event Code-FPON/FPOF then Event Parameter must be 9-digit RTN. If Event Code-FPCD/FPCR then Event Parameter must be 7-digit Hexa decimal (connection point ID).
T584	At least one connection point must be connected per Connection party.
T600	Balance inquiry service not available at this time. Or balance inquiry not allowed for this account.
F101	Entry pair matches the Receiver FI's fraud controls
F002	Entry pair matches an entry on the Sender FI's Negative List with a send restriction

## Reason codes

The `reasonCode` attribute in an Instant Payments response provides the reason a message did not complete. The `resultCode` will be **OK** if the payment was successfully sent or received. These codes are specifically for RTP via TCH.

Refer to the [ISO 20022 External Code Set](#) on the ISO website for additional information on FedNow reason codes.

Code	Description
650	Cannot parse the message
690	Signature mismatch or verification error
BLKD	Payment has been blocked
AC02	Debtor account Is Invalid
AC03	Creditor account Is Invalid
AC04	Account closed
AC06	Account is blocked
AC07	Creditor account closed
AC10	Debtor account currency is invalid or missing
AC11	Creditor account currency is invalid or missing
AC13	Debtor account type missing or invalid
AC14	Creditor account type missing or invalid
AG01	Transaction is forbidden on this type of account
AG03	Transaction type is not supported/authorized on this account
AGNT	Incorrect Agent
AM02	Specific transaction/message amount is greater than allowed maximum
AM04	Amount of funds available to cover specified message amount is insufficient
AM09	Amount received is not the amount agreed or expected
AM11	Transaction currency is invalid or missing
AM12	Amount is invalid or missing
AM13	Transaction amount exceeds limits set by clearing system
AM14	Transaction Amount exceeds limits agreed between bank and client
BE01	Inconsistent end customer (FedNow specific)

Code	Description
BE04	Specification of creditor's address, which is required for payment, is missing/not correct
BE06	End customer specified is not known at associated Sort/National Bank Code or no longer exist in the books
BE07	Specification of debtor's address, which is required for payment, is missing/not correct
BE10	Debtor country code is missing or invalid
BE11	Creditor country code is missing or invalid
BE13	Country code of debtor's residence is missing or Invalid
BE14	Country code of creditor's residence is missing or Invalid
BE16	Debtor identification code missing or invalid
BE17	Creditor identification code missing or invalid
DS24	Waiting time expired due to incomplete order
DT04	Future date is not supported
DUPL	Payment is a duplicate of another payment
DS0H	Signer is not allowed to sign for this account
FF02	Syntax error reason is provided as narrative information in the additional reason information
FF03	Invalid Payment Type Information
FF08	End to End Id is missing or invalid
MD07	End customer is deceased
NARR	Reason is provided as narrative information in the additional reason information
RC01	Bank identifier code specified in the message has an incorrect format
RC02	Bank Identified is invalid or missing
RC03	Debtor FI identifier is invalid or missing

Code	Description
RC04	Creditor FI identifier is invalid or missing
RR04	Regulatory reason (FedNow-specific)
TM01	Invalid Cut Off Time
TK01	Invalid Token
TK02	Sender Token Not Found
TK03	Receiver Token Not Found
TK04	Token Expired
TK05	Token Found with Counterparty Mismatch
TK06	Token Found with Value Limit Rule Violation
TK07	Single Use Token Already Used
TK08	Token Suspended
NOAT	Receiving Customer Account does not support/accept this message type
OK	Completed
1100	Any Other Reasons Reason is provided as narrative in the additional information
9909	Central Switch (RTP) system malfunction
9910	Instructed Agent signed-off
9912	Recipient connection is not available
9934	Instructing Agent signed-off
9946	Instructing Agent suspended
9947	Instructed Agent suspended
9948	Central Switch (RTP) service is suspended

## Error codes



Code	Description
2000	General exception
2001	Payment not found
2002	Invalid payment status
2003	Original Payment Not Found
2004	Return Request must be inbound and in a pending state to be rejected
2005	Credit transfers must be outbound in a hold status to be rejected
2006	Payment type can not be rejected
2007	Direction not inbound
2008	Direction not outbound
2009	Payment cannot be canceled
2010	Payment must be outbound
2011	Payment must be inbound
2012	Payment cannot be competed
2013	Routing number isn't a valid RTP Participant
2014	Payment does meet criteria to be resent
2015	Payment had already been refunded
2016	Ultimate Debtor can only be used when account is a business account
2017	Payments of \$3,000 or more require a complete debtor address
2018	Invalid Debtor
2019	Return can not be requested for a Credit Transfer due to payment status
2020	Return can not be requested for an inbound Credit Transfer
2021	Only Credit Transfers can be refunded
2022	Payment already has an existing Return Request in process
2023	Payment must be a Return Response
2024	Payment must be awaiting response

Code	Description
2025	Return Request was not found
2026	Credit Transfer was not found
2027	Cannot manually update payment, funds will be returned through RTP Network
2028	Simulator-Credit Transfer must be inbound
2029	Payment has already been accepted
2030	Credit Transfer status must be Completed
2031	Payment already has an existing Payment Ack in process
2032	Payment can not manually be updated with funds received from an alternate rail
2033	Payment <code>AwaitingResponse</code> already set to submitted status
2034	<code>AwaitingResponse</code> is not a settable property for this payment type
2035	<code>AlternatePaymentDetail</code> is not a settable property for this payment type
2036	Response has already been sent for payment
2037	Invalid amount specified
2038	Payment has expired
2039	<code>ReasonCode</code> is missing or invalid
2040	<code>AdditionalInfo</code> cannot be used with specified <code>ReasonCode</code>
2041	Payment must be a credit transfer
2042	<code>AdditionalInfo</code> is required for specified <code>ReasonCode</code>
2043	Invalid Creditor
2044	ExpiryDate exceeds configured MaxRfpExpiryDays
2045	Receiver does not support Request For Payment
2046	Payment type must be Request For Payment
2047	Requested Execution Date cannot be after Expiry Date

Code	Description
2048	Requested Queued Payment Expires At Date has already passed
2049	Participant is not active
2050	Debtor does not support Return Payment
2051	Unsupported message type on specified network platform
2200	Global profile change not pending
2201	Global profile not found or is invalid
2202	Global Change not approved
2203	Global config change approval requires dual control
2204	Global config required
2205	Global config accounts must be unique
2206	Offset account not found
2207	Unposted account not found
2209	Account change not approved
2210	Account change not pending
2211	This action requires dual control
2212	Account has not been enabled to send credit transfers
2213	Account config change approval requires dual control
2214	Account configuration not found or is invalid
2215	Product change not pending
2216	Product change not approved
2217	Operator Account not found
2218	Product config change approval requires dual control
2219	Account config required
2220	Product config required
2221	Custom originator data not allowed for account

Code	Description
2222	Max payment amount exceeded
2224	Daily Total Dollar Amount Sent Limit Exceeded
2225	Daily Total Dollar Amount Received Limit Exceeded
2226	Daily Sent Transfer Count Limit Exceeded
2227	Daily Received Transfer Count Limit Exceeded
2228	Birth Fields should not be defined on Business Products
2229	Legal Entity Id should not be defined on Personal Products
2230	Enabling Payment Request Receive requires Credit Transfer Send to also be enabled
2231	Enabling Payment Request Send requires Credit Transfer Receive to also be enabled
2232	Account has not been enabled to send Payment Requests
2233	Custom originator data not allowed for account
2234	Daily Sent Payment Request Count Limit Exceeded
2235	FedNow Operator Account not found
2236	Enabling FedNow Payment Request Receive requires FedNow Credit Transfer Send to also be enabled
2237	Enabling FedNow Payment Request Send requires FedNow Credit Transfer Receive to also be enabled
2300	Hold is not active
2301	Hold is not evaluating
2302	Manual holds can only be placed when the payment already has a status of hold
2350	Public key ID already exists
2351	Public key ID does not exist

## 6.2. Card payments

---

### Card payments concepts

Use Cross River card management APIs to issue a new card, find out details about an existing card, as well as activate or deactivate a card.

**Get started** with Cross River APIs.

Once you have access to our **sandbox** you can access these API endpoints in the `card` module through **Swagger**.

### GET calls

- Add query parameters to filter the response the API returns to general GET calls.
- Use **pagination** to control presentation of your results.

## Card

The endpoints listed in this table are described in detail in the sections below.

Action	API Call	Description
<b><u>Add card</u></b>	POST /api/Card	Signs up a new payee card
<b><u>Setup iFrame</u></b>	POST /api/iFrameConfiguration BuildSignupCardUrl	Renders an iFrame into your website. Then, you can add the payee's card details into the frame.
<b><u>Generate iFrame</u></b>	POST /api/V2/iFrameConfiguration GenerateOtcSignupCard	Returns the parameters you need to build an iFrame web address (URL).
<b><u>Card list</u></b>	GET /api/Card	Returns details about payees' cards
<b><u>Card details</u></b>	GET /api/Card/{cardToken}	Returns a payee's card details
<b><u>Deactivate/re activate</u></b>	PUT /api/Card	Turn a payee's card on or off

## Transaction management

Use Cross River card payments transaction management APIs to push or pull funds to or from a debit card. Use the GET endpoints to find out about a specific transaction or many transactions. The POST calls require a credit card token.

Action	API Call	Description
<a href="#"><u>Send funds</u></a>	POST /api/transaction	Sends funds (push) to a specific debit card
<a href="#"><u>Request funds</u></a>	POST /api/PullTransaction	Pulls funds to a specific card from a payer
<a href="#"><u>Push transactions</u></a>	GET /api/transaction	Returns a list of transactions sent from a merchant to a payee
<a href="#"><u>Pull transactions</u></a>	GET /api/PullTransaction	Returns a list of transactions sent from a payee to a merchant
<a href="#"><u>Push transaction by ID</u></a>	GET /api/transaction/{transactionId}	Returns push transaction details of a specific transaction
<a href="#"><u>Pull transaction by ID</u></a>	GET /api/PullTransaction/{transactionId}	Returns pull transaction details of a specific transaction

## Foreign exchange rates

### International transactions concepts

Action	API Call	Description
<a href="#"><u>Exchange rates</u></a>	GET /api/FXRates	For a currency pair, gets updated foreign exchange rate information from the card network

## OFAC

### OFAC screening overview

Use Cross River OFAC screening APIs to verify OFAC compliance, get screening status, and get details about a compliance scan using a scan ID, so you can be sure that individuals don't put you or the bank at risk of fraud or other criminal behavior.

Action	API Call	Description
<a href="#"><u>OFAC compliance</u></a>	POST /api/ofacScan	Checks OFAC compliance before proceeding with an international transaction
<a href="#"><u>OFAC status</u></a>	GET /api/ofacScan/{scanId}/{sourceSenderId}	Returns international transaction screening requirements compliance details
<a href="#"><u>OFAC scan ID</u></a>	GET /api/ofacScan/{scanId}	Returns details about a compliance scan

## Related topics

- [Dynamic descriptor](#)
- [iFrames](#)
- [Webhook events](#)
- [Request and response codes](#)
- [Error codes](#)

## Tutorials

- [Send a push transaction](#): Sign up a card and send funds to that card.
- [Send a pull transaction](#): Sign up a card and pull funds from that card.
- [Set up iFrame](#): Set up a PCI-compliant iFrame. This function will be deprecated. See the [template-based](#) way of generating iFrame code.



## 6.2.1. Card APIs

### 6.2.1.1. Setup iFrame

---

#### IMPORTANT

This function will be deprecated. See the [template-based way](#) of generating iFrame code.

## Endpoint: /api/iFrameConfiguration/BuildSignupCardUrl

Makes an iFrame display on your merchant website. Once the iFrame is added, you can enter payee card information directly into the iFrame.

**POST**

https://pushtopaystaging.crbnj.net/api/iFrameConfiguration/Build? 

Request

Response

### BODY PARAMETERS

**requestId** String **required**

The GUID is a unique identifier that enables the application to link request with response

**customerReferenceNumber** String **required**

4-digit code the merchant assigns

**domain** String **required**

The name of a valid top-level internet domain where consumers will use the iFrame. For example, myfintech.com. This domain must appear in the Cross River allowlist.

**successContinueNavigationPoint** String **optional**

The landing page you are directed to if the sign up was successful

**failureContinueNavigationPoint** String **optional**

The landing page you are directed if the sign-up wasn't successful

**sourceSenderId** String **optional**

The GUID that identifies the merchant partner

**firstName** String optional

Cardholder first name

**lastName** String optional

Cardholder last name

**address1** String optional

Primary location details of cardholder. For instance, street name, house or building number, and PO box. Maximum 100 characters.

**address2** String optional

Secondary location details of cardholder. For instance, number of apartment or floor. Maximum 100 characters.

**city** String optional

Name of the cardholder's city

**state** String optional

2-letter code of the cardholder's state

**countryCode** String optional

2-letter country code

**zipCode** String optional

- US: 5 digits, A ZIP+4 code with optional space or dash separator (12345, 12345-6789, 12345 6789, 123456789).
- Canada: A valid canadian zip code
- Other: up to 9 chars , letters, numbers, spaces, dashes. Do not contain double spaces or dashes

**email** String optional

A valid cardholder email address, for example, me@mailprovider.com

**phoneNumber** String optional

Optional + at the start, 7-15 characters made of: digits, spaces, dashes, brackets

**showOptionalFields** Boolean optional

True if optional fields will appear in the iFrame, otherwise false

```
curl
-X POST
--header 'Content-Type: application/json'
--header 'Accept: application/json'
--header 'Authorization: Bearer token'
-d '{ \
  "requestId": "A2996348-159D-4C53-89E2-2EE89CE71E7C", \
  "customerReferenceNumber": "4457", \
  "domain": "https://myfintech.com", \
  "successContinueNavigationPoint": "string", \
  "failureContinueNavigationPoint": "string", \
  "sourceSenderId": "23a02763-f976-48f9-8b60-938209bfbf30", \
  "firstName": "string", \
  "lastName": "string", \
  "address1": "string", \
  "address2": "string", \
  "city": "string", \
  "state": "string", \
  "countryCode": "string", \
  "zipCode": "string", \
  "email": "string", \
  "phoneNumber": "string", \
  "showOptionalFields": true \
}' 'https://pushtopaystaging.crbnj.net/api/IFrameConfiguration/BuildSi
```

## Responses

● 200



```
{
  "version": "1.0.0.0",
  "result": "Success"
  [
    {
      "code": 200,
      "message": "iFrame successfully authorized for merchant website."
    }
  ],
  "isSuccessful": true
}
```



## 6.2.1.2. Generate iFrame

---

For details on how to customize and embed an iFrame into your merchant website, see [iFrames](#).

## Endpoint: /api/V2/iFrameConfiguration/GenerateOtcSignupCard

Returns the parameters you need to build an iFrame web address (URL). This endpoint generates the iFrame code snippet dynamically. It does this based on a merchant's specific template and card payments environment. The OTC is a one-time code.

**POST**

<https://pushtopaystaging.crbnj.net/api/V2/iFrameConfiguration/GenerateOtcSignupCard> 

**Request**

Response

### BODY PARAMETERS

**templateId** String **required**

The GUID that identifies the template

**domain** String **required**

The name of a valid top-level internet domain where consumers will use the iFrame. For example, myfintech.com. This domain must appear in the Cross River allowlist.

**firstName** String **optional**

If provided, the firstName will be populated internally and the iFrame will not present a firstName field for completion by the user.

If you provide a firstName you must provide a lastName.

Minimum 2 characters, maximum 35 characters

**lastName** String **optional**

If provided, the lastName will be populated internally and the iFrame will not present a lastName field for completion by the user.

If you provide a lastName you must provide a firstName.

Minimum 2 characters, maximum 35 characters



```
curl
-X POST
--header 'Content-Type: application/json'
--header 'Accept: application/json'
--header 'Authorization: Access token'
-d '{ \
  domain: "https://p2pcusto.com", \
  templateId: "1bc3adff-59af-41c6-6218-08dc990404cd" \
}' 'https://pushtopay.crbccloud.com/api/V2/IFrameConfiguration/GenerateOtc'
```

## Responses

● 200



```
{
  "version": "1.0.0.0",
  "result": {
    "templateId": "1bc3adff-59af-41c6-6218-08dc990404cd",
    "domain": "
https://p2pcusto.com"
  },
  "otcId": "b3595dba-1188-492e-b4d0-e244e210bb29",
  "expiresAt": "2024-07-22T05:40:44.2759814-04:00",
  "createdAt": "2024-07-22T05:40:29.2759814-04:00",
  "warning": null,
  "params": "createdAt=2024-07-22T05:40:29.2759814Z&domain=
https://p2pcusto.com&expiresAt=2024-07-22T05:40:44.2759814Z&otcId=b3595dba
",
  "isSuccessfull": true,
  "isSuccessful": true
}
```

### 6.2.1.3. Add card

---

This API call triggers Account name inquiry (ANI) and Address verification (AVS).  
These are fraud and risk detection services supported by Cross River.

## Endpoint: /api/Card

Signs up a new consumer payment card to the merchant's account. When you call this endpoint, the card payments API stores the card number and returns a token. The API uses the token to identify the card, in place of the actual card number. Tokens provide a security measure to help prevent criminals from accessing card numbers.

**POST**

https://pushtopaystaging.crbnj.net/api/Card



**Request**

Response

### BODY PARAMETERS

**requestId** String **required**

The GUID is a unique identifier that enables the application to link request with response

**firstName** String **required**

Cardholder first name

**lastName** String **required**

Cardholder last name

**sourceSenderId** String **required**

The GUID that identifies the merchant partner

**ownerExternalId** String **optional**

A 4-digit code, assigned by the merchant, to identify the cardholder

**address1** String **optional**

Primary location details of cardholder. For instance, street name, house or building number, and PO box. Maximum 100 characters.

**address2** String optional

Secondary location details of cardholder. For instance, number of apartment or floor. Maximum 100 characters.

**city** String optional

Name of the cardholder's city

**countryCode** String optional

2-letter country code

**state** String optional

2-letter code of the cardholder's state

**zipCode** String optional

- US: 5 digits, A ZIP+4 code with optional space or dash separator (12345, 12345-6789, 12345 6789, 123456789).
- Canada: A valid canadian zip code
- Other: up to 9 chars , letters, numbers, spaces, dashes. Do not contain double spaces or dashes

**phoneNumber** String optional

Optional + at the start, 7-15 characters made of: digits, spaces, dashes, brackets

**email** String optional

A valid cardholder email address, for example, me@mailprovider.com

**creditCardNumber** String **required**

The 15- or 16-digit number found on the front or back of a credit card

**expirationMonth** String required

Card expiration month, 2-digits, numeric

**expirationYear** String required

Card expiration year, 2-digits, numeric

**cvv** String required

The personal customer code on the card. Add the CVV to a function call only when you add a card (POST /api/Card). The API never returns the CVV when you perform a GET or PUT call.



Curl



Node.js



Python



Ruby



Go



```
curl -L 'https://pushtopaystaging.crbnj.net/api/Card' ^  
-H 'Content-Type: application/json' ^  
-H 'Authorization: Bearer eyJhbGciOiJSUzI1NiIsImtpZCI6IjdQREJ4dGVldl9IMXhp  
--data-raw '{  
  "requestId": "f37a0d5a-d62e-46e3-8c20-2842486b55c8",  
  "cardToken": "D116E6A82B704CDFA2942F9B16ED3A65000000",  
  "firstName": "Millie68@hotmail.com",  
  "lastName": "Nitzsche",  
  "ownerExternalId": "411979997",  
  "address1": "1053 Milo Drives",  
  "address2": "",  
  "city": "New Myrltown",  
  "zipCode": "08527",  
  "state": "NJ",  
  "countryCode": "US",  
  "phoneNumber": "779-388-5814",  
  "email": "Akeem.Lesch11@example.net",  
  "creditCardNumber": "5102 5899 9999 9905",  
  "expirationYear": "30",  
  "expirationMonth": "12",  
  "cvv": "909",  
  "cardCompany": "MasterCard"  
'
```

## Responses

● 200



```
{
  "version": "1.0.0.0",
  "result": {
    "requestId": "c2deec7f-6230-42d8-8219-243d8c6209a1",
    "cardToken": "57193C2ghgdh98459E8DC0B26031D72DC4000000",
    "cardFingerprint": "npJNEEghghf3U1Cknd57MqFo582lCxHewI8ptbt7Zum0NQ",
    "firstName": "Millie68@hotmail.com",
    "lastName": "Roob",
    "ownerExternalId": "411979997",
    "address1": "96371 Giles View",
    "address2": "",
    "city": "Okunevaview",
    "zipCode": "08527",
    "isAuthorizationSucceeded": true,
    "state": "NJ",
    "countryCode": "US",
    "phoneNumber": "771-730-2219",
    "email": "Torrey.Donnelly@example.com",
    "expirationYear": "30",
    "expirationMonth": "12",
    "last4Digits": "9905",
    "cardCompany": "MasterCard",
    "isActive": true,
    "deactivatedAt": null,
    "addedOn": "2025-06-30T11:16:44.880045Z",
    "authorizationJson": "{\r\n  \"RequestId\": \"c2deec7f-6230-42d8-8",
    "isFastFundsSupported": true,
    "rail": "McSend",
    "pushEnabled": true,
    "pullEnabled": true,
    "issuerCountryCode": "US",
    "binNumber": "510258",
    "productType": "Unknown"
  },
  "isSuccessfull": true,
  "isSuccessful": true
}
```

## 6.2.1.4. Card list

---



## Endpoint: /api/Card

Returns information about payment cards of consumers signed up to the merchant's account.

GET

https://pushtopaystaging.crbnj.net/api/Card



Request

Response

### QUERY PARAMETERS

**dateAddedFrom** String optional

Date and time from which cards were added (for the beginning of a date range query). Format: YYYY-MM-DDThh:mm:ss, for example, 2022-10-01T01:30:00.

**dateAddedTo** String optional

Date and time up to which cards were added (for the end of a date range query). Format: YYYY-MM-DDThh:mm:ss, for example, 2022-10-01T01:30:00.

**sourceSenderId** String optional

The GUID that identifies the merchant partner

**firstName** String optional

Cardholder first name

**lastName** String optional

Cardholder last name

**isActive** String optional

True if the card is active, otherwise false

**cardCompany**

String

optional

Name of the payment card network:

- Visa
- Mastercard



Curl



Node.js



Python



Ruby



Go



```
curl --location --request  
GET 'https://pushtopaystaging.crbnj.net/api/card/'  
--header 'Accept: application/json'  
--header 'Authorization: Bearer /{xxx}' https://pushtopaystaging.crbnj.net
```

Responses

● 200



```
{
  "version": "1.0.0.0",
  "result": {
    "results": [
      {
        "requestId": "b27232c5-1f18-4d28-a979-57ea3c21839c",
        "cardToken": "1A91B4AF5CF7473C86FB611893795105000000",
        "cardFingerprint": "npJNEEutPf3UlCknd57MqFo582lCxHeWI8ptbt",
        "firstName": "Grayson",
        "lastName": "Schimmel",
        "ownerExternalId": "724742791",
        "address1": "381_Keira_Hill",
        "address2": "",
        "city": "Rogahnborough",
        "zipCode": "08527",
        "isAuthorizationSucceeded": true,
        "state": "NJ",
        "countryCode": "US",
        "phoneNumber": "211-662-5590",
        "email": "Billie.Mayert78@yahoo.com",
        "expirationYear": "26",
        "expirationMonth": "01",
        "last4Digits": "9905",
        "cardCompany": "MasterCard",
        "isActive": true,
        "deactivatedAt": null,
        "addedOn": "2025-06-30T10:57:59.2",
        "authorizationJson": "{\r\n  \"RequestId\": \"59f89722-1d5",
        "isFastFundsSupported": true,
        "rail": "McSend",
        "pushEnabled": true,
        "pullEnabled": true,
        "issuerCountryCode": "US",
        "binNumber": "510258",
        "productType": "Unknown"
      },
      {
        "requestId": "b27232c5-1f18-4d28-a979-57ea3c21839c",
        "cardToken": "57193C2BF898459E8DC0B26031D72DC40000000",
        "cardFingerprint": "npJNEEutPf3UlCknd57MqFo582lCxHeWI8ptbt",
        "firstName": "Millie68@hotmail.com",
        "lastName": "Roob",
        "ownerExternalId": "411979997",
        "address1": "96371 Giles View",
        "address2": "",
```

```
    "city": "Okunevaview",
    "zipCode": "08527",
    "isAuthorizationSucceeded": true,
    "state": "NJ",
    "countryCode": "US",
    "phoneNumber": "771-730-2219",
    "email": "Torrey.Donnelly@example.com",
    "expirationYear": "30",
    "expirationMonth": "12",
    "last4Digits": "9905",
    "cardCompany": "MasterCard",
    "isActive": true,
    "deactivatedAt": null,
    "addedOn": "2025-06-30T11:16:44.88",
    "authorizationJson": "{\\r\\n  \\\"RequestId\\\": \\\"c2deec7f-623",
    "isFastFundsSupported": true,
    "rail": "McSend",
    "pushEnabled": true,
    "pullEnabled": true,
    "issuerCountryCode": "US",
    "binNumber": "510258",
    "productType": "Unknown"
  },
  {
    "requestId": "b27232c5-1f18-4d28-a979-57ea3c21839c",
    "cardToken": "8E5AF0DB8B0740C98792FEC15680632E000000",
    "cardFingerprint": "npJNEEutPf3U1Cknd57MqFo5821CxHewI8ptbt",
    "firstName": "Riley",
    "lastName": "Kshlerin",
    "ownerExternalId": "809811149",
    "address1": "05217_Schuppe_Flats",
    "address2": "",
    "city": "Bergnaumfort",
    "zipCode": "08527",
    "isAuthorizationSucceeded": true,
    "state": "NJ",
    "countryCode": "US",
    "phoneNumber": "723-569-3708",
    "email": "Delpha_Veum59@hotmail.com",
    "expirationYear": "25",
    "expirationMonth": "10",
    "last4Digits": "9905",
    "cardCompany": "MasterCard",
    "isActive": true,
    "deactivatedAt": null,
    "addedOn": "2025-06-23T14:12:04.63",
```

```
    "authorizationJson": "{\r\n  \"RequestId\": \"2c0ad593-03a
    \"isFastFundsSupported\": true,
    \"rail\": \"McSend\",
    \"pushEnabled\": true,
    \"pullEnabled\": true,
    \"issuerCountryCode\": \"US\",
    \"binNumber\": \"510258\",
    \"productType\": \"Unknown\"
  },
  {
    \"requestId\": \"b27232c5-1f18-4d28-a979-57ea3c21839c\",
    \"cardToken\": \"CE957634C7A4467C8E31F8266B43D98C000000\",
    \"cardFingerprint\": \"bmc70G4AmlpF29SIljqVMCaAvy8hHoh3xJAZtc
    \"firstName\": \"Johnson\",
    \"lastName\": \"Beier\",
    \"ownerExternalId\": \"603525412\",
    \"address1\": \"984_Oswald_Way\",
    \"address2\": \"\",
    \"city\": \"Sauerburgh\",
    \"zipCode\": \"10033\",
    \"isAuthorizationSucceeded\": true,
    \"state\": \"NY\",
    \"countryCode\": \"US\",
    \"phoneNumber\": \"218-963-5272\",
    \"email\": \"Nola94@gmail.com\",
    \"expirationYear\": \"26\",
    \"expirationMonth\": \"01\",
    \"last4Digits\": \"5556\",
    \"cardCompany\": \"Visa\",
    \"isActive\": true,
    \"deactivatedAt\": null,
    \"addedOn\": \"2025-06-23T12:40:48.97\",
    \"authorizationJson\": \"{\\r\\n  \\\"RequestId\\\": \\\"255e83bc-a03
    \"isFastFundsSupported\": true,
    \"rail\": \"VisaDirect\",
    \"pushEnabled\": true,
    \"pullEnabled\": true,
    \"issuerCountryCode\": \"US\",
    \"binNumber\": \"400005\",
    \"productType\": \"Unknown\"
  },
  {
    \"requestId\": \"b27232c5-1f18-4d28-a979-57ea3c21839c\",
    \"cardToken\": \"CFEA025BDCB640869914F113F4417A8A000000\",
    \"cardFingerprint\": \"npJNEEutPf3U1Cknd57MqFo5821CxHewI8ptbt
    \"firstName\": \"Ceasar\",
```

```
    "lastName": "Wisozk",
    "ownerExternalId": "449985742",
    "address1": "55584_Gerlach_Forges",
    "address2": "",
    "city": "Lake Violetbury",
    "zipCode": "08527",
    "isAuthorizationSucceeded": true,
    "state": "NJ",
    "countryCode": "US",
    "phoneNumber": "668-622-7591",
    "email": "Matilde_Wyman@yahoo.com",
    "expirationYear": "26",
    "expirationMonth": "04",
    "last4Digits": "9905",
    "cardCompany": "MasterCard",
    "isActive": true,
    "deactivatedAt": null,
    "addedOn": "2025-06-30T09:14:37.29",
    "authorizationJson": "{\r\n  \"RequestId\": \"9beab0d3-8cb",
    "isFastFundsSupported": true,
    "rail": "McSend",
    "pushEnabled": true,
    "pullEnabled": true,
    "issuerCountryCode": "US",
    "binNumber": "510258",
    "productType": "Unknown"
  },
  {
    "requestId": "b27232c5-1f18-4d28-a979-57ea3c21839c",
    "cardToken": "D116E6A82B704CDFA2942F9B16ED3A65000000",
    "cardFingerprint": "npJNEutPf3U1Cknd57MqFo5821CxHeWI8ptbt",
    "firstName": "Haley",
    "lastName": "Schmeler",
    "ownerExternalId": "347704938",
    "address1": "6456_Anderson_Cliffs",
    "address2": "",
    "city": "Hampton",
    "zipCode": "08527",
    "isAuthorizationSucceeded": true,
    "state": "NJ",
    "countryCode": "US",
    "phoneNumber": "462-428-9439",
    "email": "Zetta.Steuber@hotmail.com",
    "expirationYear": "25",
    "expirationMonth": "10",
    "last4Digits": "9905",
```



"cardCompany": "MasterCard".

## 6.2.1.5. Card details

---

## Endpoint: /api/Card/{cardToken}

Returns the parameter values defining a card authorized for transactions with a merchant partner. You get card details using the cardToken attribute.

GET

https://pushtopaystaging.crbnj.net/api/Card/{cardToken}



Request

Response

### BODY PARAMETERS

**cardToken**

String

**required**

Created when initially signing up a card. A randomly generated string of characters that relates to the consumer's payment card (15 to 38 alphanumeric characters). Matches the creditCardToken.



Curl



Node.js



Python



Ruby



Go



```
curl -X GET  
--header 'Accept: application/json'  
--header 'Authorization: Bearer token' 'https://pushtopaystaging.crbnj.net'
```

## Responses

● 200



```
{
  "version": "1.0.0.0",
  "result": {
    "requestId": "2f113def-e511-4340-bb0b-a0b23f40ccf3",
    "cardToken": "06IJMAI0WI4GSPV3C1CP4ZDCRKV4UT1369GN0D",
    "cardFingerprint": "oHMPYVwJBuH2v/HuqIVgEmeMwhpvHqJaopw82bdGUv++Vx",
    "firstName": "CRBFirstNameCMIEE",
    "lastName": "CRBLastNameNGVEU",
    "ownerExternalId": "1",
    "address1": "CRB Street1",
    "address2": "CRB Street2",
    "city": "FortLee",
    "zipCode": "92121",
    "isAuthorizationSucceeded": true,
    "state": null,
    "countryCode": null,
    "phoneNumber": "8586518295",
    "email": "shalom8e15b71b7306410@crbtest.com",
    "expirationYear": "27",
    "expirationMonth": "08",
    "last4Digits": "6129",
    "cardCompany": "MasterCard",
    "isActive": true,
    "deactivatedAt": null,
    "addedOn": "2024-05-15T21:27:59.927",
    "authorizationJson": "{\r\n  \"RequestId\": \"fe5b3675-0048-4140-a",
    "isFastFundsSupported": true,
    "rail": "McSend",
    "pushEnabled": true,
    "pullEnabled": true,
    "issuerCountryCode": "US",
    "binNumber": "222300",
    "productType": "Consumer"
  },
  "isSuccessfull": true,
  "isSuccessful": true
}
```



## 6.2.1.6. Deactivate/reactivate

---

## Endpoint: /api/Card/{cardToken}

Deactivates a consumer's payment card, for instance, if the card is lost or stolen.  
Reactivates the card, for example, if the card is found.

**PUT**

https://pushtopaystaging.crbnj.net/api/Card



**Request**

Response

### BODY PARAMETERS

**requestId** String **required**

The GUID is a unique identifier that enables the application to link request with response

**sourceSenderId** String **optional**

The GUID that identifies the merchant partner

**cardToken** String **required**

Created when initially signing up a card. A randomly generated string of characters that relates to the consumer's payment card (15 to 38 alphanumeric characters).  
Matches the creditCardToken.

**isActive** String **optional**

True if it is active, otherwise false





Curl



Node.js



Python



Ruby



Go



```
package main

import (
    "fmt"
    "strings"
    "net/http"
    "io"
)

func main() {

    url := "https://pushtopaystaging.crbnj.net/api/card"
    method := "PUT"

    payload := strings.NewReader(`{`+"
"+`
    "requestId": "22c2a3c3-6434-46bc-abe8-dc7daf34df7",`+"
"+`
    "cardToken": "[cardToken]",`+"
"+`
    "isActive": false`+"
"+`
    `}`)

    client := &http.Client {
    }
    req, err := http.NewRequest(method, url, payload)

    if err != nil {
        fmt.Println(err)
        return
    }
    req.Header.Add("Content-Type", "application/json")
    req.Header.Add("Authorization", "Bearer eyJhbGciOiJSUzI1NiIsImtpZCI6IjdQ

    res, err := client.Do(req)
    if err != nil {
        fmt.Println(err)
        return
    }
    defer res.Body.Close()

    body, err := io.ReadAll(res.Body)
    if err != nil {
        fmt.Println(err)
    }
}
```

## Responses

● 200



```
{
  "version": "1.0.0.0",
  "result": {
    "requestId": "22c2a3c3-6434-46bc-abeb-dc7daf34df7c",
    "cardToken": "57193C2BF898459E8DC0B26031D72DC4000000",
    "cardFingerprint": "npJNEEutPf3U1Cknd57MqFo582lCxHewI8ptbt7Zum0NQg",
    "firstName": "Millie68@hotmail.com",
    "lastName": "Roob",
    "ownerExternalId": "411979997",
    "address1": "96371 Giles View",
    "address2": "",
    "city": "Okunevaview",
    "zipCode": "08527",
    "isAuthorizationSucceeded": true,
    "state": "NJ",
    "countryCode": "US",
    "phoneNumber": "771-730-2219",
    "email": "Torrey.Donnelly@example.com",
    "expirationYear": "30",
    "expirationMonth": "12",
    "last4Digits": "9905",
    "cardCompany": "MasterCard",
    "isActive": false,
    "deactivatedAt": "2025-06-30T11:37:48.9638193Z",
    "addedOn": "2025-06-30T11:16:44.88",
    "authorizationJson": "{\r\n  \"RequestId\": \"c2deec7f-6230-42d8-8",
    "isFastFundsSupported": true,
    "rail": "McSend",
    "pushEnabled": true,
    "pullEnabled": true,
    "issuerCountryCode": "US",
    "binNumber": "510258",
    "productType": "Unknown"
  },
  "isSuccessfull": true,
  "isSuccessful": true
}
```

## 6.2.2. **Transaction mgmt APIs**

### 6.2.2.1. **Send funds**

---

## Endpoint: /api/transaction

Transactions are classified as domestic or international based on the country of the card issuer. Transactions made with non-US card issuers are considered international. Use this endpoint for both domestic and international transactions.

**POST**

<https://pushtopaystaging.crbnj.net/api/transaction>



**Request**

Response

### BODY PARAMETERS

**requestId** String **required**

The GUID that enables the application to link request with response

**cardToken** String **required**

A randomly generated string of characters that relates to the consumer's payment card (15-38 alphanumeric characters).

The response attribute this matches depends on the payment instrument:

Card. Matches the creditCardToken in the Add a card response.

Wallet. Matches the payAliasToken in the Add a payAlias response.

**amount** Integer **required**

The amount to be transferred, in cents. For example, 10001 = \$100.01.

**sourceSenderName** String **optional**

The name of the merchant. This attribute (alphanumeric) identifies the sender of the funds.

Minimum 1 character.

Maximum 17 characters.

Note: Money transfers must contain the sender's name.

**sourceSenderId** String optional

Money transfers must contain the sender's identifier. This ID identifies the sender of the funds.

For a funds disbursement, non-money transfers must contain either the name of the merchant or the entity sending the disbursement.

**merchantTransactionsIdentifierId** String optional

Identifier of the merchant's transaction

**sourceMcc** String optional

Merchant Category Code (MCC). 4-digits that a credit card issuer uses to categorize consumer transactions for their card.

**purposeOfPayment** String optional

A 1-12 character code that indicates the reason for a transaction. This code is required for certain markets (countries).

**programId** String optional

Populate this attribute when you want to connect a transaction to a given sub-merchant network program ID (Visa: MVV, Mastercard: MCID).

Otherwise, your merchant-level program ID is added automatically. You don't need to enter it.

**sourceAddress ▶** Object optional

Use this field to append online transaction data to the statement descriptor, for example to use a sub-merchant address instead of the merchant address

**correspondingEntity ▶**

Object

optional

The name and address details of the sender of a push transaction or receiver of a pull transaction.

IMPORTANT: This attribute is required in transactions that are account-to-account (A2A) or person-to-person (P2P). These are transactions using business type of AA, BI, CD, CP, FT, PP, TU, WT

**businessType**

String

optional

\* One of: BB, FD, AA, PP, BI, CD, FT, WT, CP, TU, CI, CO, GD, GP, LO, MD, MP, OG, PD



Curl



Node.js



Python



Ruby



Go



```
curl -L 'https://pushtopaystaging.crbnj.net/api/transaction' ^  
-H 'Content-Type: application/json' ^  
-H 'Authorization: Bearer token' ^  
-d '{  
  "requestId": "563288b0-ccf4-4d4c-978b-d87ba30889d3",  
  "cardToken": "A5D4B6AA33A64F718A49B18FE0CDC6AF000000",  
  "amount": 200,  
  "sourceSenderName": "My Company"  
}'
```



## Responses

● 200



```
{
  "version": "1.0.0.0",
  "result": {
    "transactionRequestId": "cf97a1ed-eb43-4e8c-aa3b-6f877f182529",
    "amount": 200,
    "transactionRequestedAt": "2025-07-02T07:06:45.3755054Z",
    "transactionStatus": "Succeeded",
    "errorDescription": null,
    "creditCardId": "A5D4B6AA33A64F718A49B18FE0CDC6AF000000",
    "railId": "VisaDirect",
    "network": "Visa",
    "retrievalReferenceId": "518307414003",
    "actualTransactionDoneAt": "2025-07-02T07:06:45.6601706Z",
    "paymentSent": true,
    "requestApproved": true,
    "responseReceived": true,
    "responseCode": "00",
    "responseDescription": "Approved",
    "traceNumber": "414003",
    "error": null,
    "sourceSenderName": "My Company",
    "sourceMcc": null,
    "programId": null,
    "businessType": "FD"
  },
  "isSuccessfull": true,
  "isSuccessful": true
}
```

- Business types

## 6.2.2.2. Request funds

---

## Endpoint: /api/PullTransaction/

Pulls funds from a registered card.

You also use this call for Account Funding Transactions (AFT), which pull funds from a specific account for use in a non-merchant account. For example, use this to load a prepaid card, top up a wallet, or fund a person-to-person (P2P) money transfer.

The API response includes the card token, amount transferred, and a transaction status that indicates success or failure.

**POST**

<https://pullfromcardapistg.crbnj.net/api/PullTransaction/>



**Request**

Response

### BODY PARAMETERS

**merchantTransactionsIdentifierId** String optional

Identifier of the merchant's transaction

**requesterId** String optional

The merchant partner unique identifier. 32 hexadecimal characters.

**requesterName** String optional

The name of the merchant

**requesterMcc** String optional

Merchant Category Code (MCC). 4-digits that a credit card issuer uses to categorize consumer transactions for their card.

**cvv** String optional

Personal customer code on the card. Note: Add the CVV to a function call only when you add a card (POST /api/Card). The API never returns CVV when you perform a GET or PUT call.

**acceptPartialAmount** Boolean optional

This field is sent only to participating issuers.

Enter a value of TRUE if partially approved transactions can be accepted.

FALSE. Partial authorization not supported (default)

TRUE. Partial authorization supported

**requestId** String required

The GUID that enables the application to link request with response

**cardToken** String required

A randomly generated string of characters that relates to the consumer's payment card (15-38 alphanumeric characters). Matches the creditCardToken in the response.

**amount** Integer required

The amount to be transferred, in cents. For example, 10001 = \$100.01.

**programId** Integer optional

Populate this attribute when you want to connect a transaction to a given sub-merchant network program ID (Visa: MVV, Mastercard: MCID).

Otherwise, your merchant-level program ID is added automatically. You don't need to enter it.

**sourceAddress** ▶ Object optional

Use this field to append online transaction data to the statement descriptor, for example to use a sub-merchant address instead of the merchant address

**correspondingEntity ▶**

Object

optional

The name and address details of the sender of a push transaction or receiver of a pull transaction.

IMPORTANT: This attribute is required in transactions that are account-to-account (A2A) or person-to-person (P2P).

**businessType**

String

optional

\* One of: BB, FD, AA, PP, BI, CD, FT, WT, CP



Curl



Node.js



Python



Ruby



Go



```
curl -L 'https://pullfromcardapistg.crbnj.net/api/pulltransaction' ^  
-H 'Content-Type: application/json' ^  
-H 'Authorization: Bearer token' ^  
-d '{  
  "requestId": "f6ef8097-7602-451a-8b37-e568e2d628ae",  
  "cardToken": "A5D4B6AA33A64F718A49B18FE0CDC6AF000000",  
  "amount": 200,  
  "sourceSenderName": "My Company"  
}'
```

## Responses

● 200



```
{
  "version": "1.0.0.0",
  "result": {
    "transactionRequestId": "174d1bfa-be7e-4bdc-b74e-983118b84a36",
    "amount": 200,
    "originalAmount": 200,
    "transactionRequestedAt": "2025-07-02T07:22:27.881564Z",
    "transactionStatus": "Succeeded",
    "errorDescription": null,
    "creditCardId": "A5D4B6AA33A64F718A49B18FE0CDC6AF000000",
    "railId": "VisaDirect",
    "network": "Visa",
    "retrievalReferenceId": "518307216716",
    "actualTransactionDoneAt": "2025-07-02T07:22:27.9133011Z",
    "requestApproved": true,
    "responseReceived": true,
    "responseCode": "00",
    "responseDescription": "Approved",
    "traceNumber": "216716",
    "error": null,
    "requesterName": null,
    "requesterMcc": null,
    "programId": null,
    "businessType": "FD"
  },
  "isSuccessful": true,
  "isSuccessfull": true
}
```

- Business types

### 6.2.2.3. Push transactions

---



## Endpoint: /api/transaction

Returns a list of transactions, and their details, for payments sent from a merchant to a payee.

GET

<https://pushtopaystaging.crbnj.net/api/transaction>



Request

Response

### QUERY PARAMETERS

**cardToken** String optional

A randomly generated string of characters that relates to the consumer's payment card (15-38 alphanumeric characters). Matches the creditCardToken in the response.

**statusEnum** String optional

Transaction status:

- Pending
- Succeeded
- Failed
- Rejected

**sourceSenderId** String optional

The GUID that identifies the merchant partner

**fromDate** String optional

Report start date.

**toDate** String optional

Report end date.

```
curl -X
GET
--header 'Accept: application/json'
--header 'Authorization: Bearer {xxx}'
'https://pushtopaystaging.crbnj.net/api/transaction'
```

## Responses

● 200



```
{
  "transactionRequestId": "ji7abzpy-8xp3-gpby-2604-jab7tpxy36tp",
  "amount": 400,
  "transactionRequestedAt": "2021-08-03T21:09:13.5882004Z",
  "transactionStatus": "Succeeded",
  "errorDescription": null,
  "creditCardId": "KR7D47MNJIL2R390ZNRVPC0DIS82ES8JE6J7VU",
  "railId": "TabaPay",
  "network": "MasterCard",
  "retrievalReferenceId": "8abcb1fa-51f1-4b2d-9998-5948877bdcc0",
  "actualTransactionDoneAt": "2021-08-03T21:09:13.937108Z",
  "requestApproved": true,
  "responseReceived": true,
  "responseCode": "00",
  "responseDescription": "Approved",
  "traceNumber": "hl7a9zpg-8zpy-92po-bhp2-xhpg5804pz2k",
  "error": null,
  "requesterName": null,
  "requesterMcc": null
  "isSuccessful": true
}
```

## 6.2.2.4. Pull transactions

---

## Endpoint: /api/PullTransaction

Returns a list of transactions where a merchant (the payee) takes funds from an end-consumer (the payer).

GET

<https://pullfromcardapistg.crbnj.net/api/PullTransaction>



Request

Response

### QUERY PARAMETERS

**cardToken** String optional

A randomly generated string of characters that relates to the consumer's payment card (15-38 alphanumeric characters). Matches the creditCardToken in the response.

**statusEnum** String optional

Transaction status:

- Pending
- Succeeded
- Failed
- Rejected

**requesterId** String optional

The merchant partner unique identifier. 32 hexadecimal characters.

**fromDate** String optional

Date from which transactions were added (for beginning of a date range query). Format: YYYY-MM-DDThh:mm:ss.<UTC offset>Z, for instance 2022-12-19T18:51:21.22Z

**toDate** String optional

Date up to which transactions were added (for end of a date range query). Format: YYYY-MM-DDThh:mm:ss.<UTC offset>Z, for instance 2022-12-19T18:51:21.22Z.

```
curl -X GET
--header 'Accept: application/json'
--header 'Authorization: Bearer {xxx}'
'https://pullfromcardapistg.crbnj.net/api/PullTransaction'
```

## Responses

● 200



```
{
  "transactionRequestId": "ji7abzpy-8xp3-gpby-2604-jab7tpxy36tp",
  "amount": 400,
  "transactionRequestedAt": "2021-08-03T21:09:13.5882004Z",
  "transactionStatus": "Succeeded",
  "errorDescription": null,
  "creditCardId": "KR7D47MNJIL2R390ZNRVPC0DIS82ES8JE6J7VU",
  "railId": "Tabapay",
  "network": "MasterCard",
  "retrievalReferenceId": "8abcb1fa-51f1-4b2d-9998-5948877bdcc0",
  "actualTransactionDoneAt": "2021-08-03T21:09:13.937108Z",
  "requestApproved": true,
  "responseReceived": true,
  "responseCode": "00",
  "responseDescription": "Approved",
  "traceNumber": "h17a9zpg-8zpy-92po-bhp2-xhpg5804pz2k",
  "error": null,
  "requesterName": null,
  "requesterMcc": null
  "isSuccessful": true
}
```

#### 6.2.2.5. Push transaction by ID

---

## Endpoint: /api/transaction/{transactionId}

Returns details of a specific transaction where a merchant (the payer) sends funds to an end-consumer (the payee). Use the transactionId attribute as the transaction identifier.

GET

<https://pushtopaystaging.crbnj.net/api/transaction/{transactionId}> 

Request

Response

### PATH PARAMS

**transactionId** String **required**

A unique string (32 hexadecimal characters) that identifies the transaction in a definite way. The payment card network passes this identifier, without changing it, throughout the whole interbank chain.

### QUERY PARAMETERS

**sourceSenderId** String **optional**

The GUID that identifies the merchant partner

```
curl -X
GET
--header 'Accept: application/json'
--header 'Authorization: Bearer token'
'https://pushtopaystaging.crbnj.net/api/transaction/230bc14b-970b-4ee2-8fd'
```

## Responses

● 200



```
{
  "transactionRequestId": "ji7abzpy-8xp3-gpby-2604-jab7tpxy36tp",
  "amount": 400,
  "transactionRequestedAt": "2021-08-03T21:09:13.5882004Z",
  "transactionStatus": "Succeeded",
  "errorDescription": null,
  "creditCardId": "KR7D47MNJIL2R390ZNRVPC0DIS82ES8JE6J7VU",
  "railId": "TabaPay",
  "network": "MasterCard",
  "retrievalReferenceId": "8abcb1fa-51f1-4b2d-9998-5948877bdcc0",
  "actualTransactionDoneAt": "2021-08-03T21:09:13.937108Z",
  "requestApproved": true,
  "responseReceived": true,
  "responseCode": "00",
  "responseDescription": "Approved",
  "traceNumber": "hl7a9zpg-8zpy-92po-bhp2-xhpg5804pz2k",
  "error": null,
  "requesterName": null,
  "requesterMcc": null
  "isSuccessful": true
}
```



## 6.2.2.6. Pull transaction by ID

---

## Endpoint: /api/PullTransaction/{transactionId}

Returns details of a specific transaction where a merchant (the payee) takes funds from an end-consumer (the payer). Use the transactionId attribute as the transaction identifier.

GET

<https://pullfromcardapistg.crbnj.net/api/PullTransaction/{transactionId}> 

Request

Response

### PATH PARAMS

**transactionId** String **required**

A unique string (32 hexadecimal characters) that identifies the transaction in a definite way. The payment card network passes this identifier, without changing it, throughout the whole interbank chain.

### QUERY PARAMETERS

**requesterId** String **optional**

The merchant partner unique identifier. 32 hexadecimal characters.

```
curl -X
GET
--header 'Accept: application/json'
--header 'Authorization: Bearer token'
'https://pullfromcardapistg.crbnj.net/api/transaction/230bc14b-970b-4ee2-8
```

## Responses

● 200



```
{
  "transactionRequestId": "ji7abzpy-8xp3-gpby-2604-jab7tpxy36tp",
  "amount": 400,
  "transactionRequestedAt": "2021-08-03T21:09:13.5882004Z",
  "transactionStatus": "Succeeded",
  "errorDescription": null,
  "creditCardId": "KR7D47MNJIL2R390ZNRVPC0DIS82ES8JE6J7VU",
  "railId": "TabaPay",
  "network": "MasterCard",
  "retrievalReferenceId": "8abcb1fa-51f1-4b2d-9998-5948877bdcc0",
  "actualTransactionDoneAt": "2021-08-03T21:09:13.937108Z",
  "requestApproved": true,
  "responseReceived": true,
  "responseCode": "00",
  "responseDescription": "Approved",
  "traceNumber": "hl7a9zpg-8zpy-92po-bhp2-xhpg5804pz2k",
  "error": null,
  "requesterName": null,
  "requesterMcc": null
  "isSuccessful": true
}
```

## 6.2.3. Foreign exchange APIs

### 6.2.3.1. Exchange rates

---

International transactions concepts

## Endpoint: /api/FXRates

Returns up-to-date foreign exchange rate information for card-based payments. As a Cross River merchant partner, you do this before making an international transaction.

GET

<https://pushtopaystaging.crbnj.net/api/FXRates>



Request

Response

### QUERY PARAMETERS

**requestId** String optional

The GUID that enables the application to link request with response

**sourceCurrency** String optional

The currency code for the source (monetary) amount. This is a numeric value based on the currencies listed in ISO-4217. For example, if converting from (USD) 840 to (CAD) 124, the value would be 840.

**sourceAmount**

Number

optional

The amount of money the payer is sending, in the currency indicated in sourceCurrency.

This value is needed for source-to-destination lookup.

Minor currency units after the decimal point must be  $\leq$  (less than or equal to) the defined currency exponent (the digits following the decimal).

This field cannot be 0 or null.

The field contains between 1 – 13 characters, including a decimal, for example, 100.55.

For example:

Two (2) digits indicated after the decimal point for a two-exponent currency (99.85, 99.00, etc.).

Three (3) digits indicated after the decimal point for a three-exponent currency (989.333, 989.340, 989.000, etc.).

No decimal point or minor units indicated for a zero (0) exponent currency (95, 100, etc.).

**destinationCurrency**

String

required

The currency code for the destination (monetary) amount. A numeric value based on the currencies listed in ISO-4217. For example, if converting from (USD) 840 to (CAD) 124, the value would be 124.

```
{
  "requestId": "3fa85f64-5717-4562-b3fc-2c963f66afa6",
  "sourceCurrency": "840",
  "sourceAmount": 100,
  "destinationCurrency": "826",
  "markupPercentage": 0
}
```

## Responses

● 200

```
{
  "requestId": "3fa85f64-5717-4562-b3fc-2c963f66afa6",
  "markupRateApplied": 0.03,
  "fxRateNetworkResponses": [
    {
      "network": visa,
      "conversionRate": 1.21,
      "destinationAmountWithMarkup": 103,
      "destinationAmountWithoutMarkup": 100
    },
    {
      "network": mastercard,
      "conversionRate": 1.21,
      "destinationAmountWithMarkup": 103,
      "destinationAmountWithoutMarkup": 100
    }
  ]
}
```

## 6.2.4. OFAC APIs

### 6.2.4.1. OFAC compliance

---

#### **IMPORTANT**

Every international transaction requires a new *OFAC key*.



## Endpoint: /api/OfacScan

Checks OFAC compliance. Cross River requires OFAC screening before sending an international money transfer. This API endpoint makes this check.

**POST**

https://pushtopaystaging.crbnj.net/api/OfacScan



**Request**

Response

### BODY PARAMETERS

**requestId** String optional

The GUID that enables the application to link request with response.

**sourceSenderId** String optional

The GUID that identifies the merchant partner.

**cardToken** String **required**

Created when initially signing up a card. A randomly generated string of characters that relates to the consumer's payment card (15 to 38 alphanumeric characters).  
Matches the creditCardToken.



Curl



Node.js



Python



Ruby



Go



```
curl -L 'https://pushtopaystaging.crbnj.net/api//OfacScan' ^  
-H 'Content-Type: application/json' ^  
-H 'Authorization: Bearer token' ^  
-d '{  
  "requestId": "{{requestId}}",  
  "cardToken": "DC57D6A268374505BC7EA37A461BB69D000000"  
}'
```

## Responses

● 200



```
{  
  "requestId": "3fa85f85-5217-4562-b3fc-2c963f66afa4",  
  "creditCardToken": "xh7alh99sxn7b5v",  
  "error": "",  
  "requestedAt": "2023-04-27T10:25:53.685Z",  
  "respondedAt": "2023-04-27T10:25:53.885Z",  
  "updatedAt": "2023-04-27T10:27:25.555Z",  
  "status": "Hit",  
}
```

## 6.2.4.2. OFAC status

---

## Endpoint: /api/OfacScan/{scanId}/{sourceSenderId}

Returns details showing compliance with international transaction screening requirements. The API makes the request based on the scanId and the sourceSenderId (partner) identifier.

GET

<https://pushtopaystaging.crbnj.net/api/OfacScan/{scanId}/{sourceSenderId}> 

**Request**

Response

### PATH PARAMS

**scanId**

String

**required**

Matches the RequestId found in the response. Use this GUID to identify this scan at a later time.

Note: This is the RequestId you sent in a specific OfacScan call.

**sourceSenderId**

String

**required**

The GUID that identifies the merchant partner



Curl



Node.js



Python



Ruby



Go



```
curl -L -X GET 'https://pushtopaystaging.crbnj.net/api/OfacScan/5c90e10f-e
-H 'Content-Type: application/json' ^
-H 'Authorization: Bearer token' ^
-d '{
  "requestId": "57c39d38-f780-4917-babe-ff67fb44b882",
  "cardToken": "DC57D6A268374505BC7EA37A461BB69D000000"
}'
```

## Responses

● 200



```
{
  "requestId": "3fa85f85-5217-4562-b3fc-2c963f66afa4",
  "creditCardToken": "xh7alh99sxn7b5v",
  "error": "",
  "requestedAt": "2023-04-27T10:25:53.685Z",
  "respondedAt": "2023-04-27T10:25:53.885Z",
  "updatedAt": "2023-04-27T10:27:25.555Z",
  "status": "Hit",
}
```

#### 6.2.4.3. OFAC scan ID

---

## Endpoint: /api/OfacScan/{scanId}

Returns details about a particular compliance scan, based on a specific request identifier.

GET

https://pushtopaystaging.crbnj.net/api/OfacScan/{scanId}



Request

Response

### PATH PARAMS

**scanId** String **required**

Matches the RequestId found in the response. Use this GUID to identify this scan at a later time.

Note: This is the RequestId you sent in a specific OfacScan call.

### QUERY PARAMETERS

**sourceSenderId** String **optional**

The GUID that identifies the merchant partner



Curl



Node.js



Python



Ruby



Go



```
{
  "requestId": "3fa85f85-5217-4562-b3fc-2c963f66afa4",
  "sourceSenderId": "298fa967-97dp-vvy6-afag-11sghvk6nx12",
  "cardToken": "xh7a1h99sxn7b5v"
}
```

## Responses

● 200



```
{
  "requestId": "3fa85f85-5217-4562-b3fc-2c963f66afa4",
  "creditCardToken": "xh7a1h99sxn7b5v",
  "error": "",
  "requestedAt": "2023-04-27T10:25:53.685Z",
  "respondedAt": "2023-04-27T10:25:53.885Z",
  "updatedAt": "2023-04-27T10:27:25.555Z",
  "status": "Hit",
}
```



## 6.2.5. Dynamic descriptor

---

If you're configured for dynamic statement descriptors, you first choose whether to include the abbreviated name or not. Like with static descriptors, this text always appears for every transaction and does not change. In effect, the dynamic descriptor is in addition to the static descriptor.

You add additional text using a specific field in the transaction API POST request. The descriptor will look like this:

- `MerchantName*AdditionalText`
- `AbbreviatedName*MerchantName*AdditionalText`

To configure the dynamic field:

- **Send funds to payee (push)**  
Use `sourceSenderName` in the [POST/api/Transaction](#) call
- **Request funds from payer (pull)**  
Use `requesterName` in the [POST/api/pullTransaction](#) call.

For example, if the merchant name is **Cross River**, the abbreviated name is **CRBT**, and your additional text is **ExpressPayment**, the descriptor would look like one of these:

- `CrossRiver*ExpressPayment`
- `CRBT*CrossRiver ExpressPayment`

Some processors append city and state to the descriptor:

- `CrossRiver*ExpressPayment NewJersey, NJ`
- `CRBT*CrossRiver ExpressPayment NewJersey, NJ`

For information about the static statement descriptor refer to [Statement descriptor](#)

## 6.2.6. iFrames

---

### Introduction

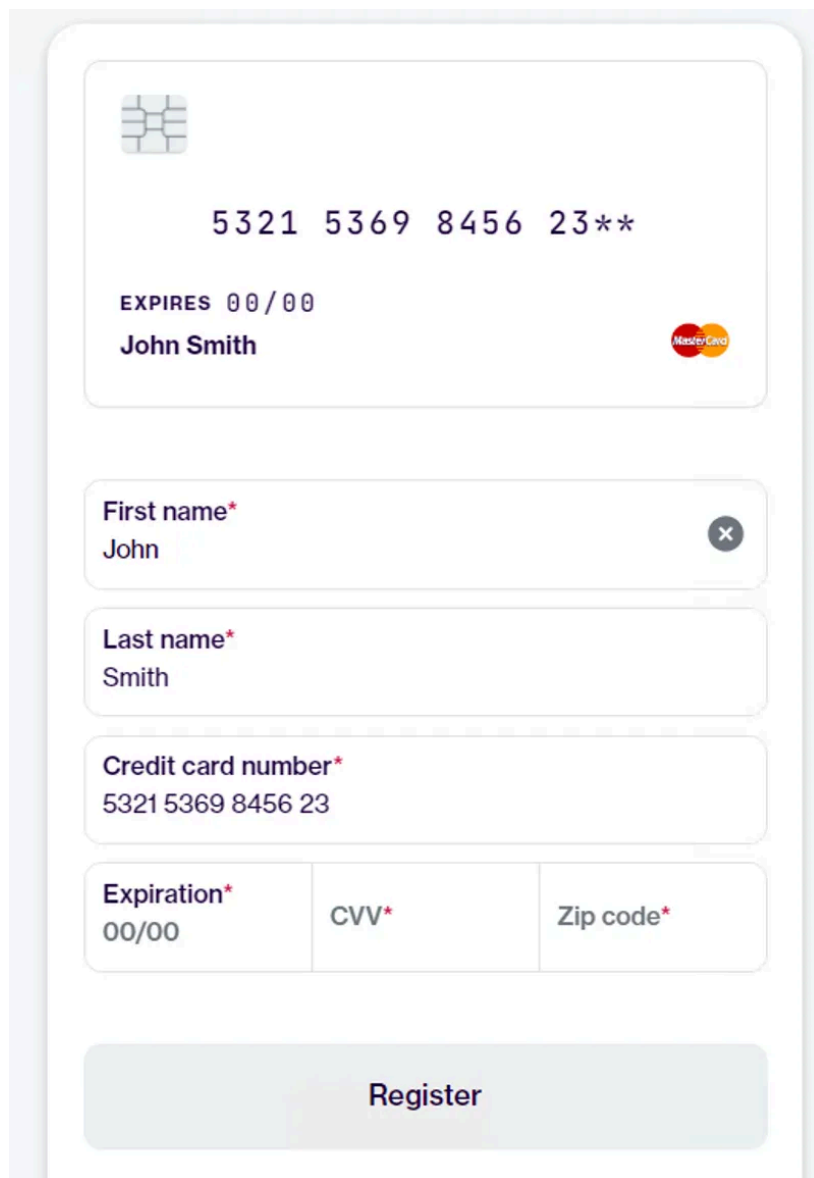
An iFrame makes embedding payment functionality into websites convenient and straightforward. Through secure handling of payment information, iFrames ensure PCI compliance. Card details that a customer enters go directly into the iFrame. Cross River sends merchants a token that represents those details. Then, merchants can use that token to request a card-based transaction.

#### **IMPORTANT**

In Cross River's Card Payments API, iFrames are compliant with the PCI DSS standard. PCI-compliant iFrames lower the cost of PCI compliance by merchants. That's because credentials are already saved onto a Cross River server. This means those credentials don't have to be stored by CR partners.

### How to use iFrames

Do the configuration steps (described in Generating an iFrame, below). Then, integrate the iFrame into the look and feel of your website.

A sample iFrame image of a credit card registration form. The form is white with rounded corners and a light gray border. At the top, there is a card preview section containing a chip icon, the card number '5321 5369 8456 23\*\*', the expiration date 'EXPIRES 00/00', the cardholder's name 'John Smith', and the Mastercard logo. Below this, there are four input fields: 'First name\*' with the value 'John', 'Last name\*' with the value 'Smith', 'Credit card number\*' with the value '5321 5369 8456 23', and a three-part field for 'Expiration\*' (00/00), 'CVV\*', and 'Zip code\*'. At the bottom is a large, light blue 'Register' button.

Sample iFrame image

## Generating an iFrame

To set up an iFrame and get it working on your merchant site, use the iFrame Generator and the [Generate iFrame](#) API.

### iFrame generator overview

<b>Overview</b>	<p>Welcome to the iFrame Generator, your self-service interface for creating PCI-compliant iFrames. iFrame Generator provides an efficient and streamlined way to embed iFrames onto your merchant website.</p> <p>With iFrame Generator, creating an iFrame is a one-time action. Once you have generated an iFrame, you can immediately begin entering customer card information via your website. Of course, if you wish, you may add more iFrames in the future.</p>
<b>Use case (high-level)</b>	<p>Do the following to create and embed an iFrame onto your site:</p> <ol style="list-style-type: none"><li>1. Create an iFrame design template.</li><li>2. Customize the template based on the requirements of your merchant website.</li><li>3. Save the template.</li><li>4. Download the iFrame code.</li><li>5. Embed the iFrame code into your website.</li></ol>

## Use the iFrame Generator

The procedure for generating an iFrame contains 3 main steps:

1. Add a template.
2. Define a style.
3. Build a URL.

### 1 Add a template

1. Launch the **Customer Portal UI**.
2. In the left-side icon bar, click **iFrame**.
3. In the iFrames window, click **+ Add template**.



## 2 Define a style

1. Assign the template a name and domain. Define registration and error paths.
2. You can use the **Credit card design**, and **Form design** sections to customize your iFrame look and feel.
3. Click **Save template** when you're done.

COS Payments

Dashboard

Profile

Transactions

Cards

iFrame

Users

Sign out

Save template

Copy embedded code

Template name\*

Example1

Domain (where your iframe will live)\*

https://www.crbexample.com

Uri to redirect on error\*

/error

Uri to redirect on successful registration\*

/success1

Credit card design

Card background color

#e9e3e3

Card text color

#1f0040

Company logo url

Show credit card

Form design

Background color

#ffffff

Primary text color

#1f0040

Secondary text color

#6c7871

Button background color

#e9e3e3

Preview

4580 \*\*\*\* \* \* \* \*

EXPIRES 00/00

Card Holder

VISA

First name\*

Last name\*

Credit card number\*

4580

Expiration\*

00/00

CVV\*

Zip code\*

Register

3

## Build a URL

### 1. Click **Copy embedded code**.

A detailed pop-up tells you how to:

- Build the iFrame URL using the [OTC Card Signup API](#).
- Insert the API-generated parameters into the iFrame source code.
- Paste the iFrame code into your merchant site HTML.

### 2. Click **OK, got it**.

Now your iFrame is ready to sign up cards.

1. Use the following api together with the idToken to get the params to build the iframe url.

```
Url: https://p2pcustomerportalsandbox.crbcos.com/api/Iframes/IframeOtc
Method: POST
Body: {
  domain: "https://www.theclientgoogle.com",
  templateId: "92f9c06a-84c6-4c62-ab69-08dbe8fdbff0"
}
```

2. Insert the params into the following iframe src:

```
<iframe src="https://p2pcreditcardiframesandbox.crbcos.com/?{response.result.params}" width="100%" height="100%" frameborder="0" scrolling="no"></iframe>
```

3. Paste the iframe code into the html of your website.

**Note:** The URL will expire in 5 minutes. To enhance the users experience repeat the steps to generate a new OTC before the time expires.

OK, Got It

# 6.2.7. Webhook events

When you work with Cross River accounts, cards and payments, you register for specific webhooks which have a specific format.

This table presents Card Payments Webhook events that are described in detail below.

Event Name	Description
<u>CardAuthorized</u>	A card was authorized
<u>CardStatusChanged</u>	Card status changed from <span>OldStatus</span> to <span>NewStatus</span> Example: active to inactive
<u>TransactionCompleted</u>	A transaction was completed
<u>TransactionStatusChanged</u>	Transaction status changed from <span>OldTransactionStatus</span> to <span>NewTransactionStatus</span> Example: succeeded to pending
<u>AccountCreditReceived</u>	The COS account of the merchant is credited with the amount mentioned on the payload at the timestamp mentioned on the payload.

## Event examples

### CardAuthorized





Sample Event



```
{
  "EventId": "8e153f2e-5876-46aa-32a4-b2e1008a7f53",
  "EventName": "CardAuthorized",
  "Data": {
    "RequestId": "27be8469-d8e6-4186-aa61-57d828ae14d6",
    "CardToken": "516093D8F4C1AD7882F01AC73097E864000000",
    "CardFingerprint": "ZWzLZaWrLj0qK4XNvZZYPGX186xZPp4VXEsUikdi8HNo+s4jg20A7",
    "Authorized": true,
    "SourceSenderId": "5173a0f8-0931-441e-ac27-ae14412336c3",
    "ResponseReceived": true,
    "ResponseCode": "OK",
    "ResponseDescription": "OK",
    "Error": null,
    "AddressVerified": true,
    "Rail": "VisaDirect",
    "CustomerReferenceNumber": null,
    "PushEnabled": true,
    "PullEnabled": true
  },
  "ClientMessageBase64Encoded": null,
  "RequestId": "27be8469-d8e6-4186-aa61-57d828ae14d6"
}
```

## CardStatusChaged



Sample Event



```
{
  "EventId": "a41c20cb-2cc9-a257-8368-bae420d88a45",
  "EventName": "CardStatusChanged",
  "Data": {
    "StatusChangedAt": "2025-05-21T17:08:23.9055016Z",
    "CreditCardId": "A815CE3DBF414961AA11BC4E4C351193000000",
    "OldStatus": "Active",
    "NewStatus": "Inactive"
  },
  "ClientMessageBase64Encoded": null,
  "RequestId": "1e79e774-d218-4850-8355-fe25588c6dbd"
}
```

# TransactionCompleted



Sample Event



```
{
  "EventId": "92175b24-f01e-4fab-91d4-baf400da765e",
  "EventName": "TransactionCompleted",
  "Data": {
    "TransactionId": "ab82a520-bd1f-ed55-9b9e-5140321e7215",
    "Amount": 10000,
    "SourceSenderId": "7813a0f8-0937-441e-5557-ae19412306c3",
    "TransactionRequestedAt": "2025-05-21T17:15:23.3539347Z",
    "TransactionStatus": "Succeeded",
    "ErrorDescription": null,
    "CreditCardId": "42ABEE0CEB7B3314BC087366929541F2000000",
    "Rail": "VisaDirect",
    "TransactionType": "PullTransaction",
    "RequesterId": null
  },
  "ClientMessageBase64Encoded": null,
  "RequestId": "ab82a520-bd1f-ed55-9b9e-5140321e7215"
}
```

# TransactionStatusChanged



Sample Event



```
{
  "EventId": "5d10113d-74d4-4835-be46-ae2300c03745",
  "EventName": "TransactionStatusChanged",
  "Data": {
    "TransactionId": "3c7bf1aa-af58-4f71-8345-43f39d8f3378",
    "Amount": 20,
    "TransactionRequestedAt": "2022-01-13T14:34:29.0304876",
    "OldTransactionStatus": "Failed",
    "NewTransactionStatus": "Succeeded",
    "StatusChangedAt": "2022-01-20T16:39:14.3167755Z",
    "CreditCardId": "R00XWSJJIV9AUKWS3IZ8LIKRVSZLYWWGMDI20F",
    "Rail": "TabaPay"
  },
  "ClientMessageBase64Encoded": null,
  "RequestId": "3c7bf1aa-af58-4f71-8345-43f39d8f3378"
}
```

## AccountCreditReceived



Sample Event



```
{
  "EventId": "880a1390-b516-4801-bb41-88e400d91c0b",
  "EventName": "AccountCreditReceived",
  "Data": {
    "SubMerchantId": "3373a0f8-0937-467e-ac27-ae19772306c3",
    "AccountNumber": "2726411651",
    "CreditInCents": 72500,
    "CreditTimestampInUtc": "2025-05-21T17:10:28.2938885Z"
  },
  "ClientMessageBase64Encoded": null,
  "RequestId": null
}
```

## 6.2.8. Request and response codes

### Network Identifications

	Network name	EST Cutoff time
1	<b>Domestic</b>	
2	Accel/Exchange	15:59:59
3	Interlink	5:59:59
4	ET	00:59:59
5	Mastercard/ Maestro	17:29:59
6	MoneySend	01:59:59
7	NYCE	15:29:59
8	PULSE	00:59:59
9	RPPS	16:59:59
10	STAR	17:59:59
11	Visa/Plus	5:59:59
12	<b>International</b>	
13	VisaIntl	
14	IntlMasterCard	

### Purpose of payment (PoP) codes

International transactions in the following countries require `purposeOfPayment` values in the transaction request:

- Argentina
- Bangladesh
- Chile

- Columbia
- Egypt
- India
- Mexico

Code	Description
ISACCT	Account management
ISAIRB	Air transport related business
ISALLW	Transaction is the payment of allowance
ISANNI	Settlement of annuity
ISBENE	Unemployment disability benefit
ISBEXP	Business expenses
ISBONU	Bonus payment
ISBUSB	Bus transport related business
ISCASH	Cash management transfer
ISCBTV	Payment of cable TV bill
ISCCHD	Government institute issued related to cash compensation, helplessness, and disability
ISCCRD	Credit card payment
ISCDBL	Payment of credit card bill
ISCHAR	Payment for charity reasons
ISCOLL	Collection payment
ISCOMC	Commercial payment
ISCOMM	Commission
ISCOMP	Compensation relating to interest loss / value date adjustment and can include fees
ISCPYR	Payment of copyright
ISDCRD	Related to a debit card payment
ISDEPT	Payment of a deposit
ISDIVD	Payment of dividend
ISEDUC	Payment of study / tuition fees

Code	Description
ISELEC	Payment of electricity bill
ISENRG	Energies
ISFEES	General fees
ISFERB	Payment for ferry related business
ISFREX	Foreign exchange
ISGASB	Payment of gas bill
ISGFRP	Compensation to unemployed persons during insolvency procedures
ISGOVT	Government payment
ISHLTI	Health insurance
ISICCP	Reimbursement of credit card payment
ISIDCP	Reimbursement of debit card payment
ISINPC	Payment of car insurance premium
ISINSC	Transaction is related to the payment of an insurance claim
ISINSM	Installment
ISINSU	Insurance premium
ISINVS	Payment of mutual funds, investment products and shares
ISINTC	Intra-company payment
ISINTE	Interest
ISINTX	Income tax
ISINVS	Investment
ISLBRI	Labor insurance
ISLICF	License fee
ISLIFI	Life insurance
ISLOAN	Loan
ISMDCS	Medical services

Code	Description
ISMP2B	Mobile P2B payment
ISMP2P	Mobile P2P payment
ISMTUP	Mobile top up
ISNOWS	Not otherwise specified
ISOTHR	Other
ISOTLC	Transaction is related to a payment of another telecom-related bill
ISPAYR	Payroll
ISPEFC	Contribution to pension fund
ISPENS	Pension payment
ISPHON	Payment of telephone bill
ISPPTI	Property insurance
ISRELG	Transaction is for general rental / lease
ISPENS	The payment of rent
ISRLWY	Payment for railway transport related business
ISROYA	Royalties
ISSALA	Salary payment
ISSAVG	Payment to savings / retirement account
ISSECU	Securities
ISSSBE	Social security benefit
ISSTDY	Study
ISSUBS	Subscription
ISSUPP	Supplier payment
ISTAXR	Refund of a tax payment or obligation
ISTAXS	Tax payment
ISTBIL	Transaction is related to a payment of telecommunications related bill



Code	Description
ISTRAD	Trade services operation
ISTREA	Treasury payment
ISTRPT	Payment for travel
ISUBIL	Utility bill payment
ISVATX	Value added tax payment
ISWHLD	Withholding
ISWTER	Payment of water bill

## Card issuer response codes table

When adding a card the the following `response code` will be received in the response body.

Reason code	Description	Suggested Action
00	Approved	
01	Refer to card issuer	
02	Refer to card issuers special conditions	
03	Invalid merchant	Do not try the transaction again immediately. Contact: <a href="mailto:p2p.support@crossriver.com">p2p.support@crossriver.com</a> .
04	Pickup	Trying again is not permitted with the same PAN or token. Ask for an alternate payment method from the customer and/or advise the cardholder to contact their issuer.
05	Do not honor	Advise the cardholder to contact their issuer.
06	General error	
07	Pickup card, special conditions	
08	Honor with identification	
09	Request in progress	
10	Partial amount approved	
11	VIP approval	
12	Invalid transaction	Trying again is not permitted. Check for potential fraud or issues with the merchant terminal.  1. Review these fields for possible failure conditions: <input type="text" value="CVV"/> or <input type="text" value="Card Authorization Result"/>  2. Check for potential fraud.

Reason code	Description	Suggested Action
13	Invalid amount (currency conversion field overflow) or amount exceeds maximum for card program	
14	Invalid account number (no such number)	<p>Trying again is not permitted with the same PAN or token.</p> <ol style="list-style-type: none"> <li>1. Check card details or reauthorize the card.</li> <li>2. Check for potential fraud</li> </ol>
15	No such issuer	<ol style="list-style-type: none"> <li>1. Trying again is not allowed with the same PAN or token.</li> <li>2. Check the account number again for accuracy.</li> <li>3. Check for potential fraud</li> </ol>
16	Approved, update track 3	
17	Customer cancellation	
18	Customer dispute	
19	Re-enter transaction	
20	Invalid response	
21	No action taken	
22	Suspected malfunction	
23	Unacceptable transaction fee	
24	File update not supported by receiver	
25	Unable to locate record on file	
26	Duplicate file update record, old record replaced	

Reason code	Description	Suggested Action
27	File update field edit error	
28	File is temporarily unavailable	
29	File update not successful, contact acquirer	
30	Format error	Contact: <a href="mailto:p2p.support@crossriver.com">p2p.support@crossriver.com</a> .
31	Bank not supported	Contact: <a href="mailto:p2p.support@crossriver.com">p2p.support@crossriver.com</a> .
32	Completed partially	
33	Expired card	
34	Suspected fraud	
35	Card acceptor, contact acquirer	
36	Restricted card	
37	Card acceptor, call acquirer security	
38	Allowed PIN - tries exceeded	
39	No credit account	
40	Requested function not supported	Contact: <a href="mailto:p2p.support@crossriver.com">p2p.support@crossriver.com</a> .
41	Merchant should retain card (card reported lost)	
42	No universal account	
43	Merchant should retain card (card reported stolen)	Trying again is not permitted with the same PAN or token. Ask for an alternate payment method from the customer and/or advise the cardholder to contact their issuer.

Reason code	Description	Suggested Action
51	Insufficient funds	<p>Merchant can try again for a lesser amount, or they can try again later. This allows the cardholder to fund their debit account or pay down their credit account.</p> <p>Merchants can reduce NSF declines by implementing the Partial Authorization service.</p>
52	No checking account	Advise the cardholder to contact their issuer.
53	No savings account	
54	Expired card	<ol style="list-style-type: none"> <li>1. Check the expiration date before trying again.</li> <li>2. Look out for potential fraud happening through repeated attempts or from suspicious activity.</li> </ol>
55	Incorrect PIN	<ol style="list-style-type: none"> <li>1. Try again with a valid value (re-enter PIN).</li> <li>2. Retry failed transactions as non-PIN, if possible (POS).</li> </ol>
56	No card record	
57	Transaction not permitted for cardholder	Trying again is not permitted with the same PAN or token. Ask for an alternate payment method from the customer and/or advise the cardholder to contact their issuer.

Reason code	Description	Suggested Action
58	Transaction not allowed at terminal	Contact: <a href="mailto:p2p.support@crossriver.com">p2p.support@crossriver.com</a> .
59	Suspected fraud	Advise the cardholder to contact their issuer. Try again with the same PAN or token after the cardholder confirms.
61	Activity amount limit exceeded	To allow the limits to reset, don't retry the transaction the same day. Advise the cardholder to call their issuing bank to find out the limit and reason for the decline.
62	Restricted card - for example, in country exclusion table	We don't recommend trying again immediately with the same PAN or token. Try again after the customer confirms the restriction has been removed.
63	Security violation	
65	Activity count limit exceeded	To allow the limits to reset, don't try the transaction again the same day. Advise the cardholder to contact their issuer.
68	Response received too late	
75	Allowable number of PIN entry tries exceeded	<ol style="list-style-type: none"> <li>1. Try again with a valid value (re-enter PIN).</li> <li>2. If possible, retry failed transactions as non-PIN.</li> </ol>
76	Unable to locate previous message (no match on retrieval reference)	

Reason code	Description	Suggested Action
77	Previous message located for a repeat or reversal, but data is inconsistent with original message	
78	"Blocked, first used" (transaction is from a new card and the card wasn't properly unblocked)	If the cardholder confirms that the card has been activated or reactivated, the merchant can try the transaction again.
80	Visa transactions: credit issuer unavailable - private label and check acceptance: invalid date	
81	PIN cryptographic error found (error found by VIC security module during PIN decryption)	
82	Negative CAM, dCVV, iCVV, or CVV results	Track attempts for potential fraud and check verification values. Focus on CVV.
83	Unable to verify PIN	
85	No reason to decline a request for account number verification, address verification, CVV2 verification or a credit voucher or merchandise return	
91	Issuer unavailable or switch inoperative (STIP not applicable or available for this transaction)	In these cases, <b>don't attempt to retry the transaction</b> . Instead, wait until the next day and check the status in the daily settlement file. Contact: <a href="mailto:p2p.support@crossriver.com">p2p.support@crossriver.com</a> .
92	Destination can't be found for routing	Trying again immediately with the same PAN or token is not allowed.

Reason code	Description	Suggested Action
		Try again after the customer confirms the restriction has been removed.
93	Transaction can't be completed, violation of law	
94	Duplicate transmission	
95	Reconcile error	
96	System malfunction, or certain field error conditions	Contact: <a href="mailto:p2p.support@crossriver.com">p2p.support@crossriver.com</a> .
B1	Surcharge amount not permitted on Visa cards (US acquirers only)	
N0	Force STIP	
N3	Cash service not available	
N4	Cashback request exceeds issuer limit	
N7	Decline for CVV2 failure	
P2	Invalid biller information	
P5	PIN change/unblock request declined	
P6	Unsafe PIN	
Q1	Card authentication failed	
R0	Stop payment order	
R1	Revocation of authorization order	
R3	Revocation of all authorization orders	
XA	Forward to issuer	
XD	Forward to issuer	



Reason code	Description	Suggested Action
Z3	Unable to go online	

## 6.2.9. Error codes

---

Error code	Description
1000 validation	A field in the message didn't pass validation. See the description for more information.
2000 application	The message format is incorrect.
3000 security	A security issue occurred.
9999 system	An internal server system error occurred.

## 6.3. International payments

---

### International payments concepts

The Cross River International Payments APIs enable cross border money movement for International Payments.

**Get started** with Cross River APIs.

Once you have access to our **sandbox** you can access these API endpoints in the `international` module through **Swagger**.

#### GET calls

- Add query parameters to filter the response the API returns to general GET calls.
- Use **pagination** to control presentation of your results.

This table presents APIs that we describe in detail.

Action	API Call	Description
<a href="#"><u>Exchange rate estimate</u></a>	GET /International/v1/estimates	Returns the estimated cost of sending an international payment, including the exchange rate
<a href="#"><u>Required fields list</u></a>	GET /International/v1/meta/quote-requirements	Returns country-specific fields you need to submit when requesting a quote
<a href="#"><u>Request payment quote</u></a>	POST /International/v1/quotes	Requests an executable payment quote for sending either USD or foreign currency before actually sending funds
<a href="#"><u>Send payment</u></a>	POST /International/v1/payments	Executes a payment quote to send funds internationally

## Related topics

- [Webhook events](#)
- [Error codes](#)

## Tutorials

- [International payments](#)

## 6.3.1. APIs

### 6.3.1.1. Exchange rate estimate

---

#### IMPORTANT

You must have a value for either a `fromAmount` or a `toAmount`, but not both.

## Endpoint: /international/v1/estimates

Returns an estimate of an exchange rate. The estimate shows the current foreign exchange rate, including spread and the transaction fee. Unlike a quote, no beneficiary data is required and the estimate cannot be executed. The estimate is not saved. For this call, you must supply your Cross River accountNumber, the desired currency of the received payment, and either amount in USD you plan to send (fromAmount) or the amount in the foreign currency you want sent (toAmount).

GET

<https://sandbox.crbcos.com/international/v1/estimates>



Request

Response

### QUERY PARAMETERS

**accountNumber** String **required**

Account number of the sending Cross River account

**currency** String **required**

Currency the customer is converting funds into. 3-letter ISO currency code.

**fromAmount** Integer **optional**

The amount in USD to be transferred.  
Leave blank if you provide a toAmount.  
Leave out the decimal point between the dollars and the cents.

**toAmount** Integer **optional**

The amount in the specified currency to be received.  
Leave blank if you provide a fromAmount.  
Leave out the decimal point between the dollars and the cents.

**conversionFeeBps** Integer optional

Requested fee for conversion of the funds using basis points (BPS) added onto the foreign exchange rate and earned by the COS partner into their specified account. The fee is added onto the foreign exchange rate when returned in the response. A basis point is 1/100th of a percent of the funds to be transferred.

Partners must be approved to use this attribute to earn conversion fees.

**regularTransactionFeeAmount** Integer optional

Requested fee taken from the sending account for regular payment, earned by the COS partner into their specified account. The regular transaction fee is added onto the platform fee to get the total transaction fee.

Partners must be approved to use this attribute to earn transaction fees.

**priorityTransactionFeeAmount** Integer optional

Requested fee taken from the sending account for priority (SWIFT) payment, earned by the COS partner into their specified account. The priority transaction fee is added onto platform fee to get the total Transaction Fee.

Partners must be approved to use this attribute to earn transaction fees.



Curl



Node.js



Python



Ruby



Go



```
curl --location 'https://sandbox.crbcos.com/international/v1/estimates?acc  
--data ''
```

## Responses

● 200



```
{  
  "accountNumber": "158560897007",  
  "currency": "gbp",  
  "fromAmount": 1000,  
  "toAmount": 789,  
  "exchangeRate": "0.7889",  
  "regularTransactionFeeAmount": 129,  
  "priorityTransactionFeeAmount": 1400  
}
```

For additional information, refer to our [International payments](#) tutorial.



## 6.3.1.2. Required fields list

---

The attributes in the response are required for the quote with the parameters defined in the request for this call (except where indicated in the table below). The values provided describe the required responses. For example, for `lastName` the value is `^.{1,255}`, indicating that the regular expression characters are permitted, up to 255 characters maximum. The `bankCountryCode` is **fr**, as provided in the request.

## Endpoint: /international/v1/meta/quote-requirements

Returns the country-specific fields you need to submit when requesting a quote.

GET

<https://sandbox.crbcos.com/International/v1/meta/quote-requirements> 

Request

Response

### QUERY PARAMETERS

**currency** String **required**

Currency the customer is converting funds into. 3-letter ISO currency code.

**beneficiaryCountryCode** String **required**

Two-letter ISO home bank country code. For a company entity, the country where it's registered. For an individual, the country where they're based.

**bankCountryCode** String **required**

Two-letter ISO transfer bank country code. For a company entity, the country where it's registered. For an individual, the country where they're based.

**entityType** String **optional**

Legal status of the entity, whether beneficiary or originator:

- Individual
- Company
- The default value is individual.

**priority** Boolean **optional**

True if the transfer is via SWIFT. Otherwise false. If you don't supply a value, the default is false.

**pageNumber** Integer **optional**

Number of items per page in a paginated response.

**pageSize** Integer optional

Number of items per page in a paginated response.



Curl



Node.js



Python



Ruby



Go



```
curl --location 'https://sandbox.crbcos.com/International/v1/meta/quote-re'
--data ''
```

## Responses

● 200



```
[
  {
    "firstName": "^[^0-9]{2,255})$",
    "lastName": "^[^0-9]{2,255})$",
    "currency": "gbp",
    "address": "^.{1,255}",
    "city": "^.{1,255}",
    "postalCode": "^.{1,12}$",
    "countryCode": "US",
    "routingCodeType1": "SortCode",
    "routingCodeValue1": "^\\d{6}$",
    "bankName": "^.{1,255}",
    "bankCountryCode": "GB",
    "receiverAccountNumber": "^\\d{8}$",
    "entityType": "Individual",
    "paymentNetwork": "Regular",
    "priority": false
  }
]
```

For additional information, refer to our [International payments](#) tutorial.

### 6.3.1.3. Request payment quote

---

## Endpoint: /international/v1/quotes

Calls an executable international payment quote that returns the cost of the transfer, including any fees and the exchange rate to a foreign currency.

Many beneficiary details are conditionally optional depending on destination country. Call GET /International/v1/meta/quote-requirements for a list of specific required parameters for your quote.

You need beneficiary details to create a quote. You use the value of the id attribute in the response (the Quote ID) to originate an international transfer. Quotes are typically valid for only 30 seconds.

**POST**

<https://sandbox.crbcos.com/International/v1/quotes>



**Request**

Response

### BODY PARAMETERS

**currency** String **required**

Currency the customer is converting funds into. 3-letter ISO currency code.

**accountNumber** String **required**

Account number of the sending Cross River account

**fromAmount** Integer **optional**

The amount in USD to be transferred.

Leave blank if you provide a toAmount.

Leave out the decimal point between the dollars and the cents.

**toAmount** Integer **optional**

The amount in the specified currency to be received.

Leave blank if you provide a fromAmount.

Leave out the decimal point between the dollars and the cents.

**beneficiary ▶**

Object optional

The individual or company who gets the payment

**beneficiaryFi ▶**

Object optional

Details of the beneficiary's financial institution (where the payment goes)

**priority**

Boolean optional

True if the transfer is via SWIFT. Otherwise false. If you don't supply a value, the default is false.

**conversionFeeBps**

Integer optional

Requested fee for conversion of the funds using basis points (BPS) added onto the foreign exchange rate and earned by the COS partner into their specified account. The fee is added onto the foreign exchange rate when returned in the response. A basis point is 1/100th of a percent of the funds to be transferred.

Partners must be approved to use this attribute to earn conversion fees.

**transactionFeeAmount**

Integer optional

The requested flat fee amount for the transaction in USD earned by the COS partner into their specified account

**purpose**

String required

A 1-12 character code that indicates the reason for a transaction. This code is required for certain markets (countries).



Curl



Node.js



Python



Ruby



Go



```
curl --location 'https://sandbox.crbcos.com/International/v1/quotes' \
--data '{
  "currency": "gbp",
  "accountNumber": "158560897007",
  "fromAmount": "500",
  "toAmount": "",
  "beneficiary": {
    "firstName": "Jon",
    "lastName": "Smith",

    "fullName": "JonSmith",

    "birthDate": "2001-06-18T13:05:09.015Z",
    "address": "1 Street",
    "city": "Winfield",
    "stateProvince": "",
    "postalCode": "GB12345",
    "countryCode": "GB",
    "entityType": "Individual"
  },
  "beneficiaryFi": {
    "bankName": "Bank UK",
    "bankCountryCode": "GB",
    "bankAddress": "1 Avenue",
    "bankAccountType": "Checking",

    "routingCodeType1": "SortCode",
    "routingCodeValue1": "123456789",
    "routingCodeType2": "aba",
    "routingCodeValue2": "123456789",
    "bicSwift": "TGCLGB99",
    "iban": "GB33BUKB20201555555555"
  },
  "priority": true,

  "purpose": "SRV"
}'
```

## Responses

● 200



```
{
  "id": "f710a42a-e03a-47b8-a415-b3050061085e",
  "accountNumber": "158560897007",
  "currency": "gbp",
  "beneficiary": {
    "firstName": "Jon",
    "lastName": "Smith",
    "birthDate": "2001-06-18T00:00:00-04:00",
    "address": "1 Street",
    "city": "Winfield",
    "postalCode": "GB12345",
    "countryCode": "GB",
    "entityType": "Individual"
  },
  "beneficiaryFi": {
    "bankName": "Bank UK",
    "bankCountryCode": "GB",
    "bankAddress": "1 Avenue",
    "bankAccountType": "Checking",
    "routingCodeType1": "SortCode",
    "routingCodeValue1": "123456789",
    "routingCodeType2": "ABA",
    "routingCodeValue2": "123456789",
    "bicSwift": "TGCLGB99",
    "iban": "GB33BUKB20201555555555"
  },
  "fromAmount": 500,
  "toAmount": 394,
  "transactionFee": 100,
  "conversionRate": 0.7889,
  "estimatedDeliveryDate": "2025-06-23T00:00:00-04:00",
  "expiresAt": "2025-06-23T01:54:18.8357042-04:00",
  "status": "Created",
  "priority": true,
  "paymentNetwork": "Priority",
  "purpose": "SRV"
}
```



For additional information, refer to our [International payments](#) tutorial.

## 6.3.1.4. Send payment

---

You must get a quote ID using the [request payment quote](#) API. The quote ID is returned in the `id` field in the response to that call.

Quote IDs submitted in request are typically only valid for 30 seconds.

## Endpoint: /international/v1/payments

Originates a cross border money transfer.

**POST**

https://sandbox.crbcos.com/International/v1/payments



**Request**

Response

### BODY PARAMETERS

**quoteld**

String

**required**

ID in GUID format received in the response to the quote call. The quote ID is valid for a limited time, typically 30 seconds.

**clientIdentifier**

String

**optional**

Use this attribute to add your own unique identifying string to a payment call or COS record. This attribute is useful for idempotency purposes.



Curl



Node.js



Python



Ruby



Go



```
curl --location 'https://sandbox.crbcos.com/International/v1/payments' \  
--data '{  
  "quoteId": "f710a42a-e03a-47b8-a415-b3050061085e"  
}'
```

Responses

● 200 ● 400



For additional information, refer to our [International payments](#) tutorial.

# 6.3.2. Webhook events

When you work with Cross River accounts, cards and payments, you register for specific webhooks which have a specific format.

This table presents `international payments` webhook events that are described in detail below.

Event Name	Description
<u>International.Hold.Cleared</u>	Hold on the international payment has cleared and the funds have be sent to the receiver's bank account
<u>International.Hold.Escalated</u>	International payment is on hold and it's status was escalated. This is often used to indicate that additional actions are needed
<u>International.Payment.Sent</u>	International payment has been successfully sent to the receiving institution and the funds are available in the receiver's bank account
<u>International.Payment.Canceled</u>	International payment canceled at partner request
<u>International.Payment.Rejected</u>	International payment was rejected from the receiving bank

## Hold cleared

A hold on the international payment has cleared

### Extended webhook event details



```
{
  "id": "2618e009-7339-41a8-bd4d-b0fb013a523b",
  "eventName": "International.Hold.Cleared",
  "status": "Pending",
  "partnerId": "e30bfd38-907a-45f1-b5b7-af08014c8e79",
  "createdAt": "2024-01-18T14:04:24.507-05:00",
  "resources": [
    "international/v1/payments/aefff994-148e-4f28-9ddc-b0fb013610c7/holds/62a
  ],
  "details": [
    {
      "holdId": "62a79dad-8e47-48a3-a707-b0fb0136137b",
      "paymentId": "aefff994-148e-4f28-9ddc-b0fb013610c7",
      "holdType": "InternalReview"
    }
  ]
}
```

## Hold escalated

International payment is on hold and it's status was escalated. This is often used to indicate that additional actions are needed

### Extended webhook event details





```
{
  "id": "9fd99994-f1e9-4b84-9386-b0fb01381f6f",
  "eventName": "International.Hold.Escalated",
  "status": "Pending",
  "partnerId": "e30bfd38-907a-45f1-b5b7-af08014c8e79",
  "createdAt": "2024-01-18T13:56:24.25-05:00",
  "resources": [
    "international/v1/payments/aefff994-148e-4f28-9ddc-b0fb013610c7/holds/62a
  ],
  "details": [
    {
      "holdId": "62a79dad-8e47-48a3-a707-b0fb0136137b",
      "paymentId": "aefff994-148e-4f28-9ddc-b0fb013610c7",
      "holdType": "InternalReview"
    }
  ]
}
```

## Payment sent

International payment has been successfully sent to the receiving institution and the funds are available in the receiver's bank account

### Extended webhook event details

```
{
  "id": "9370fa9b-3e1e-4077-a1f9-b0fb013a7568",
  "eventName": "International.Payment.Sent",
  "status": "Pending",
  "partnerId": "e30bfd38-907a-45f1-b5b7-af08014c8e79",
  "createdAt": "2024-01-18T14:04:54.533-05:00",
  "resources": [
    "international/v1/payments/aefff994-148e-4f28-9ddc-b0fb013610c7"
  ],
  "details": [
    {
      "paymentId": "aefff994-148e-4f28-9ddc-b0fb013610c7",
      "productId": "e8164629-96ad-47d9-ba6a-af08014dd6d1",
      "quoteId": "b7f05b12-3205-4328-8eb6-b0fb0135fa26",
      "fromCurrency": "usd",
      "toCurrency": "eur",
      "fromAmount": "1026",
      "toAmount": "1075",
      "feeAmount": "1510",
      "accountNumber": "2283489405",
      "status": "Completed",
      "reason": null,
      "clientIdentifier": "string",
      "priority": "True",
      "beneficiaryEntityType": "Individual",
      "beneficiaryCompanyName": null,
      "beneficiaryFirstName": "John",
      "beneficiaryLastName": "Doe",
      "beneficiaryAddress": "1234 somewhere lane",
      "beneficiaryCity": "seattle",
      "beneficiaryStateProvince": "WA",
      "beneficiaryPostalCode": "98001",
      "beneficiaryCountryCode": "us"
    }
  ]
}
```

## Payment canceled

International payment canceled at partner request

## Extended webhook event details



Sample Event



```
{
  "id": "9370fa9b-3e1e-4077-a1f9-b0fb013a7568",
  "eventName": "International.Payment.Canceled",
  "status": "Pending",
  "partnerId": "e30bfd38-907a-45f1-b5b7-af08014c8e79",
  "createdAt": "2024-01-18T14:04:54.533-05:00",
  "resources": [
    "international/v1/payments/aefff994-148e-4f28-9ddc-b0fb013610c7"
  ],
  "details": [
    {
      "paymentId": "aefff994-148e-4f28-9ddc-b0fb013610c7",
      "productId": "e8164629-96ad-47d9-ba6a-af08014dd6d1",
      "quoteId": "b7f05b12-3205-4328-8eb6-b0fb0135fa26",
      "fromCurrency": "usd",
      "toCurrency": "eur",
      "fromAmount": "1026",
      "toAmount": "1075",
      "feeAmount": "1510",
      "accountNumber": "2283489405",
      "status": "Canceled",
      "reason": null,
      "clientIdentifier": "string",
      "priority": "True",
      "beneficiaryEntityType": "Individual",
      "beneficiaryCompanyName": null,
      "beneficiaryFirstName": "John",
      "beneficiaryLastName": "Doe",
      "beneficiaryAddress": "1234 somewhere lane",
      "beneficiaryCity": "seattle",
      "beneficiaryStateProvince": "WA",
      "beneficiaryPostalCode": "98001",
      "beneficiaryCountryCode": "us"
    }
  ]
}
```

**Payment rejected**

International payment was rejected from the receiving bank

## Extended webhook event details



Sample Event



```
{
  "id": "9370fa9b-3e1e-4077-a1f9-b0fb013a7568",
  "eventName": "International.Payment.Rejected",
  "status": "Pending",
  "partnerId": "e30bfd38-907a-45f1-b5b7-af08014c8e79",
  "createdAt": "2024-01-18T14:04:54.533-05:00",
  "resources": [
    "international/v1/payments/aefff994-148e-4f28-9ddc-b0fb013610c7"
  ],
  "details": [
    {
      "paymentId": "aefff994-148e-4f28-9ddc-b0fb013610c7",
      "productId": "e8164629-96ad-47d9-ba6a-af08014dd6d1",
      "quoteId": "b7f05b12-3205-4328-8eb6-b0fb0135fa26",
      "fromCurrency": "usd",
      "toCurrency": "eur",
      "fromAmount": "1026",
      "toAmount": "1075",
      "feeAmount": "1510",
      "accountNumber": "2283489405",
      "status": "Rejected",
      "reason": null,
      "clientIdentifier": "string",
      "priority": "True",
      "beneficiaryEntityType": "Individual",
      "beneficiaryCompanyName": null,
      "beneficiaryFirstName": "John",
      "beneficiaryLastName": "Doe",
      "beneficiaryAddress": "1234 somewhere lane",
      "beneficiaryCity": "seattle",
      "beneficiaryStateProvince": "WA",
      "beneficiaryPostalCode": "98001",
      "beneficiaryCountryCode": "us"
    }
  ]
}
```



### 6.3.3. Error codes

---

Code	Description
2000	General exception
2100	Beneficiary is not valid
2101	Currency is not supported
2102	Country is not supported
2103	Validation response is invalid
2200	This action requires dual control
2201	Global profile change not pending
2202	Global profile change not approved
2203	Global config change approval requires dual control
2204	Global config required
2205	Product change not pending
2206	Product change not approved
2207	Product config change approval requires dual control
2208	Product config required
2209	Account config required
2210	Account change not pending
2211	Account change not approved
2212	Account config change approval requires dual control
2213	Account configuration not found or is invalid
2214	Global profile not found or is invalid
2215	No active configuration found for account
2216	Currency not supported
2217	Product Config Change includes settings that are not supported for Product Type
2218	The Operator Account Number has not been set in the Global Configuration

Code	Description
2219	Invalid currency for partner fee
2220	Duplicate currency for partner fees
2221	All required accounts have not been configured in the Global Configuration
2222	All required accounts have not been configured in the Account Configuration
2223	The Customer for the specified Account Number was not found
2224	The Swift Aco Flat Rate has not been set in the Global Configuration
2225	The Local Payment Fee has not been set in the Global Configuration for the requested currency
2226	The Transaction Fees have not been set in the Global Configuration
2227	Inbound payments not enabled for account
2228	Outbound payments not enabled for account
2229	Limit start time (HH:mm) not set for account
2230	Daily count limit not set for account
2231	Daily amount limit not set for account
2232	Transfer amount limit not set for account
2233	Cannot exceed daily count limit for account
2234	Cannot exceed daily amount limit for account
2235	Cannot exceed transfer amount limit for account
2236	Unsupported Country Code(s)
2237	There are accounts misconfigured in the AccountConfig or GlobalConfig
2300	No estimate rate found for request
2301	Fees not found for account or currency
2302	Unable to retrieve rates at this time
2303	Currency not supported



Code	Description
2400	Beneficiary failed to create for quote
2401	Estimate not found for quote
2402	Fees not found for account or currency
2403	Quote not found for originating payment
2404	Quote has expired
2405	Currency is not supported for selected bank country
2406	Quote amount does not exceed minimum amount for conversion
2500	Account configuration not found or is invalid
2501	Invalid payment status
2502	Payment not found for id
2503	Quote has already been executed
2504	Invalid payment Type
2505	Account Type cannot be used for payments
2506	Posting status must be failed to attempt a retry
2507	Invalid payment fee status
2600	Hold is not active
2601	Hold payment not found
2602	Hold account not found
2700	Unable to parse line from CSV file for conversion detail report
2701	Failed to generate conversion report
2702	Failed to generate payment report
2703	Invalid response after generating report
2704	Unable to parse line from CSV file for payment detail report
2705	Vendor report not found
2706	Conversion file not found

Code	Description
2707	Vendor report not found using external id
2800	Payment detail not found"

## 6.4. ACH

---

### ACH concepts

The Cross River ACH API allows you to both originate and receive payments through the Federal Reserve ACH network.

**Get started** with Cross River APIs.

If you have access to our **sandbox** you can access these API endpoints in the `ach` module through **Swagger**.

### GET calls

- Add query parameters to filter the response the API returns to general GET calls.
- Use **Pagination** to control presentation of your results.

This table presents ACH APIs that we describe in detail.

Action	API Call	Description
<a href="#"><u>Send payment</u></a>	POST <code>/ach/v1/payments</code>	Sends a single payment using ACH
<a href="#"><u>Originate ACH client batches</u></a>	POST <code>/ach/v1/client-batches</code>	Originates several payments together in a batch
<a href="#"><u>Cancel batch</u></a>	POST <code>/ach/v1/client-batches/{id}/cancel</code>	Cancels any incomplete payments and stops further processing of any remaining payments in the batch
<a href="#"><u>Return a payment</u></a>	POST <code>/ach/v1/payments/{id}/returns</code>	Initiates a return for an existing ACH payment.
<a href="#"><u>Check limit utilization</u></a>	GET <code>ach/v1/limits-utilization/{productId}</code>	Returns the percentage of your ACH payment limit used according to the specific product used to initiate ACH payments
<a href="#"><u>Check limit utilization by product and limit date</u></a>	GET <code>ach/v1/limits-utilization/{productId}/{limitDate}</code>	Returns the percentage of your ACH payment limit used for a specific product and date

## Related topics

- [Webhook events](#)
- [Request and response codes](#)
- [Error codes](#)

## Tutorials

- [Send an ACH payment](#): Originate a standard or same-day ACH payment.
- [Send a client batch](#): Send two or more payments in the same call.

- **Simulate inbound ACH payments**: Test the system with our simulation endpoints. You can also use this endpoint to fund a test account.

## 6.4.1. APIs

### 6.4.1.1. Send payment

---

#### IMPORTANT

The payment information and data you send to Cross River must include a Nacha SEC code and conform to Nacha settlement rules.

## Endpoint: /ach/v1/payments

Sends a single payment using ACH.

**POST**

https://sandbox.crbcos.com/ach/v1/payments



**Request**

Response

### BODY PARAMETERS

**accountNumber**

String

**required**

Cross River account number for payment originator. Depending on the account type, the number of digits will vary. The first digit indicates the kind of account.

**originator ▶**

Object

**optional**

Details about the entity sending the payment. If the account originating the payment is enabled to provide this information according to the supplied account number, all the values should be null. If these values are supplied in the request they override the originator profile defined for the account.

**receiver ▶**

Object

**required**

In the case of Pull payments, this is the account the funds are being pulled from.

**secCode**

String

**required**

Standard Entry Class (SEC) code required by Nacha for every ACH transaction. See the ACH Request and response code page.

**description**

String

**required**

Description of the payment. Maximum 10 characters.

**transactionType** String **required**

Sending or requesting funds:

- Push (originator is sending funds—debit)
- Pull (originator is receiving funds—credit)

**amount** Integer **required**

Dollar amount of payment in positive integral cents. For example, \$1.00 appears as 100.

**serviceType** String **required**

When the payment completes:

- Standard (Payment effective the following business day. International payments must use this service type.)
- SameDay (Payment effective same day, subject to certain constraints. See ACH overview.)

**fedBatchWindow** Integer **optional**

**extendedDetails** ▶ Object **optional**

ACH-specific details.

If the transactionType is pull and secCode is WEB you must include a paymentType value in this object.

If the secCode is POS you must include a cardTransactionType value.

[Learn more on the Nacha site.](#)

**iatDetails** ▶ Object **optional**

Required if the secCode is IAT.

Do not enter values for these attributes for any other secCode.



**addenda**                      String              optional

Supplemental data to identify an account holder or provide information about the payment. This may include details such as invoice numbers.

**purpose**                      String              optional

Reason for the ACH transfer. For internal use only. The data is not included with the outgoing ACH batch file.

**clientIdentifier**              String              optional

Use this attribute to add your own unique identifying string to a payment call or COS record. This attribute is useful for idempotency purposes.

**externalPrefix**              String              optional

Prefix from external system integration

**externalAccountNumber**      String              optional

External-facing account reference number



Curl



Node.js



Python



Ruby



Go



```
curl --location 'https://sandbox.crbcos.com/ach/v1/payments' \
--header 'Idempotency-key: c10314ee-e737-4681-8e8d-f9cf547c9967' \
--data '{
  "accountNumber": "2207975570",
  "receiver": {
    "routingNumber": "021000021",
    "accountNumber": "2151546989",
    "accountType": "Checking",
    "name": "Peter Griffin"
  },
  "secCode": "PPD",
  "description": "string",
  "transactionType": "Push",
  "amount": 2500,
  "serviceType": "SameDay",
  "clientIdentifier": "c10314ee-e737-4681-8e8d-f9cf547c9967"
}'
```

Responses

● 200





## 6.4.1.2. Send batch

---

## Endpoint: /ach/v1/client-batches

Submits 2 or more payments as a batch. Note that each batch payment must be originated from the same Cross River account number. If originating from multiple Cross River accounts, the payments from each account must be submitted as separate batches. For each payment, include the request attributes in the payments object of the request as shown in the code snippet.

**POST**

https://sandbox.crbcos.com/ach/v1/client-batches



**Request**

Response

### BODY PARAMETERS

**clientIdentifier**

String

optional

Use this attribute to add your own unique identifying string to a payment call or COS record. This attribute is useful for idempotency purposes.

Use this field to enter a clientIdentifier for the entire batch. You can also add a separate clientIdentifier for each payment.

**payments ▶**

Object

optional

The information for each payment in the batch.

**extendedDetails ▶**

Object

optional

ACH-specific details.

If the transactionType is pull and secCode is WEB you must include a paymentType value in this object.

If the secCode is POS you must include a cardTransactionType value.

Learn more on the Nacha site.

**iatDetails ▶**

Object

optional

Required if the secCode is IAT.

Do not enter values for these attributes for any other secCode.

**addenda**

String

optional

Supplemental data to identify an account holder or provide information about the payment. This may include details such as invoice numbers.

**purpose**

String

optional

Reason for the ACH transfer. For internal use only. The data is not included with the outgoing ACH batch file.

**clientIdentifier**

String

optional

Use this attribute to add your own unique identifying string to a payment call or COS record. This attribute is useful for idempotency purposes.

**externalPrefix**

String

optional

Prefix from external system integration

**externalAccountNumber**

String

optional

External-facing account reference number



Curl



Node.js



Python



Ruby



Go





```
curl --location 'https://sandbox.crbcos.com/ach/v1/client-batches' \
--header 'Idempotency-key: 9edef3ad-8084-45d2-ab71-c0d706cd221a' \
--data '{
  "payments": [
    {
      "accountNumber": "2151546989",
      "receiver": {
```

## Responses

● 200



```
{
  "id": "c51248db-9e07-4439-ace4-b1d800fd803a",
  "referenceId": "CB239T00YK9IF",
  "status": "Processing",
  "accountNumber": "2207075570"
```

### 6.4.1.3. Cancel batch

---

## Endpoint: /ach/v1/client-batches/{id}/cancel

Cancel an ACH client batch by its ID. Cancels any incomplete payments and stops further processing of any remaining payments in the batch. If you need to cancel many payments in a client batch, it's most efficient to cancel a client batch before all the payments have imported rather than try to cancel a large number of payments individually.

**POST**

<https://sandbox.crbcos.com/ach/v1/client-batches/{id}/cancel>



**Request**

Response

### PATH PARAMS

**id**

String

**required**

The unique ID of the batch payment, received in the batch origination response as the id attribute. This ID is in GUID format.



Curl



Node.js



Python



Ruby



Go



```
curl --location 'https://sandbox.crbcos.com/ach/v1/client-batches/63f87c2b' \
--header 'Content-Type: application/json' \
--header 'Authorization: Bearer <token>' \
--data '
```

Responses

● 200





## 6.4.1.4. Returns

---

## Endpoint: /ach/v1/payments/{id}/returns

Initiates a return for an existing ACH payment. Returns must include a valid Nacha return code, can also include descriptive metadata. Returned payments reference the original transaction and include updated payment details.

**POST**

https://sandbox.crbcos.com/ach/v1/payments/{id}/returns



**Request**

Response

### PATH PARAMS

**id** String **required**

Unique identifier of the original ACH payment to be returned

### BODY PARAMETERS

**returnCode** String **required**

Valid ACH return code in accordance with NACHA guidelines (for example, R01, R02). See the ACH Request and Response Codes page for the return codes.

**reasonData** String **optional**

More information about the reason for the return. 34 character maximum.





Curl



Node.js



Python



Ruby



Go



```
curl --location 'https://sandbox.crbcos.com/ach/v1/payments/2df92265-858e-  
--header 'Content-Type: application/json' \  
--header 'Authorization: Bearer <token>' \  
--data '{  
"returnCode": "R05",  
'
```

Responses

● 200





## 6.4.1.5. Limits utilization

### Endpoint: ach/v1/limits-utilization/{productId}

Returns the percentage of your ACH payment limit used

GET

<https://sandbox.crbcos.com/ach/v1/limits-utilization/{productId}> 

Request

Response

#### PATH PARAMS

**productId** String **required**

ID in GUID format of the product used to initiate ACH payments



Curl



Node.js



Python



Ruby



Go



```
curl --location 'https://sandbox.crbcos.com/ach/v1/limits-utilization/a68a
```

#### Responses

● 200



```
{
  "productId": "a68a0fa5-98fe-4cd5-b626-af5b0137e0cb",
  "partnerId": "cd9c12f4-7691-424a-b38b-af5b0134c611",
  "limitDate": "2025-07-03T00:00:00-04:00",
  "pullPercent": 0,
  "pushPercent": 0,
  "totalCountPercent": 0
}
```

#### 6.4.1.6. Limits utilization by limit date

---

## Endpoint: ach/v1/limits-utilization/{productId}/{limitDate}

Returns the percentage of your ACH payment limit used according to limit date

GET

<https://sandbox.crbcos.com/ach/v1/limits-utilization/{productId}/{limitDate}> 

Request

Response

### PATH PARAMS

**productId**

String

required

ID in GUID format of the product used to initiate ACH payments

**limitDate**

String

required

Date the limit is set for. This attribute is data type DateTime. Use the format YYYY-MM-DD.



Curl



Node.js



Python



Ruby



Go



```
curl --location 'https://sandbox.crbcos.com/ach/v1/limits-utilization/a68a0fa5-98fe-4cd5-b626-af5b0137e0cb' \
--header 'Authorization: Bearer <token>'
```

### Responses

● 200



```
{
  "productId": "a68a0fa5-98fe-4cd5-b626-af5b0137e0cb",
  "partnerId": "cd9c12f4-7691-424a-b38b-af5b0134c611",
  "limitDate": "2025-07-10T00:00:00-04:00",
  "pullPercent": 0,
  "pushPercent": 0,
  "totalCountPercent": 0
}
```

## 6.4.2. Webhook events

---

When you work with Cross River accounts, cards and payments, you register for specific **webhooks** which have a specific **format**.

This table presents ACH webhook events that are described in detail below.

Event Name	Description
<u><b>Ach.Batch.Canceled</b></u>	Any pending or on-hold payments in an ACH client batch canceled
<u><b>Ach.Batch.Imported</b></u>	Entire ACH client batch import process into the CR system complete
<u><b>Ach.Payment.Canceled</b></u>	ACH payment canceled at partner request
<u><b>Ach.Payment.Rejected</b></u>	ACH payment rejected due to a compliance failure or fail to fund
<u><b>Ach.Payment.Sent</b></u>	ACH payment sent to the Federal Reserve
<u><b>Ach.Payment.Sent (Client Batch)</b></u>	
<u><b>Ach.Payment.Received</b></u>	ACH inbound payment received
<u><b>Ach.Payment.ReceivedEarly</b></u>	ACH early direct deposit payment received
<u><b>Ach.Return.Received</b></u>	ACH payment was returned from the receiving bank and has been marked as received
<u><b>Ach.Noc.Received</b></u>	A Notification of Change was sent from the receiving bank regarding a previous origination
<u><b>Ach.Hold.Escalated</b></u>	ACH inbound or outbound payment is on hold and it's status was escalated. This is often used to indicate that additional actions are needed

The values used in the sample responses are examples and are not valid.

## Ach.Batch.Canceled

Description:



## Any pending or on-hold payments in an ACH client batch have been canceled

The extended webhook response includes the batch ID and the account number.

### Extended webhook event response



```
{
  "id": "1a29dbb3-df40-42f0-aaa8-b04900e37f0e",
  "eventName": "Ach.Batch.Canceled",
  "status": "Pending",
  "partnerId": "1e5d3f04-ae24-4af6-9e30-aecf012b99dd",
  "createdAt": "2023-07-24T09:48:17.33-04:00",
  "resources": [
    "ach/v1/payments/652c16be-b660-426f-9690-bd41d5960d0b"
  ],
  "details": [
    {
      "clientBatchId": "652c16be-b660-426f-9690-bd41d5960d0b",
      "accountNumber": "1234567890"
    }
  ]
}
```

## Ach.Batch.Imported

Description:

**The entire ACH client batch import process into the CR system is complete**

The extended webhook response includes the batch ID and the account number.



```
{
  "id": "1a29dbb3-df40-42f0-aaa8-b04900e37f0e",
  "eventName": "Ach.Batch.Imported",
  "status": "Pending",
  "partnerId": "1e5d3f04-ae24-4af6-9e30-aecf012b99dd",
  "createdAt": "2023-07-24T09:48:17.33-04:00",
  "resources": [
    "ach/v1/payments/652c16be-b660-426f-9690-bd41d5960d0b"
  ],
  "details": [
    {
      "clientBatchId": "652c16be-b660-426f-9690-bd41d5960d0b",
      "accountNumber": "1234567890"
    }
  ]
}
```

## Ach.Payment.Canceled

Description:

### ACH payment canceled at partner request

The extended event response includes the client identifier, the account number the payment was sent to, the payment ID, the reason the payment was rejected, and if relevant, the [memo post](#) ID and any client batch information.

```
{
  "id": "62d4c953-6ee2-438f-8966-b04900def646",
  "eventName": "Ach.Payment.Canceled",
  "status": "Pending",
  "partnerId": "1e5d3f04-ae24-4af6-9e30-aecf012b99dd",
  "createdAt": "2023-07-24T09:31:46.793-04:00",
  "resources": [
    "ach/v1/payments/73da01c7-b85b-4a58-9395-b04900de43cf"
  ],
  "details": [
    {
      "paymentId": "73da01c7-b85b-4a58-9395-b04900de43cf",
      "coreTransactionId": null,
      "memoPostId": "85b509f9-61f0-4af3-b64c-b04900de43da",
      "clientBatchId": null,
      "clientBatchSequence": null,
      "accountNumber": "2151546989",
      "postingCode": null,
      "clientIdentifier": null,
      "purpose": "ENTERED BY #60C3C9C8FD17BA0070A7FB4F#"
    }
  ]
}
```

## Ach.Payment.Rejected

Description:

### ACH payment rejected due to a compliance failure or fail to fund

The extended event response includes the client identifier, the account number the payment was sent to, the payment ID, the reason the payment was rejected, and if relevant, the [memo post](#) ID and any client batch information.



```
{
  "id": "68ef3d97-758c-470e-b3f7-b1c6012cfd79",
  "eventName": "Ach.Payment.Rejected",
  "status": "Pending",
  "partnerId": "1e5d3f04-ae24-4af6-9e30-aecf012b99dd",
  "createdAt": "2024-08-08T14:15:52.297-04:00",
  "resources": [
    "ach/v1/payments/edb404e7-5a00-4e07-8da2-b1c6012ce044"
  ],
  "details": [
    {
      "paymentId": "edb404e7-5a00-4e07-8da2-b1c6012ce044",
      "coreTransactionId": "36d78c83-7be8-40e9-9aa9-b1c6012ce044",
      "memoPostId": "4a6d8ff4-22cf-437a-acbe-b1c6012ce044",
      "clientBatchId": null,
      "clientBatchSequence": null,
      "accountNumber": "2153808510",
      "postingCode": "NSF",
      "clientIdentifier": null,
      "purpose": null,
      "rejectionReason": "Other"
    }
  ]
}
```

## Ach.Payment.Sent

**The ACH payment was sent to the Federal Reserve.**

The extended event response includes the client identifier, the account number the payment was sent to, the payment ID, the purpose the payment, and if relevant, the memo post ID and any client batch information.



```
{
  "id": "51d2e4e8-5fee-4ad8-8b03-b1c6012a8459",
  "eventName": "Ach.Payment.Sent",
  "status": "Pending",
  "partnerId": "1e5d3f04-ae24-4af6-9e30-aecf012b99dd",
  "createdAt": "2024-08-08T14:06:52.013-04:00",
  "resources": [
    "ach/v1/payments/55f641cf-102b-4920-83fd-b1c6012a3d2d"
  ],
  "details": [
    {
      "paymentId": "55f641cf-102b-4920-83fd-b1c6012a3d2d",
      "coreTransactionId": "fddea7f7-556e-4c93-acc4-b1c6012a6bf3",
      "memoPostId": "fddea7f7-556e-4c93-acc4-b1c6012a6bf3",
      "clientBatchId": null,
      "clientBatchSequence": null,
      "accountNumber": "2151546989",
      "postingCode": null,
      "clientIdentifier": null,
      "purpose": null,
      "fedBatchId": "29f4b78f-f056-4ac5-9a4e-b1c6012a60d2",
      "fedBatchSequence": null
    }
  ]
}
```

## Ach.Payment.Sent (Client Batch)





```
{
  "id": "af676b07-fb1f-4f7e-9ea1-b2120122c52e",
  "eventName": "Ach.Payment.Sent",
  "status": "Pending",
  "partnerId": "ba9d0234-26e9-43a2-9070-aedd0108eadb",
  "createdAt": "2024-10-23T13:38:39.72-04:00",
  "resources": [
    "ach/v1/payments/76b1aaaa-4c65-400f-85ee-b21201225e4c",
    "ach/v1/payments/65120c3b-3cd1-4909-b126-b21201225e4c",
    "ach/v1/payments/7eb3ab00-7137-4650-ac5e-b21201225e4c",
    "ach/v1/payments/0e8fc9e4-8f76-4505-b99a-b21201225e4c"
  ],
  "details": [
    {
      "paymentId": "76b1aaaa-4c65-400f-85ee-b21201225e4c",
      "coreTransactionId": "fb7c002e-e5ba-4dd3-97a9-b21201225e5b",
      "memoPostId": "007c4052-b8f8-4f91-aa8e-b21201225e5b",
      "clientBatchId": "e2fdb85b-ab31-42b5-88e9-b21201225e4c",
      "clientBatchSequence": null,
      "accountNumber": "2405952710",
      "postingCode": null,
      "clientIdentifier": "2fd4fcb2-e4ad-43fb-88b7-4c06b054d1c3",
      "purpose": null,
      "fedBatchId": "0fade146-d07b-4b9c-a173-b2120122999d",
      "fedBatchSequence": null
    },
    {
      "paymentId": "65120c3b-3cd1-4909-b126-b21201225e4c",
      "coreTransactionId": "a874d749-c347-4b87-a552-b21201225e56",
      "memoPostId": "27199405-5cdd-4d84-aca5-b21201225e56",
      "clientBatchId": "e2fdb85b-ab31-42b5-88e9-b21201225e4c",
      "clientBatchSequence": null,
      "accountNumber": "2405952710",
      "postingCode": null,
      "clientIdentifier": "9c1f2b48-48ef-4bba-8275-06ae6bb5d17a",
      "purpose": null,
      "fedBatchId": "0fade146-d07b-4b9c-a173-b2120122999d",
      "fedBatchSequence": null
    },
    {
      "paymentId": "7eb3ab00-7137-4650-ac5e-b21201225e4c",
      "coreTransactionId": "85f632c0-9e73-4cdc-b458-b21201225e69",
      "memoPostId": "c89b2ba0-3330-4b69-9e09-b21201225e69",
      "clientBatchId": "e2fdb85b-ab31-42b5-88e9-b21201225e4c",
      "clientBatchSequence": null,
    }
  ]
}
```



```

    "accountNumber": "2405952710",
    "postingCode": null,
    "clientIdentifier": "1feae24a-58ab-4f92-b917-a250f157ccf4",
    "purpose": null,
    "fedBatchId": "0fade146-d07b-4b9c-a173-b2120122999d",
    "fedBatchSequence": null
  },
  {
    "paymentId": "0e8fc9e4-8f76-4505-b99a-b21201225e4c",
    "coreTransactionId": "371543d0-21b7-4325-a27e-b21201225e6e",
    "memoPostId": "895ebd94-7e20-4bb6-b693-b21201225e6e",
    "clientBatchId": "e2fdb85b-ab31-42b5-88e9-b21201225e4c",
    "clientBatchSequence": null,
    "accountNumber": "2405952710",
    "postingCode": null,
    "clientIdentifier": "6c5dfeac-6dce-4ac0-a2fe-39f2bbf1722d",
    "purpose": null,
    "fedBatchId": "0fade146-d07b-4b9c-a173-b2120122999d",
    "fedBatchSequence": null
  }
]
}

```

## Ach.Payment.Received

Description:

### An ACH inbound payment was received to your account

The extended event response includes the payment and core transaction IDs, the trace number and routing number of the payment, the name, ID and account number of the party who received the payment, general information on the payment, and the amount.

```
{
  "id": "1630d863-d1b3-4c3a-8943-b2db01558dab",
  "eventName": "Ach.Payment.Received",
  "status": "Pending",
  "partnerId": "1e5d3f04-ae24-4af6-9e30-aecf012b99dd",
  "createdAt": "2025-05-12T16:43:33.473+00:00",
  "resources": [
    "ach/v1/payments/8e72c7df-c52c-48d4-aa5b-b2db01554571"
  ],
  "details": [
    {
      "paymentId": "8e72c7df-c52c-48d4-aa5b-b2db01554571",
      "coreTransactionId": "9d1f39e4-4c78-48ac-89ce-b2db01554583",
      "accountNumber": "2254575653",
      "traceNumber": "021000020412472",
      "routingNumber": "021000021",
      "originatorName": "SUBTESTING",
      "originatorIdentification": "3333385474",
      "description": "Refund",
      "transactionType": "Push",
      "serviceType": "SameDay",
      "amount": "15000",
      "addenda": null,
      "receiverName": "James Webb",
      "receiverIdentification": "",
      "receiverAccountNumber": "2254575653",
      "secCode": "PPD",
      "effectiveDate": "250512",
      "settlementDate": "132"
    }
  ]
}
```

## Ach.Payment.ReceivedEarly

Description:

**An ACH early direct deposit payment received**

The extended event response includes the payment and core transaction IDs, the trace number and routing number of the payment, the name, ID and account number of the party who originated the payment, general information on the payment, and the amount.

#### Extended webhook event response



```
{
  "id": "b7ebcb38-5242-4d02-83c4-b1a9010fedb6",
  "eventName": "Ach.Payment.ReceivedEarly",
  "status": "Pending",
  "partnerId": "1e5d3f04-ae24-4af6-9e30-aecf012b99dd",
  "createdAt": "2024-07-10T12:30:03.707-04:00",
  "resources": [
    "ach/v1/payments/91ec2ab1-a06e-4862-9f88-b1a9010f61e3"
  ],
  "details": [
    {
      "paymentId": "91ec2ab1-a06e-4862-9f88-b1a9010f61e3",
      "coreTransactionId": "9c32bbe8-7488-4d28-b525-b1a9010f61f1",
      "accountNumber": "2863654337",
      "traceNumber": "021214896305256",
      "routingNumber": "021214891",
      "originatorName": "Pay Corp",
      "originatorIdentification": "123",
      "description": "Payout",
      "transactionType": "Push",
      "serviceType": "Standard",
      "amount": "50000",
      "addenda": null,
      "receiverName": "B Sanderson",
      "receiverIdentification": "",
      "receiverAccountNumber": "2863654337",
      "secCode": "PPD"
    }
  ]
}
```

## Ach.Return.Received

Description:

**ACH payment was returned from the receiving bank and has been marked as received.**  
**An ACH return is a new unique payment with a unique payment ID**

The extended event response includes the client identifier, the original payment ID, the account number the payment was sent to, the payment ID, and the reason the payment was returned.

Extended webhook event response



```
{
  "id": "6c66de98-138b-40b2-9a16-b1c6012be415",
  "eventName": "Ach.Return.Received",
  "status": "Pending",
  "partnerId": "1e5d3f04-ae24-4af6-9e30-aecf012b99dd",
  "createdAt": "2024-08-08T14:11:52.173-04:00",
  "resources": [
    "ach/v1/payments/806a4c4f-d536-45aa-89f6-b1c6012b9cbe"
  ],
  "details": [
    {
      "paymentId": "806a4c4f-d536-45aa-89f6-b1c6012b9cbe",
      "coreTransactionId": "fc79ca11-350e-4b08-8bf1-b1c6012b9ccd",
      "originalPaymentId": "55f641cf-102b-4920-83fd-b1c6012a3d2d",
      "accountNumber": "2151546989",
      "traceNumber": "021000029551621",
      "reasonCode": "R01",
      "reasonData": "",
      "amount": "4500",
      "transactionType": "Push"
    }
  ]
}
```

## Ach.Noc.Received

Description:

**A Notification of Change was sent from the receiving bank regarding a previous origination**

The extended webhook response includes information on the payment origination and the reason for the NOC.

#### Extended webhook event response



```
{
  "id": "f26b80ef-cfc2-4cfb-8888-b1c6012c70c9",
  "eventName": "Ach.Noc.Received",
  "status": "Pending",
  "partnerId": "1e5d3f04-ae24-4af6-9e30-aecf012b99dd",
  "createdAt": "2024-08-08T14:13:52.243-04:00",
  "resources": [
    "ach/v1/payments/9e0c6a4b-303a-4cdd-a987-b1c6012c297f"
  ],
  "details": [
    {
      "paymentId": "9e0c6a4b-303a-4cdd-a987-b1c6012c297f",
      "coreTransactionId": "",
      "originalPaymentId": "6f733a5a-7614-4895-bf56-b1c6012a854c",
      "accountNumber": "2153808510",
      "traceNumber": "021000020305877",
      "reasonCode": "C01",
      "reasonData": "452136846",
      "amount": "0",
      "transactionType": "Push"
    }
  ]
}
```

## Ach.Hold.Escalated

Description:

**ACH inbound or outbound payment is on hold and it's status was escalated.**

This can indicate that additional actions are needed.

```
{
  "id": "1a29dbb3-df40-42f0-aaa8-b04900e37f0e",
  "eventName": "Ach.Hold.Escalated",
  "status": "Pending",
  "partnerId": "1e5d3f04-ae24-4af6-9e30-aecf012b99dd",
  "createdAt": "2023-07-24T09:48:17.33-04:00",
  "resources": [
    "ach/v1/payments/652c16be-b660-426f-9690-bd41d5960d0b"
  ],
  "details": [
    {
      "holdId": "7413f441-910b-4e75-bbee-9b9ac27d717d",
      "paymentId": "652c16be-b660-426f-9690-bd41d5960d0b",
      "holdType": "
    }
  ]
}
```

## 6.4.3. Request and response codes

---

### SEC codes

A SEC code indicates the Standard Entry Class required by Nacha for every ACH transaction. When you **originate an ACH payment** by API, you enter this code in the `secCode` field of the request. The table below explains common SEC codes. Learn more about SEC codes on the **Nacha website**.

Which code should I use?

Your Operations Support Team will guide with this based on the use case you are trying to support.

Code	Name	Description	Type	Transaction Type	Credit/Debit
ARC	Accounts Receivable Entry	A single debit entry to an account initiated for purchases or payments that are made in person, via US mail, or placing the payment in a dropbox.	Consumer or Non-Consumer Mail Order or Retail	Single Entry	Debit only
BOC	Back Office Conversion Entry	A single debit entry to an account for in-person purchases or payments made at the point-of-purchase. <i>Check is not returned to check writer.</i>	Consumer or Non-Consumer Retail	Single Entry	Debit only
CCD	Corporate Credit/Debit Entry	Funds are transferred between unrelated corporate entities or transferred as intra-company cash concentration and disbursement transactions.	Non-Consumer	Single or Recurring Entry	Credit/Debit



Code	Name	Description	Type	Transaction Type	Credit/Debit
		Proof of Authorization for transactions ran on the web using CCD will adhere to the same requirements as a web transaction.	Retail, Phone Order, Ecommerce , and Mail Order	Single or Recurring Entry	Credit/Debit
CIE	Customer Initiated Entry	A single credit entry initiated by a consumer (originator) to a non-consumer account. CIE is a consumer-initiated credit entry, a credit push to a merchant, typically used by a financial institution's or third-party's bill pay service and would be not be Originated by a merchant.	Consumer	Single Entry	Credit only
COR	Notification of Change, or Refused Notification of Change	This Standard Entry Class Code is used by an RDFI or ODFI when originating a Notification of	Consumer or Non-Consumer	N/A	N/A

Code	Name	Description	Type	Transaction Type	Credit/Debit
		Change or Refused Notification of Change in automated format. It is also used by the ACH operator that converts paper Notifications of Change to automated format.			
CTX	Corporate Trade Exchange	Used to transfer funds between a buyer's and a seller's bank accounts.	Non-Consumer	Single or Recurring Entry	Credit/Debit
DNE (Federal Govt. Agency Use Only)	Death Notification Entry	A non-monetary entry from a Federal Government agency. Notifies the Financial Institution that the recipient of a government benefit has passed away.	Consumer	N/A	N/A
ENR	Automated Enrollment	A Non-Monetary Entry that triggers the origination of	Non-Consumer (Federal	N/A	N/A

Code	Name	Description	Type	Transaction Type	Credit/Debit
	Entry	ACH credit or debit transactions to the account holder at the DFI. The ENR process allows DFIs to transmit information to Federal Government agencies, on behalf of their account holders, that serves as enrollment for either ACH credit or debit activity.	Government Agency)		
IAT	International ACH Transaction	Credit or Debit Entry is part of a payment transaction that involves a financial agency's office not located within the territorial jurisdiction of the United States.  NOTE: This SEC code is implemented only for very specific	Consumer or Non-Consumer	Single or Recurring Entry	Credit/Debit

Code	Name	Description	Type	Transaction Type	Credit/Debit
		merchants and is not generally available.			
POS	Point of Sale payments	See <a href="#"><b>POS card transaction types</b></a>	Consumer	Single Entry	Credit/Debit
PPD	Prearranged Payment and Deposit Entry	A single or recurring credit transaction for payment of payroll, expense reimbursement, dividends, retirement, interest, etc.	Consumer	Single or Recurring Entry	Credit/Debit
RCK	Re-presented Check Entry	A re-presented check entry with security measures in place.	Consumer	Single Entry	Debit
TEL	Telephone-Initiated Entry	A single or recurring debit transaction initiated orally via the telephone.	Consumer	Single or Recurring Entry	Debit
WEB	Internet-Initiated Entry	<b>Credit</b> - A single or recurring credit transaction from the account of a natural person to the account of a natural person. Cannot be used	Consumer	Single or Recurring Entry	Credit or Debit

Code	Name	Description	Type	Transaction Type	Credit/Debit
		for business-to-consumer transactions. <b>Debit</b> - A single or recurring debit transaction initiated during a secure (minimum 128-bit encryption) internet or mobile session.	Ecommerce		

# Rejection codes

These codes are returned in the details of the `Ach.Payment.Rejected` webhook in the `postingCode` field.

Code	Description
ERR	General Error
RES	Account Restriction
NSF	Insufficient Funds
ANF	Account Not Found
CLS	Account Closed
INA	Account Inactive
DRM	Account Dormant
ESC	Account in escheatment
CHR	Account in charge-off status
STP	Stop payment active

## Payment types

This value appears in the `paymentType` field in ACH payment responses. The `paymentType` value tells you what kind of payment was sent. Do not confuse it with the `extendedDetails.paymentType` !

Type	Code	Description
Origination	0	A new payment originating from either Cross River or another bank. Most payments are of this type.
Return	1	Related to a previous origination that has been returned by the receiving bank.
DishonoredReturn	2	Related to a previous return, that has been dishonored by the receiving bank.
Contested return	3	Related to a previous origination. The receiving bank accepted the original payment but is now notifying you of information you should correct next time you send a payment to this receiver (e.g. use a different account number).
NOC	4	Notification of Change. The payment was sent with bad info but the receiving FI fixed it and sent back the information (NOC).
Refused NOC	5	

## Payment status

This value appears in the `status` field in ACH payment responses.

Status	Description
Created	We have received the payment, but have not started processing it yet. This status should only appear briefly under normal circumstances.
Pending	The payment is waiting to be batched and sent to the Federal Reserve.
Hold	Payment is being held at the moment and reviewed by our Operations Team.
Batched	The payment has been batched is a final review is being done before we send it out in a file to the Federal Reserve.
Processing	For inbound payments, we are attempting to post the payment to the receiving account. For outbound payments, the payment has been sent to the Federal Reserve, but has not posted yet. An outbound standard payment may remain in this status for a day or more. Same day payments will transition to Complete soon after Processing.
Completed	The payment has been posted and accepted by the Federal Reserve (in the case of outbound payments). This is a final status.
Rejected	Our Operations Team wasn't able to process the payment and has rejected it. In the case of inbound, the payment has been returned to the originating bank. This is a final status.
Canceled	An outbound payment has been canceled at the request of the partner. A payment may only be canceled while either pending or on hold. This is a final status.
Blocked	The CR compliance team blocked the payment

## Return codes

Sometimes an ACH payment is returned by the receiving bank. A `returnCode` value appears in the details object of the extended `Ach.Return.Received` webhook event.



Code	Description
R01	Insufficient Funds
R02	Account closed
R03	No account or unable to locate account
R04	Invalid account number
R05	Unauthorized debit to consumer account
R06	Returned per ODFI's request
R07	Authorization revoked by customer
R08	Payment stopped or stop payment on item
R09	Uncollected funds
R10	Customer advises not authorized
R11	Customer Advises Entry Not in Accordance with the Terms of the Authorization
R12	Branch sold to another DFI
R13	Invalid ACH routing number
R14	Representment payee deceased or unable to continue in that capacity
R15	Beneficiary of account holder deceased
R16	Account frozen
R17	File record edit criteria
R18	Improper effective entry date
R19	Amount field error
R20	Non-transaction account
R21	Invalid company identification
R22	Invalid individual ID number
R23	Credit entry refused by receiver
R24	Duplicate entry

Code	Description
R25	Addenda error
R26	Mandatory field error
R27	Trace number error
R28	Routing number or check digit error
R29	Corporate customer advises not authorized
R30	RDFI not participant in check truncation program
R31	Permissible return entry
R32	RDFI nonsettlement
R33	Return of XCK entry
R34	Limited participation DFI
R35	Return of improper debit entry
R36	Return of improper credit entry
R37	Source Document Presented for Payment
R38	Stop payment on source document
R39	Improper Source Document
R40	Return of ENR
R41	Invalid Transaction Code
R42	Routing No. / Check Digit Error
R43	Invalid DFI Account No.
R44	Invalid Individual ID No.
R45	Invalid Individual / Company Name
R46	Invalid Representative Payee Indicator
R47	Duplicate Enrollment
R50	State Law Affecting RCK Acceptance
R51	Ineligible / Improper Item Related to RCK

Code	Description
R52	Stop Payment on Item Related to RCK
R53	Item and RCK Presented for Payment
R73	Timely Original Return
R74	Corrected Return
R75	Return Not Duplicate
R76	No Errors Found
R77	Non-Acceptance of R62
R80	IAT Coding Error
R81	Non-Participant in IAT Program
R82	Invalid Foreign RDFI Identification
R83	Foreign RDFI Unable to Settle
R84	Not Processed by Gateway
R85	Incorrectly Coded Outbound Int'l Payment

## Dishonored returns

Code	Description
R61	Misrouted return
R62	Return of Erroneous or Reversing Debit
R63	Incorrect dollar amount
R64	Incorrect individual identification
R65	Incorrect transaction code
R66	Incorrect company identification
R67	Duplicate return
R68	Untimely return
R69	Multiple errors
R70	Permissible return entry not accepted
R71	Misrouted Dishonored Return
R72	Untimely Dishonored Return

## ACH correction codes

There are times when Cross River receives an ACH notification of change (NOC) related to an outbound payment. The `reasonCode` field in the extended `Ach.Noc.Received` webhook event provides the correction code indicating what information was wrong and needed to be changed for the payment to settle.

Code	Description
C01	Incorrect DFI Account Number
C02	Incorrect Routing Number
C03	Incorrect Routing Number and Incorrect DFI Number
C04	Incorrect Individual Name/Receiving Company Name
C05	Incorrect Transaction Code
C06	Incorrect DFI Account Number and Incorrect Transaction Code
C07	Incorrect Routing Number, Incorrect DFI Account Number, and Incorrect Transaction Code
C08	Incorrect Receiving DFI Identification (IAT only)
C09	Incorrect Identification Number
C13	Addenda Format Error
C14	Incorrect SEC Code for Outbound International Payment

## POS card transaction types

If the **secCode** value in the request is **POS**, in the `extendedDetails` object you must include a value for the `cardTransactionType`.

Type	Description
01	Purchase of goods
02	Cash
03	Return Reversal
11	Purchase Reversal
12	Cash Reversal
13	Return
21	Adjustment
99	Misc. Transaction

# IAT transaction types

When sending an IAT transaction, you must also supply a code for the transaction type in the `iatDetails.transactionTypeCode` field.

Code	Description
ANN	Annuity
BUS	Business/Commercial
DEP	Deposit
LOA	Loan
MIS	Miscellaneous
MOR	Mortgage
PEN	Pension
REM	Remittance
RLS	Rent/Lease
SAL	Salary/Payroll
TAX	Tax
ARC	Accounts Receivable Entry
BOC	Back Office Conversion Entry
MTE	Machine Transfer Entry
POP	Point-of-Purchase Entry
POS	Point-of-Sale Entry
RCK	Re-presented Check Entry
TEL	Telephone-Initiated Entry
WEB	Internet-Initiated Entry

## Nacha field mappings

This table presents field names as they appear in a Nacha record and how they appear in CR APIs.

Nacha Record	Nacha Field	Cross River API Field
Batch	Company Name	Originator.Name
Batch	Company Discretionary Data	Originator.Data
Batch	Company Identification	Originator.Identification
Batch	Standard Entry Class Code	SecCode
Batch	Company Entry Description	Description
Batch	Company Descriptive Date	N/A
Batch	Effective Entry Date	EffectiveDate
Batch	Settlement Date	SettlementDate
Batch	Originator Status Code	N/A
Batch	Originating DFI Identification	Originator.RoutingNumber
Entry	Transaction Code	TransactionType / Receiver.AccountType
Entry	Receiving DFI Identification	Receiver.RoutingNumber
Entry	DFI Account Number	Receiver.AccountNumber
Entry	Amount	Amount
Entry	Individual Identification Number	Receiver.Identification
Entry	Individual Name	Receiver.Name



Nacha Record	Nacha Field	Cross River API Field
Entry	Discretionary Data	Receiver.Data /ExtendedDetails.PaymentType (WEB/TEL only)
Entry	Trace Number	TraceNumber
Addenda (05)	Payment Related Information	Addenda (only one informational addenda record is supported)

## 6.4.4. Error codes

---

Code	Description
1000	General exception
1001	Payment required
2000	General exception
2001	Invalid payment status
2002	Invalid posting status
2003	Payment cannot be canceled
2004	Account not found
2005	Payment must be outbound
2006	Payment must be inbound
2007	Payment must be an inbound origination
2008	Payment must be status completed or rejected to be corrected
2009	Invalid change code
2010	Payment must be an inbound return
2011	Payment must be an origination
2012	Payment must be status completed or rejected to be returned
2013	Payment must be status completed or rejected to be dishonored
2014	Invalid return code
2015	Invalid dishonored return code
2016	Original payment not found
2017	Cannot link to same payment ID
2018	Payment type must be return or notification of change
2019	Original payment must be completed
2020	Payment must be on hold to request a rescan
2021	No scan lists are configured for account
2022	Scan already pending

Code	Description
2023	Previous payment not found
2024	Previous payment must be a completed outbound origination
2025	Receiver account not found
2026	Active holds found
2027	Posting account cannot be changed due to current payment status
2028	Posting status must be pending or failed to change posting account
2029	Posting status must be failed to attempt a retry
2030	Originator profile address missing or invalid
2200	Batch not authorizing
2201	Batch requires one or more payment
2202	No filters were found on the request

## 6.5. Wires

As part of the industry-wide transition to ISO 20022, Fedwire will adopt this new messaging standard on **July 14th, 2025**.

We are introducing new **API validations** across different wire message types. These validations ensure compliance, ensure data accuracy, and improve processing accuracy. See [IMPORTANT: Wires API update](#) for details on the new validations launched in April 2025 to our Sandbox environment.

### Wires concepts

Use wire transfers for both domestic and international USD payments. You or your customer can transfer money electronically to a recipient and a recipient can access the money, typically the same day.

Wire transfers go through [Fedwire®](#), operated by the U.S. Federal Reserve.

At Cross River, both inbound and outbound international wire transfers go through a domestic intermediary bank. To send or receive international wires in USD, you *must* contact the beneficiary bank to get the needed wire transfer information.

[Get started](#) with Cross River APIs.

Once you have access to our [sandbox](#) you can access these API endpoints in the `wires` module through [Swagger](#).

### GET calls

- Add query parameters to filter the response the API returns to general GET calls.
- Use [Pagination](#) to control presentation of your results.

This table presents APIs that we describe in detail.

Action	API Call	Description
<a href="#"><u>Wire transfer</u></a>	POST /wires/v1/payments	Sends a single wire transfer payment
<a href="#"><u>Cancel transfer</u></a>	POST /wires/v1/payments/{id}/cancel	Cancels a wire transfer payment
<a href="#"><u>Drawdown request</u></a>	POST /wires/v1/payments/corporate-drawdown-requests	Requests an inbound payment by wire transfer
<a href="#"><u>Drawdown response</u></a>	POST /wires/v1/payments/{id}/drawdown-responses	Responds to a request for a payment by wire transfer

## Related topics

- [Webhook events](#)
- [Request and response codes](#)
- [Error codes](#)

## Tutorials

- [Send a wire payment](#): Originate a wire payment.
- [Send a drawdown request](#): Request a wire payment from an external account.
- [Respond to a drawdown request](#): Respond to a request for payment sent from an external account.
- [Simulate an inbound wire](#): Test the system with our simulation endpoint. You can also use this endpoint to fund a test account.

## 6.5.1. IMPORTANT: Wires API update

---

As part of the industry-wide transition to ISO 20022, Fedwire will adopt this new messaging standard on **July 14, 2025**. This migration enhances financial messaging, improves data quality, and increases efficiency across payment networks.

### New API Validations and Default Values

As part of our Wires ISO 20022 implementation, we are introducing new **API validations** across different wire message types. These validations ensure compliance, ensure data accuracy, and improve processing accuracy. Below is a breakdown of the new validations by message type, along with default values applied if certain fields are not populated.

#### 1. Originate Customer Transfers (pacs.008)

**Endpoint:** `POST /v1/payments`

##### Validation Rules

- **Originator Requirement**
  - If Originator is null or empty, the system will use the configured account/product originator.
  - If the configured account/product originator contains any of the following ID codes, the request will be rejected:
    - "U" (CHIPS Identifier)
    - "C" (CHIPS Participant)
    - "F" (Fed Routing Number)
  - **Error Message:** `Account or product configuration error.`
- **Intermediary FI and Beneficiary FI Uniqueness**
  - If both are specified with the same identifier, the request will be rejected.
  - **Error Message:** `IntermediaryFi and BeneficiaryFi cannot be identical.`
- **Beneficiary Validation**

- Must be validated as a person or organization.
  - ***Errors follow the person/organization validation rules.***
- **Beneficiary FI Validation**
  - Must be validated as a financial institution.
  - ***Errors follow the financial institution validation rules.***
- **Intermediary FI Validation**
  - Must be validated as a financial institution.
  - ***Errors follow the financial institution validation rules.***

## Default Values

- **Beneficiary FI (if not explicitly provided — no ID Code or Identifier specified) (Note: Effective starting July 14, only when ISO format is in use)**
  - Derived from the Receiver Routing Number:
    - ID Code: "F"
    - Identifier: Routing number
    - Name: Participant's customer name
    - Address1: Participant's city and state

## 2. Originate Drawdown Requests (pain.013)

**Endpoint:** POST /v1/payments/corporate-drawdown-requests

### Validation Rules

- **Beneficiary FI and Intermediary FI Not Allowed**
  - If either field is populated, the request will be rejected.
  - **Error Message:** {EntityName} not available with ISO format.

## 3. Originate Drawdown Responses (pain.014)

**Endpoint:** POST /v1/payments/{id}/drawdown-responses



## Validation Rules

- **Originator BIC Format**
  - If the Originator Id Code is "B" (SWIFT BIC Code) and the Identifier is not in valid BIC format, the request will be rejected.
  - **Error Message:** `OriginatorIdentifier is not a valid BIC.`

## Default Values

- **Beneficiary FI (if not provided in original DRC) (Note: Effective starting July 14, only when ISO format is in use)**
  - Derived from the Receiver Routing Number:
    - ID Code: "F"
    - Identifier: Receiver's routing number
    - Name: Receiver's customer name
    - Address1: Receiver's city and state
- **Originator FI (if not provided in original DRC) (Note: Effective starting July 14, only when ISO format is in use)**
  - Derived from the Sender Routing Number:
    - ID Code: "F"
    - Identifier: Sender's routing number
    - Name: Sender's customer name
    - Address1: Sender's city and state

## 4. Reverse Payment (pacs.004)

**Endpoint:** `POST /v1/payments/{id}/reversals`

## Validation Rules

- **Reversible Payment Types Only (Note: Effective starting July 14, only when ISO format is in use)**
  - If the referenced payment is not a transfer or drawdown response, the request will be rejected.

- **Error Message:** Payment type cannot be reversed.

## Default Values

- **Originator FI (if not provided in original payment - customer transfer or drawdown response) (Note: Effective starting July 14, only when ISO format is in use)**
  - Derived from the Sender Routing Number of the original payment - transfer or drawdown response:
    - ID Code: "F"
    - Identifier: Sender's routing number
    - Name: Sender's customer name
    - Address1: Sender's city and state

## 5. Service Messages

**Endpoint:** POST /v1/payments/{id}/service-message

## Validation Rules

- **Type Code Defaulting**
  - If the type code is not specified, it will default to:
    - 1090 – Customer Transfer Service Message
- **Automatic Conversion of Type Code**
  - If the type code is 1090 (Customer Transfer Service Message) or 1690 (Bank Transfer Service Message), and the referenced payment is an ***Inbound Investigation Request***, the type code will be automatically converted to:
    - 1099 – Customer Transfer Investigation Response
    - 1699 – Bank Transfer Investigation Response
- **Service Message Reclassification**
  - Service messages in the FAIM model have been replaced with ISO-equivalent message types.
  - FAIM type codes are mapped to ISO messages as follows.
- **Investigation Requests (camt.110) — FAIM type codes: 1090 , 1690**
  - **Referenced Payment Must Be Investigable**

- If the referenced payment is not one of the following:
    - pacs.008 – Customer Credit Transfer
    - pacs.004 – Payment Return
    - pain.013 – Drawdown Request
    - pain.014 – Drawdown Response
  - **Error Message:** Payment type cannot be investigated.
- **Referenced Payment Must Be Transmitted**
  - If the referenced payment does not have an IMAD (i.e., it has not been confirmed as transmitted), the request will be rejected.
  - **Error Message:** Cannot investigate untransmitted payments.
- **Service Message Line Requirement**
  - If all ServiceMessageLine\* fields are null or empty, the request will be rejected.
  - **Error Message:** At least one line of text is required for the requested type code.
- **Investigation Responses (camt.111) — FAIM type codes: 1099 , 1699**
  - **Referenced Payment Must Be an Investigation Request**
    - If the referenced payment is not an investigation request, the request will be rejected.
    - **Error Message:** Source payment is not an investigation request.
  - **Service Message Line Requirement**
    - If all ServiceMessageLine\* fields are null or empty, the request will be rejected.
    - **Error Message:** At least one line of text is required for the requested type code.
- **Return Requests (camt.056) — FAIM type codes: 1001 , 1007**
  - **Referenced Payment Must Be a Transfer**
    - If the referenced payment is not a customer transfer or drawdown response, the request will be rejected.
    - **Error Message:** Payment type cannot be reversed.
  - **Referenced Payment Must Be Outbound**
    - If the referenced payment was not an outbound payment, the request will be rejected.

- **Error Message:** Direction not outbound.
- **Service Message Line Requirement**
  - If all ServiceMessageLine\* fields are null or empty, the request will be rejected.
  - **Error Message:** At least one line of text is required for the requested type code.

## 6. Person/Organization Validations

- **Name or Identifier Required**
  - Both cannot be null/empty.
  - **Error Message:** {EntityName} must include either a Name or an Identifier.
- **Allowed Id Codes**
  - Must be one of:
    - "1" (Passport)
    - "2" (TIN)
    - "3" (Driver's License)
    - "4" (Alien Registration)
    - "5" (Corporate ID)
    - "9" (Other ID)
    - "B" (SWIFT BIC)
    - "D" (Demand Deposit Account)
  - **Error Message:** {EntityName} IdCode must be a valid customer code.
- **Name Requirement for Non-BIC**
  - If Id Code is not "B" or "T" and Name is null/empty:
  - **Error Message:** {EntityName} must include a Name if IdCode is not a BIC.
- **BIC Format**
  - If Id Code is "B" or "T", Identifier must be a valid BIC.
  - **Error Message:** {EntityName} Identifier must be a valid BIC format.

## 7. Financial Institution Validations

- **Name or Identifier Required**
  - Both cannot be null/empty.
  - **Error Message:** {EntityName} must include either a Name or an Identifier.
- **Allowed Id Codes**
  - Must be one of:
    - "U" (CHIPS Identifier)
    - "C" (CHIPS Participant)
    - "D" (Demand Deposit Account)
    - "F" (Fed Routing Number)
    - "B" (SWIFT BIC)
    - "T" (SWIFT BIC or BEI)
  - **Error Message:** {EntityName} IdCode must be a valid financial institution code.
- **Name Requirement for Non-BIC**
  - If Id Code is not "B" or "T" and Name is null/empty:
  - **Error Message:** {EntityName} must include a Name if IdCode is not a BIC.
- **Address1 Requirement for Non-BIC**
  - If Id Code is not "B" or "T" and Address1 is null/empty:
  - **Error Message:** {EntityName} must include an Address if IdCode is not a BIC.
- **BIC Format**
  - If Id Code is "B" or "T", Identifier must be a valid BIC.
  - **Error Message:** {EntityName} Identifier must be a valid BIC format.
- **Routing Number Format**
  - If Id Code is "F" and Identifier is invalid:
  - **Error Message:** {EntityName} Identifier is not a valid Fed Routing Number.



## 6.5.2. **APIs**

### 6.5.2.1. **Wire transfer**

---

## Endpoint: /wires/v1/payments

Sends a single wire transfer payment.

POST

https://sandbox.crbcos.com/wires/v1/payments



Request

Response

### BODY PARAMETERS

**accountNumber**

String

required

Originator account number

**originator** ▶

Object

optional

Originator of the payment

When calling the API, the originator on the outbound wire is either Cross River by default or the name of the account, depending on the configuration of your product.

**businessFunctionCode**

String

required

The business purpose of the wire transfer. The most common type is customer wire transfer (CTR) where the beneficiary isn't a bank. See the full list of codes and descriptions.

**receiverRoutingNumber**

String

required

The 9-digit Fedwire routing number of the financial institution receiving the wire drawdown request. The ABA directory provides financial institution routing numbers.

**beneficiary** ▶

Object

required

Entity receiving the funds



**beneficiaryFi ▶**

Object

optional

Bank receiving the wire transfer

**intermediaryFi ▶**

Object

optional

Intermediary bank involved in the transaction. Required when sending an international payment.

**beneficiaryReference**

String

optional

Optional field for additional information. 16 characters maximum.

**originatorToBeneficiary1**

String

optional

Field for additional information from the originator to the beneficiary.

35 characters maximum.

You can add information in up to 4 originator to beneficiary fields (originatorToBeneficiary1, originatorToBeneficiary2, originatorToBeneficiary3, originatorToBeneficiary4).

**amount**

Integer

required

Dollar amount of wire transfer in positive integral cents. For example, write \$1.00 as 100.

**purpose**

String

required

Reason for the wire transfer, such as "daily settlement." Also use this field if you submit multiple wires for the same amount, such as "1 of 2, 2 of 2." This data is not included with the outgoing FedWire message. Maximum 50 characters.

**clientIdentifier**

String

optional

Use this attribute to add your own unique identifying string to a payment call or COS record. This attribute is useful for idempotency purposes.



Curl



Node.js



Python



Ruby



Go



```
curl -L 'https://sandbox.crbcos.com/wires/v1/payments' ^  
-d '{  
  "accountNumber": "2955057589",  
  "businessFunctionCode": "CTR",  
  "receiverRoutingNumber": "026009593",  
  "beneficiary": {  
    "idCode": "D",  
    "identifier": "316840684",  
    "name": "Stewie Griffin",  
    "address1": "31 Spooner St",  
    "address2": "Quahog RI 33030",  
    "Address3": "US"  
  },  
  "beneficiaryReference": "Gift",  
  "originatorToBeneficiary1": "Happy Birthday",  
  "amount": 15000,  
  "purpose": "Testing",  
  "clientIdentifier": "05d73490-65ac-4725-98b6-cbf8252e52fe"  
}'  
  
,
```

## Responses

200

403

```
{  
  "errors": [  
    {  
      "code": 3001,  
      "message": "Origination payment amount exceeds user permission"  
    }  
  ]  
}
```

## 6.5.2.2. Cancel transfer

---

## Endpoint: /wires/v1/payments/{id}/cancel

POST

https://sandbox.crbcos.com/wires/v1/payments/{id}/cancel



Request

Response

### PATH PARAMS

**id**

String

**required**

The payment ID. First appears in the initial response to the originate a payment request. This ID is in GUID format.



Curl



Node.js



Python



Ruby



Go



```
curl -L -g 'https://sandbox.crbcos.com/wires/v1/payments/{id}/cancel' ^  
-H 'Idempotency-key: 0b700631-e844-464d-a747-ecb65c15d2c5' ^  
-d '  
  
'
```

Responses

● 200





### 6.5.2.3. Drawdown request

---



## Endpoint: /wires/v1/payments/corporate-drawdown-requests

Request a payment from an external bank, to be sent by wire.

POST

<https://sandbox.crbcos.com/wires/v1/payments/corporate-drawdown-requests> 

Request

Response

### BODY PARAMETERS

**accountNumber**

String

**required**

The account number being used to originate the outbound wire drawdown request .

**originator ▶**

Object

**optional**

Originator of the payment.

When calling the API, the originator on the outbound wire is either Cross River by default or the name of the account, depending on the configuration of your product.

**receiverRoutingNumber**

String

**required**

The 9-digit Fedwire routing number of the financial institution receiving the wire drawdown request. The ABA directory provides financial institution routing numbers.

**drawdownCreditAccountNumber**

String

**required**

The 9-digit Fedwire routing number of the financial institution originating the wire drawdown request. The ABA directory provides financial institution routing numbers.

**drawdownDebitAccount ▶**

Object

**required**

Details of the account from which the funds will be debited.

**beneficiary ▶**

Object

**required**

For a drawdown request, this is the same as the originator object.

**beneficiaryReference**

String

**optional**

Optional field for additional information. 16 characters maximum.

**originatorToBeneficiary1**

String

**optional**

One of 4 fields, 35 characters each, to add additional information to be sent to the beneficiary.

**originatorToBeneficiary2**

String

**optional**

One of 4 fields, 35 characters each, to add additional information to be sent to the beneficiary.

**originatorToBeneficiary3**

String

**optional**

One of 4 fields, 35 characters each, to add additional information to be sent to the beneficiary.

**originatorToBeneficiary4**

String

**optional**

One of 4 fields, 35 characters each, to add additional information to be sent to the beneficiary.

**debitDrawdownAdviceCode**

String

**optional**

Additional wire information. One of:

- HLD (hold)
- LTR (letter)
- PHN (phone)

**debitDrawdownInformation1** String optional

Field for additional information about the drawdown debit account. 26 characters maximum.

**debitDrawdownInformation2** String optional

Field for additional information about the drawdown debit account. 33 characters maximum.

**debitDrawdownInformation3** String optional

Field for additional information about the drawdown debit account. 33 characters maximum.

**debitDrawdownInformation4** String optional

Field for additional information about the drawdown debit account. 33 characters maximum.

**debitDrawdownInformation5** String optional

Field for additional information about the drawdown debit account. 33 characters maximum.

**receiverFilnformation1** String optional

Field for additional information about the receiver financial institution. 30 characters maximum.

**receiverFilnformation2** String optional

Field for additional information about the receiver financial institution. 33 characters maximum.

**receiverFilInformation3**      String      optional

Field for additional information about the receiver financial institution. 33 characters maximum.

**receiverFilInformation4**      String      optional

Field for additional information about the receiver financial institution. 33 characters maximum.

**receiverFilInformation5**      String      optional

Field for additional information about the receiver financial institution. 33 characters maximum.

**fiToFilInformation1**      String      optional

Field for additional information to be sent from one financial institution to another. 35 characters maximum.

**fiToFilInformation2**      String      optional

Field for additional information to be sent from one financial institution to another. 35 characters maximum.

**fiToFilInformation3**      String      optional

Field for additional information to be sent from one financial institution to another. 35 characters maximum.

**fiToFilInformation4**      String      optional

Field for additional information to be sent from one financial institution to another. 35 characters maximum.

**fiToFilInformation5**                      String                      optional

Field for additional information to be sent from one financial institution to another.  
35 characters maximum.

**fiToFilInformation6**                      String                      optional

Field for additional information to be sent from one financial institution to another.  
35 characters maximum.

**amount**                                      Integer                      required

Dollar amount of wire transfer in positive integral cents. For example, write \$1.00 as 100.

**purpose**                                      String                      required

Reason for the wire transfer. For internal use only. The data is not included with the outgoing FedWire message.

**clientIdentifier**                              String                      optional

Use this attribute to add your own unique identifying string to a payment call or COS record. This attribute is useful for idempotency purposes.



Curl



Node.js



Python



Ruby



Go



```
curl -L 'https://sandbox.crbcos.com/wires/v1/payments/corporate-drawdown-r
-H 'Content-Type: application/json' ^
-H 'Authorization: Bearer eyJhbGciOiJSUzI1NiIsImtpZCI6IjdQREJ4dGVldl9IMXhp
-d '{
  "accountNumber": "",
  "originator":
  {
    "idCode": "D",
    "identifier": "",
    "name": "PMR Corp",
    "address1": "10431 Howe Drive",
    "address2": "Shawnee Mission KS 66206",
    "address3": "US"
  },
  "receiverRoutingNumber": "026009593",
  "drawdownCreditAccountNumber": "021214891",
  "beneficiaryFi":
  {
    "idCode": "D",
    "identifier": "021214891",
    "name": "Cross River Bank",
    "address1": "400 KELBY STREET",
    "address2": "Fort Lee NJ 07024",
    "address3": "US"
  },
  "drawdownDebitAccount":
  {
    "idCode": "D",
    "identifier": "777555333",
    "name": "Lindberg Incorporated",
    "address1": "5 Stop St",
    "address2": "New York NY 52555",
    "address3": "US"
  },
  "beneficiary":
  {
    "idCode": "D",
    "identifier": "2714035231",
    "name": "PMR Corp",
    "address1": "10431 Howe Drive",
    "address2": "Shawnee Mission KS 66206",
    "address3": "US"
  },
  "beneficiaryReference": "testref",
  "originatorToBeneficiary1": "1string",
```

## Responses

● 200





```
{
  "id": "000000000-0000-0000-0000-000000000000",
  "accountNumber": "string",
  "coreTransactionId": "000000000-0000-0000-0000-000000000000",
  "referenceId": "string",
  "clientBatchId": "000000000-0000-0000-0000-000000000000",
  "clientBatchSequence": 0,
  "fedBatchId": "000000000-0000-0000-0000-000000000000",
  "fedBatchSequence": 0,
  "direction": "Inbound",
  "paymentType": "Transfer",
  "source": "Phone",
  "status": "Created",
  "posting": "Pending",
  "rejectionReason": "OfacCancelUnableToDetermine",
  "blockedReason": "OfacMatch",
  "partnerAuthorization": "NotRequired",
  "authorizedAt": "2022-12-28T08:03:53.686Z",
  "authorizedById": "string",
  "authorizedByName": "string",
  "partnerAuthorization2": "NotRequired",
  "authorizedAt2": "2022-12-28T08:03:53.686Z",
  "authorizedById2": "string",
  "authorizedByName2": "string",
  "amount": 0,
  "currency": "string",
  "purpose": "string",
  "imad": "string",
  "omad": "string",
  "businessFunctionCode": "string",
  "typeCode": "string",
  "senderRoutingNumber": "string",
  "senderName": "string",
  "senderReference": "string",
  "receiverRoutingNumber": "string",
  "receiverName": "string",
  "originatingFi": {
    "idCode": "string",
    "identifier": "string",
    "name": "string",
    "address1": "string",
    "address2": "string",
    "address3": "string"
  },
  "originator": {
```

```
"idCode": "string".
```

#### 6.5.2.4. Drawdown response

---

## Endpoint: /wires/v1/payments/{id}/drawdown-responses

Responds to a wires drawdown request by sending payment.

**POST**

<https://sandbox.crbcos.com/wires/v1/payments/{id}/drawdown-responses> 

**Request**

Response

### QUERY PARAMETERS

**id** String **required**

The payment ID of the drawdown request. This is in GUID format.

### BODY PARAMETERS

**originator** ▶ Object **required**

Details of the party sending the wire

**beneficiaryReference** String **optional**

Optional field for additional information. 16 characters maximum.

**originatorToBeneficiaryLine1** String **optional**

One of 4 fields, 35 characters each, to add additional information to be sent to the beneficiary.

**originatorToBeneficiaryLine2** String **optional**

One of 4 fields, 35 characters each, to add additional information to be sent to the beneficiary.

**originatorToBeneficiaryLine3** String **optional**

One of 4 fields, 35 characters each, to add additional information to be sent to the beneficiary.

**originatorToBeneficiaryLine4** String optional

One of 4 fields, 35 characters each, to add additional information to be sent to the beneficiary.



Curl



Node.js



Python



Ruby



Go



```
curl -L 'https://sandbox.crbcos.com/wires/v1/payments/61c194b3-6405-4b7f-b
-H 'Content-Type: application/json' ^
-H 'Authorization: Bearer token' ^
-d '{
  "originator": {
    "idCode": "D",
    "identifier": "2955057589",
    "name": "PMR Corp",
    "address1": "10431 Howe Drive",
    "address2": "Shawnee Mission KS 66206",
    "address3": "US"
  },
  "beneficiaryReference": "testref",
  "originatorToBeneficiary1": "1string",
  "originatorToBeneficiary2": "2string",
  "originatorToBeneficiary3": "3string",
  "originatorToBeneficiary4": "4string"
}'
```

Responses

● 200







# 6.5.3. Webhook events

When you work with Cross River accounts, cards and payments, you register for specific webhooks which have a specific event format.

This table presents common `wires` webhook events that are described in detail below.

Event Name	Description
<u>Wire.Payment.Sent</u>	Outbound wire has been transmitted to the Federal Reserve and has been successfully acknowledged. IMAD number is now available
<u>Wire.Payment.Received</u>	Inbound wire has been received from another bank
<u>Wire.Payment.Rejected</u>	Outbound wire could not be processed due to compliance reasons or was rejected by the Federal Reserve
<u>Wire.Payment.Canceled</u>	Outbound wire was canceled at the partner's request

## Event examples

### Wire.Payment.Sent





```
{
  "id": "bfecb86b-a76f-42bf-b354-b25901132160",
  "eventName": "Wire.Payment.Sent",
  "status": "Pending",
  "partnerId": "ba9d0234-26e9-43a2-9070-aedd0108eadb",
  "createdAt": "2025-01-02T11:41:43.147-05:00",
  "resources": [
    "wires/v1/payments/b02e7f6b-a9a5-4de2-a291-b25901119db5"
  ],
  "details": [
    {
      "paymentId": "b02e7f6b-a9a5-4de2-a291-b25901119db5",
      "accountNumber": "2273475422",
      "direction": "Outbound",
      "imad": "20250102SIMULATE000121",
      "omad": "20250102SIMULATE000122",
      "paymentType": "Transfer",
      "purpose": "Invoice Payment",
      "amount": "10000",
      "clientIdentifier": null,
      "originatingFiName": "CRB 1270741",
      "originatingFiIdentifier": "021214891",
      "originatorName": "PETER GRIFFIN",
      "originatorIdentifier": "2273475422",
      "originatorAddress1": "31 SPOONER ST",
      "originatorAddress2": "",
      "originatorAddress3": "QUAHOG RI 33156 US",
      "beneficiaryFiName": null,
      "beneficiaryFiIdentifier": null,
      "beneficiaryName": "Bob Smith",
      "beneficiaryIdentifier": "77777777701",
      "beneficiaryAddress1": "123 Maple Lane",
      "beneficiaryAddress2": "Suite 123",
      "beneficiaryAddress3": "New York, NY 10025",
      "beneficiaryReference": "INV# 123",
      "senderReference": "0H1806M7C23",
      "originatorToBeneficiary1": "THANK",
      "originatorToBeneficiary2": "YOU",
      "originatorToBeneficiary3": "PAID",
      "originatorToBeneficiary4": "IN FULL",
      "coreTransactionId": "61829465-5dd1-4a99-94ce-b25901119db5",
      "senderRoutingNumber": "021214891"
    }
  ]
}
```

**Wire.Payment.Received**





```
{
  "id": "cc247ea2-bde2-4072-afce-b2590113ae11",
  "eventName": "Wire.Payment.Received",
  "status": "Pending",
  "partnerId": "ba9d0234-26e9-43a2-9070-aedd0108eadb",
  "createdAt": "2025-01-02T11:43:43.203-05:00",
  "resources": [
    "wires/v1/payments/aa8b18ca-5bc4-485b-a28d-b2590113098f"
  ],
  "details": [
    {
      "paymentId": "aa8b18ca-5bc4-485b-a28d-b2590113098f",
      "accountNumber": "2273475422",
      "direction": "Inbound",
      "imad": "2025010199999123000001",
      "omad": null,
      "paymentType": "Transfer",
      "purpose": "2012510199999123000002",
      "amount": "7500",
      "clientIdentifier": null,
      "originatingFiName": null,
      "originatingFiIdentifier": null,
      "originatorName": "JOHN SMITH",
      "originatorIdentifier": "987654321",
      "originatorAddress1": "123 MAPLE LANE",
      "originatorAddress2": "NEW YORK NY 10025 US",
      "originatorAddress3": null,
      "beneficiaryFiName": null,
      "beneficiaryFiIdentifier": null,
      "beneficiaryName": "Peter Griffin",
      "beneficiaryIdentifier": "2273475422",
      "beneficiaryAddress1": "TEST ADDRESS 1",
      "beneficiaryAddress2": "TEST ADDRESS 2",
      "beneficiaryAddress3": null,
      "beneficiaryReference": "TEST",
      "senderReference": "PAYMENT",
      "originatorToBeneficiary1": "THANK YOU",
      "originatorToBeneficiary2": "FOR THE GIFT",
      "originatorToBeneficiary3": null,
      "originatorToBeneficiary4": null,
      "coreTransactionId": "8c635968-8db0-4c56-a87a-b259011319cd",
      "senderRoutingNumber": "021000021"
    }
  ]
}
```



**Wire.Payment.Rejected**





```
{
  "id": "e9d56456-5524-406b-a97a-b25901127180",
  "eventName": "Wire.Payment.Rejected",
  "status": "Pending",
  "partnerId": "ba9d0234-26e9-43a2-9070-aedd0108eadb",
  "createdAt": "2025-01-02T11:39:13.073-05:00",
  "resources": [
    "wires/v1/payments/dd271f0b-506e-4e4f-9e75-b259011233fc"
  ],
  "details": [
    {
      "paymentId": "dd271f0b-506e-4e4f-9e75-b259011233fc",
      "accountNumber": "182026999942",
      "direction": "Outbound",
      "imad": null,
      "omad": null,
      "paymentType": "Transfer",
      "purpose": "Invoice Payment",
      "amount": "10000",
      "clientIdentifier": null,
      "originatingFiName": "CRB 1270741",
      "originatingFiIdentifier": "021214891",
      "originatorName": "PETER GRIFFIN",
      "originatorIdentifier": "182026999942",
      "originatorAddress1": "31 SPOONER ST",
      "originatorAddress2": "",
      "originatorAddress3": "QUAHOG RI 33156 US",
      "beneficiaryFiName": null,
      "beneficiaryFiIdentifier": null,
      "beneficiaryName": "Bob Smith",
      "beneficiaryIdentifier": "77777777701",
      "beneficiaryAddress1": "123 Maple Lane",
      "beneficiaryAddress2": "Suite 123",
      "beneficiaryAddress3": "New York, NY 10025",
      "beneficiaryReference": "INV# 123",
      "senderReference": "9686J5S0299",
      "originatorToBeneficiary1": "THANK",
      "originatorToBeneficiary2": "YOU",
      "originatorToBeneficiary3": "PAID",
      "originatorToBeneficiary4": "IN FULL",
      "coreTransactionId": "44faaefb-44d4-4f8c-a76e-b259011233fc",
      "senderRoutingNumber": "021214891",
      "rejectionReason": "Other"
    }
  ]
}
```

]

**Wire.Payment.Canceled**





```
{
  "id": "57c13064-90c7-4f41-952e-b2590111c1a1",
  "eventName": "Wire.Payment.Canceled",
  "status": "Pending",
  "partnerId": "ba9d0234-26e9-43a2-9070-aedd0108eadb",
  "createdAt": "2025-01-02T11:36:42.987-05:00",
  "resources": [
    "wires/v1/payments/7d5fcc6e-d881-433b-b22e-b2590111abb9"
  ],
  "details": [
    {
      "paymentId": "7d5fcc6e-d881-433b-b22e-b2590111abb9",
      "accountNumber": "2330604998",
      "direction": "Outbound",
      "imad": null,
      "omad": null,
      "paymentType": "Transfer",
      "purpose": "Invoice Payment",
      "amount": "10000",
      "clientIdentifier": null,
      "originatingFiName": "CRB 1270741",
      "originatingFiIdentifier": "021214891",
      "originatorName": "PETER GRIFFIN",
      "originatorIdentifier": "2330604998",
      "originatorAddress1": "31 SPOONER ST",
      "originatorAddress2": "",
      "originatorAddress3": "QUAHOG RI 33156 US",
      "beneficiaryFiName": null,
      "beneficiaryFiIdentifier": null,
      "beneficiaryName": "Bob Smith",
      "beneficiaryIdentifier": "77777777701",
      "beneficiaryAddress1": "123 Maple Lane",
      "beneficiaryAddress2": "Suite 123",
      "beneficiaryAddress3": "New York, NY 10025",
      "beneficiaryReference": "INV# 123",
      "senderReference": "Q3L3100J56F",
      "originatorToBeneficiary1": "THANK",
      "originatorToBeneficiary2": "YOU",
      "originatorToBeneficiary3": "PAID",
      "originatorToBeneficiary4": "IN FULL",
      "coreTransactionId": "a51dce63-c024-4c67-a3ce-b2590111abb9",
      "senderRoutingNumber": "021214891"
    }
  ]
}
```





# 6.5.4. Request and response codes

## Business function

Wire payments can be used to send funds or non-value messages to other financial institutions. The type of message is defined by the `businessFunctionCode`.

Code	Description
CTR	Customer transfer
SVC	Service message (non-value) between financial institutions

## ID

In Wires calls, the `idCode` field in `originator`, `beneficiary` and various financial institution objects, specifies the type of value in the `identifier` field that follows it.

Code	Description	Valid For
B	SWIFT Bank Identifier Code (BIC)	<code>beneficiary</code> Financial Institution Objects
D	Demand Deposit Account (DDA) Number	<code>originator</code> <code>beneficiary</code> Financial Institution Objects
F	Fed Routing Number	Financial Institution Objects
1	Passport Number	<code>beneficiary</code>
2	Tax Identification Number	<code>beneficiary</code>
3	Driver's License Number	<code>beneficiary</code>
4	Alien Registration Number	<code>beneficiary</code>
5	Corporate Identification	<code>beneficiary</code>
9	Other Identification	<code>beneficiary</code>

# Type

The type code value provides information about the type of wire transfer sent. This value appears in the `typeCode` field in the response to wires payment calls, and is combined with a subtype code (see the subtype code table below).

Code	Description
10	<b>Fund transfer</b> A transfer of funds in which the sender and/or receiver may be a bank or a third party, such as a bank customer.
15	<b>Foreign transfer</b> A transfer of funds to or from a foreign central bank or government or international organization with an account at the Federal Reserve Bank of New York.
16	<b>Settlement transfer</b> A transfer of funds between Fedwire Funds Service participants.

# Subtype

A subtype code value combines with the type code value in the `typeCode` field of Wires payment responses to provide details about the transfer.

Code	Description
00	Basic funds transfer
01	Request for Reversal
02	Reversal of Transfer
07	Request for Reversal of a Prior Day Transfer
08	Reversal of a Prior Day Transfer
31	Request for Credit (Drawdown)
32	Funds Transfer Honoring a Request for Credit (Drawdown)
33	Refusal to Honor a Request for Credit (Drawdown)
90	Service Message

## Payment statuses

See the payment status in the `status` field of Wires payment responses.

Status	Description
Created	CR received the payment, but hasn't yet started processing it. In most cases, this status appears only briefly.
Pending	The payment is waiting to be batched and sent to the Federal Reserve.
Hold	Payment is being held and reviewed by our Ops Team, often as a result of awaiting a response from an OFAC scan. When a hold is placed, funds are removed from the available balance in the account. A hold ensures that any subsequent payment won't cause this payment to fail.
Batched	The payment was batched and is in final review by our Operations Team before we send it in a file to the Federal Reserve.
Processing	For inbound payments, we are posting it to the receiving account. For outbound payments, the payment was sent to the Federal Reserve, but hasn't yet been accepted by them.
Completed	The outbound payment was posted and accepted by the Federal Reserve. This is a final status.
Canceled	An outbound payment was canceled at partner request. A payment can only be canceled while in the pending or hold statuses. A payment can be canceled after it's pending, only if it hasn't been executed. This is a final status.
Rejected	Our Operations Team wasn't able to process the payment and rejected it. In the case of an inbound payment, the payment was returned to the originating bank. This is a final status.
Reversal	In some cases, a payment can be reversed after it was executed.

## 6.5.5. Error codes

---

Code	Description
1000	General exception
1001	Config required
1002	InboundCountryWhitelist cannot add a country that is blacklisted
1003	OutboundCountryWhitelist cannot add a country that is blacklisted
2000	General exception
2001	Payment not found
2002	Invalid payment status
2003	Direction not inbound
2004	Direction not outbound
2005	Posting account cannot be changed as posting completed
2006	Acknowledgment required
2007	Originator account not found
2008	Origination not enabled for account
2009	Custom originator data not allowed for account
2010	Origination not allowed from a restricted account
2011	Receiver routing number not found
2012	Source wire not found
2013	Source wire message unavailable
2014	Service messages cannot be reversed
2015	Payment type cannot be reversed
2016	Payment must be completed before it can be reversed
2017	Only inbound payments can be reversed
2018	Invalid source business function code
2019	Payment must be completed before it can be responded to
2020	Only inbound payments can be responded to

Code	Description
2021	Original wire not found
2022	Beneficiary account not found
2023	Account not found
2024	Posting already completed
2025	Posting must be failed to attempt a retry
2026	Payment already acknowledged
2027	International origination not enabled for account
2028	Requested amount exceeded maximum threshold
2029	Drawdown credit account number does not match originator account number
2030	Payment type cannot be investigated
2031	Payment type not transfer
2032	Drawdown credit account number is not a valid routing number.
2033	DrawdownDebitAccount.IdCode is missing or invalid.
2034	DrawdownDebitAccount.Name is missing or invalid.
2035	Cannot respond to our own investigation requests.
2036	Name and Address are required when using IntermediaryFi.
2037	Beneficiary name is required.
2038	Beneficiary address is required.
2039	Beneficiary.Identifier is not a valid BIC.
2040	BeneficiaryFi.Identifier is not a valid BIC.
2041	IntermediaryFi.Identifier is not a valid BIC.
2042	Beneficiary.IdCode must be one of 1, 2, 3, 4, 5, 9, B, D.
2043	Originator.Identifier is not a valid BIC.
2044	At least one line of text is required for the requested type code.



Code	Description
2045	Invalid or missing investigation type code.
2046	Invalid investigation subtype code.
2047	Invalid or missing investigation reason code.
2048	Invalid investigation reason subtype code.
2049	Narrative is required when reason code is NARR.
2050	ISO is not enabled.
2051	Source payment is not an investigation request.
2052	Cannot investigate untransmitted payments.
2053	Invalid or missing investigation status code.
2054	Invalid or missing investigation status reason code.
2200	Invalid submission status
2201	Submission must be pending
2202	Submission must be processing
2203	Submission must be created
2300	Distribution must be pending
2301	Distribution must be verified
2302	Distribution must be released
2303	Distribution must be generating
2304	Distribution must not be released
2305	Distribution already canceled
2306	No payments to distribute
2307	Distribution currently generating
2308	Distribution must be transmitted
2309	Distribution must be transmitted but not acknowledged
2400	Hold is not active

Code	Description
2401	Hold is not evaluating
2500	Scan already pending
2501	No scan lists are configured for account
2502	Payment type must be a transfer
2503	Payment must be on hold to request a rescan
2550	Invalid posting record status
2600	This action requires dual control
2601	Accounts must be unique
2602	Inbound Suspense Account not found
2603	Outbound Suspense Account not found
2604	Operator Account not found
2605	Unposted Account not found
2606	Change not approved
2607	Change not pending
2608	Global profile not found or is invalid
2609	Reversal Account not found
2610	Blocked Account not found
2650	Account not found
2651	A whitelist entry with that criteria already exists
2652	Change not approved
2653	This action requires dual control
2700	Account not found
2701	An approval policy entry with that criteria already exists
2702	Change not approved
2703	This action requires dual control

Code	Description
3000	General exception
3001	Origination payment amount exceeds user permissions
3002	Clearing hold exceeds user permissions for payment amount
3003	Posting flags not allowed for user
3004	User not authorized to reject inbound wires
3005	Distribution max wire amount exceeds user permissions
3006	User not authorized to clear this type of hold
3200	Access denied

## 6.6. Checks

---

### Checks concepts

**Get started** with Cross River APIs.

Once you have access to our **sandbox** you can access these API endpoints in the `checks` module through **Swagger**.

### GET calls

- Add query parameters to filter the response the API returns to general GET calls.
- Use **pagination** to control presentation of your results.

This table presents checks management APIs that we describe in detail.

Action	API Call	Description
<b><u>Deposit check</u></b>	POST /checks/v1/payments	Sends COS the information needed to deposit a check into a CR account
<b><u>Cancel deposit</u></b>	POST /checks/v1/payments/{id} /cancel	Cancels a check deposit
<b><u>View check</u></b>	GET /checks/v1/payments/{id} /images/{view}	Returns the image of the front or back of the check encoded in Base64
<b><u>Image analysis</u></b>	GET /checks/v1/payments/{id} /analysis	Returns an analysis of the check's image quality assurance (IQA) status based on the Fed's metrics
Positive Pay: <b><u>Authorize check</u></b>	POST /checks/v1/positive-pay-authorizations	Authorizes a check using Positive Pay
Positive Pay: <b><u>Revoke authorization</u></b>	POST checks/v1/positive-pay-authorizations/{id}/revoke	Revokes Positive Pay authorization already given for a specific check

## Related topics

- [Webhook events](#)
- [Request and response codes](#)
- [Error codes](#)

## Tutorials

- [Checks](#)

## 6.6.1. **APIs**

### 6.6.1.1. Deposit check

---

## Endpoint: /checks/v1/payments

Sends the Cross River system information needed to deposit a check into a Cross River account using the API. This information includes images of both the front and the back of the check. You submit these images after converting each one to a Base64-encoded value.

**POST**

https://sandbox.crbcos.com/checks/v1/payments



**Request**

Response

### BODY PARAMETERS

**accountNumber** String **required**

Number of the Cross River account receiving the check

**amount** Integer **required**

Money amount that appears on the check

**frontImage** String **required**

The base64-encoded image of the front of the check

**backImage** String **required**

The base64-encoded image of the back of the check

**purpose** String **optional**

Purpose of the deposited check

50 characters maximum



**clientIdIdentifier** String optional

Use this attribute to add your own unique identifying string to a payment call or COS record. This attribute is useful for idempotency purposes.

50 characters maximum

**isRedeposit** Boolean optional

True if the check was deposited before. Otherwise false.



Curl



Node.js



Python



Ruby



Go



```
curl --location 'https://sandbox.crbcos.com/checks/v1/payments' \
--header 'Content-Type: application/json' \
--header 'Authorization: Bearer <token>' \
--data '{
  "accountNumber": "2640511503",
  "amount": 10000,
  "frontImage": "/9j/4QuqRXhpZgAATU0AKgA...",
  "backImage": "/9j/4REyRXhpZgAATU0AKgAA...",
  "isRedeposit": false,
  "purpose": "SKIP_IQA"
}'
```

## Responses

● 200

● 400



```
{
  "errors": [
    {
      "code": 2301,
      "message": "Deposits not allowed for account type"
    }
  ]
}
```

## 6.6.1.2. Cancel deposit

---

## Endpoint: /checks/v1/payments/{id}/cancel

Cancels a specific check deposit according to its payment ID. There are times when you need to cancel a check deposit. For example, to prevent a loss on an account where check kiting activity has been identified. You can only cancel a check if the payment hasn't been processed yet.

**POST**

<https://sandbox.crbcos.com/checks/v1/payments/{id}/cancel>



**Request**

Response

### PATH PARAMS

<b>id</b>	String	<b>required</b>
-----------	--------	-----------------

Check payment ID, in GUID format



Curl



Node.js



Python



Ruby



Go



```
curl --location --request POST 'https://sandbox.crbcos.com/checks/v1/payme  
--data ''
```

Responses

● 200



```
{
```

```
  "id": "18619219-ccbb-45a6-a068-b30c008f7b47",
```

```
  "accountNumber": "2640511502"
```

## 6.6.1.3. View check

### Endpoint: /checks/v1/payments/{id}/images/{view}

Returns the Base64-encoded image of the requested side of the check.

Use a Base64 decoder to view the requested check side as a human-visible image.

GET

<https://sandbox.crbcos.com/checks/v1/payments/{id}/images/{view}>

Request

Response

#### PATH PARAMS

**id** String **required**

Check payment ID, in GUID format

**view** String **required**

What part of the check image to view:

- Front
- Back
- Other



Curl



Node.js



Python



Ruby



Go



```
curl --location 'https://sandbox.crbcos.com/checks/v1/payments/18619219-cc'
```

#### Responses

● 200



```
{
  "content": "data:image/png;base64,iVBORw0KGgoAAAANSUUEUgAABLAIAAAImCAYA"
}
```





## 6.6.1.4. Image analysis

---

Returns an analysis of the check's image quality assurance (IQA) status based on the **Fed's metrics**.

Endpoint: /checks/v1/payments/{id}/analysis

GET

https://sandbox.crbcos.com/checks/v1/payments/{id}/analysis

Request

Response

PATH PARAMS

id	String	required
Check payment ID, in GUID format		



Curl



Node.js



Python



Ruby



Go



```
curl --location 'https://sandbox.crbcos.com/checks/v1/payments/18619219-cc'
```

Responses

● 200



```
{
  "analysis": {
    "data": {
      "accepted": true,
      "processingStatus": "Passed",
      "requestStatus": "Succeeded",
      "processingId": "7014d18e-dce3-478d-9078-c35e3800815e",
      "iqaMessage": "IQAGOOD",
      "transactionStatusCode": 0,
      "transactionId": 1667429,
      "groupName": "Cross River Group",
      "organizationName": "Cross River",
      "submissionDate": "2025-06-30T04:42:26.942507-04:00",
      "flexibleFields": {},
      "readFields": [
        {
          "name": "MICR",
          "value": "d314074269dc28293886c1237",
          "confidence": 1000
        },
        {
          "name": "CheckRoutingNumber",
          "value": "314074269"
        },
        {
          "name": "CheckAccountNumber",
          "value": "28293886"
        },
        {
          "name": "CheckNumber",
          "value": "1237"
        },
        {
          "name": "RecognizedAmount",
          "value": "1.00",
          "confidence": 199
        }
      ],
      "testResults": [
        {
          "checkSide": "Front",
          "name": "MICR Confidence",
          "value": "Passed",
          "threshold": 500,
          "confidence": 1000
        }
      ]
    }
  }
}
```

```
},
{
  "checkSide": "Front",
  "name": "Amounts Match",
  "value": "ManualReview",
  "threshold": 500,
  "confidence": 199
},
{
  "checkSide": "Front",
  "name": "Front Focus",
  "value": "Passed",
  "threshold": 350,
  "confidence": 1000
},
{
  "checkSide": "Back",
  "name": "Back Focus",
  "value": "Passed",
  "threshold": 100,
  "confidence": 595
},
{
  "checkSide": "Front",
  "name": "Shadow on Image",
  "value": "Passed",
  "threshold": 851,
  "confidence": 1000
},
{
  "checkSide": "Back",
  "name": "Shadow on Image",
  "value": "Passed",
  "threshold": 851,
  "confidence": 1000
},
{
  "checkSide": "Front",
  "name": "Contrast of Image",
  "value": "Passed",
  "threshold": 401,
  "confidence": 1000
},
{
  "checkSide": "Back",
  "name": "Contrast of Image",
```

```
    "value": "ManualReview",
    "threshold": 401,
    "confidence": 340
  },
  {
    "checkSide": "Front",
    "name": "Cut Corners",
    "value": "Passed",
    "threshold": 751,
    "confidence": 1000
  },
  {
    "checkSide": "Back",
    "name": "Cut Corners",
    "value": "Passed",
    "threshold": 751,
    "confidence": 1000
  },
  {
    "checkSide": "Front",
    "name": "Image Too Small",
    "value": "Passed",
    "threshold": 501,
    "confidence": 1000
  },
  {
    "checkSide": "Back",
    "name": "Image Too Small",
    "value": "Passed",
    "threshold": 501,
    "confidence": 1000
  },
  {
    "checkSide": "Front",
    "name": "Darkness",
    "value": "Passed",
    "threshold": 401,
    "confidence": 863
  },
  {
    "checkSide": "Back",
    "name": "Darkness",
    "value": "Passed",
    "threshold": 401,
    "confidence": 839
  },
}
```

```
{
  "checkSide": "Front",
  "name": "View Angle",
  "value": "Passed",
  "threshold": 701,
  "confidence": 970
},
{
  "checkSide": "Back",
  "name": "View Angle",
  "value": "Passed",
  "threshold": 701,
  "confidence": 967
},
{
  "checkSide": "Front",
  "name": "Rotation Angle",
  "value": "Passed",
  "threshold": 701,
  "confidence": 991
},
{
  "checkSide": "Back",
  "name": "Rotation Angle",
  "value": "Passed",
  "threshold": 701,
  "confidence": 998
},
{
  "checkSide": "Front",
  "name": "Folded Or Torn Corner",
  "value": "Unknown",
  "threshold": 0,
  "confidence": 0
},
{
  "checkSide": "Back",
  "name": "Folded Or Torn Corner",
  "value": "Unknown",
  "threshold": 0,
  "confidence": 0
},
{
  "checkSide": "Front",
  "name": "Folded Or Torn Edge",
  "value": "Unknown",
```



```
        "threshold": 0,
        "confidence": 0
    },
    {
        "checkSide": "Back",
        "name": "Folded Or Torn Edge",
        "value": "Unknown",
        "threshold": 0,
        "confidence": 0
    },
    {
        "checkSide": "Front",
        "name": "Excessive Skew",
        "value": "Unknown",
        "threshold": 0,
        "confidence": 0
    },
    {
        "checkSide": "Back",
        "name": "Excessive Skew",
        "value": "Unknown",
        "threshold": 0,
        "confidence": 0
    },
    {
        "checkSide": "Front",
        "name": "Piggyback Document",
        "value": "Unknown",
        "threshold": 0,
        "confidence": 0
    },
    {
        "checkSide": "Back",
        "name": "Piggyback Document",
        "value": "Unknown",
        "threshold": 0,
        "confidence": 0
    },
    {
        "checkSide": "Front",
        "name": "Too Light",
        "value": "Unknown",
        "threshold": 0,
        "confidence": 0
    },
    {
```

```
    "checkSide": "Back",
    "name": "Too Light",
    "value": "Unknown",
    "threshold": 0,
    "confidence": 0
  },
  {
    "checkSide": "Front",
    "name": "Too Dark",
    "value": "Unknown",
    "threshold": 0,
    "confidence": 0
  },
  {
    "checkSide": "Back",
    "name": "Too Dark",
    "value": "Unknown",
    "threshold": 0,
    "confidence": 0
  },
  {
    "checkSide": "Front",
    "name": "Undersize Image",
    "value": "Unknown",
    "threshold": 0,
    "confidence": 0
  },
  {
    "checkSide": "Back",
    "name": "Undersize Image",
    "value": "Unknown",
    "threshold": 0,
    "confidence": 0
  },
  {
    "checkSide": "Front",
    "name": "Oversize Image",
    "value": "Unknown",
    "threshold": 0,
    "confidence": 0
  },
  {
    "checkSide": "Back",
    "name": "Oversize Image",
    "value": "Unknown",
    "threshold": 0,
```

```
        "confidence": 0
    },
    {
        "checkSide": "Front",
        "name": "Excessive Spot Noise",
        "value": "Unknown",
        "threshold": 0,
        "confidence": 0
    },
    {
        "checkSide": "Back",
        "name": "Excessive Spot Noise",
        "value": "Unknown",
        "threshold": 0,
        "confidence": 0
    },
    {
        "checkSide": "Back",
        "name": "Endorsement Presence",
        "value": "ManualReview",
        "threshold": 750,
        "confidence": 2
    },
    {
        "checkSide": "Front",
        "name": "Aspect Ratio Validation",
        "value": "Passed",
        "threshold": 900,
        "confidence": 998
    },
    {
        "checkSide": "Front",
        "name": "MICR Intrusion Detection",
        "value": "Passed",
        "threshold": 100,
        "confidence": 1000
    },
    {
        "checkSide": "Front",
        "name": "Check Length",
        "value": "Unknown",
        "threshold": 0,
        "confidence": 0
    },
    {
        "checkSide": "Front",
```

"name": "Check Height".

## 6.6.1.5. Authorize check

---

Authorizes a check using Positive Pay.

## Endpoint: /checks/v1/positive-pay-authorizations

**POST**

<https://sandbox.crbcos.com/checks/v1/positive-pay-authorizations> 

Request

Response

### BODY PARAMETERS

**accountNumber** String **required**

Number of the account receiving the check

**amount** Integer **required**

Money amount that appears on the check

**checkNumber** String **required**

The serial number that appears on the check

20 characters maximum

**payeeName** String **required**

Name written on the pay to the order of line of the check

255 characters maximum

**expiresAt** String **optional**

The date and time the authorization expires. In this case, the date and time are in this format: yyyy-mm-ddThh:mm:ss[.mmm]



Curl



Node.js



Python



Ruby



Go



```
curl --location 'https://sandbox.crbcos.com/checks/v1/positive-pay-authori
--data '{
  "accountNumber": "2640511503",
  "amount": 10000,
  "checkNumber": 64958137456,
  "payeeName": "John Strong",
  "expiresAt": "2025-07-19T12:12:27.270Z"
}'
```

## Responses

● 200



```
{
  "id": "bf50797f-bdd7-4cc9-8efc-b30d008043d3",
  "status": "Authorized",
  "accountNumber": "2640511503",
  "payeeName": "John Strong",
  "checkNumber": "64958137456",
  "amount": 10000,
  "createdAt": "2025-07-01T03:46:59.9029464-04:00",
  "expiresAt": "2025-07-19T08:12:27.27-04:00",
  "productId": "83bed086-8182-4151-a1e3-af5b01362783",
  "partnerId": "cd9c12f4-7691-424a-b38b-af5b0134c611",
  "lastModifiedAt": "2025-07-01T03:46:59.9029464-04:00"
}
```

## 6.6.1.6. Revoke authorization

---



## Endpoint: /checks/v1/positive-pay-authorizations/{id}/revoke

Revokes authorization already given for a specific check. You can only revoke authorization prior to the authorization expiration date.

**POST**

<https://sandbox.crbcos.com/checks/v1/positive-pay-authorizations/{id}/revoke> 

Request

Response

### PATH PARAMS

**id** String **required**

Positive pay authorization ID received in the initial authorization response (id attribute).



Curl



Node.js



Python



Ruby



Go



```
curl --location --request POST 'https://sandbox.crbcos.com/checks/v1/positive-pay-authorizations/{id}/revoke' --data ''
```

### Responses

● 200



```
{
  "id": "bf50797f-bdd7-4cc9-8efc-b30d008043d3",
  "status": "Revoked",
  "accountNumber": "2640511503",
  "payeeName": "John Strong",
  "checkNumber": "64958137456",
  "amount": 10000,
  "createdAt": "2025-07-01T03:46:59.903-04:00",
  "expiresAt": "2025-07-19T08:12:27.27-04:00",
  "revokedAt": "2025-07-01T03:54:23.1845224-04:00",
  "productId": "83bed086-8182-4151-a1e3-af5b01362783",
  "partnerId": "cd9c12f4-7691-424a-b38b-af5b0134c611",
  "lastModifiedAt": "2025-07-01T03:54:23.2001547-04:00"
}
```



# 6.6.2. Webhook events

When you work with Cross River accounts, cards and payments, you register for specific webhooks which have a specific event format.

This table presents common `checks` webhook events that are described in detail below.

Event Name	Description
<u>Check.Payment.Sent</u>	A check deposit has been sent to the Federal Reserve for clearing
<u>Check.Payment.Received</u>	A check drawn on an account in COS has been deposited at another bank and has been received from the Federal Reserve for posting
<u>Check.Payment.Rejected</u>	Outbound check could not be processed due to compliance reasons or was rejected by the Federal Reserve

## Event examples

### Check.Payment.Sent

```
{
  "id": "bfdf9c39-633f-4cf2-be4a-b2e201334528",
  "eventName": "Check.Payment.Sent",
  "status": "Pending",
  "partnerId": "1e5d3f04-ae24-4af6-9e30-aecf012b99dd",
  "createdAt": "2025-05-19T14:38:44.177+00:00",
  "resources": [
    "checks/v1/payments/9a2975ce-2466-4c4b-87a0-b2e201330da8"
  ],
  "details": [
    {
      "paymentId": "9a2975ce-2466-4c4b-87a0-b2e201330da8",
      "paymentType": "Forward",
      "coreTransactionId": "2ccddeaaf-a957-4936-8837-b2e201330da8",
      "memoPostId": "879645d9-ec6e-4ffe-beeb-b2e201330da8",
      "accountNumber": "2207975570",
      "depositBusinessDate": "250519",
      "postingCode": "OK",
      "amount": "15000",
      "recognizedAmount": "100",
      "payerRoutingNumber": "314074269",
      "payerAccountNumber": "28293886",
      "checkNumber": "1237",
      "checkType": "Standard",
      "sequenceNumber": "5894754816",
      "micr": "1237",
      "purpose": "SKIP_IQA",
      "clientIdentifier": null,
      "schedule": "0,15000",
      "policy": "Standard",
      "rejectionReason": null,
      "isRedeposit": "False",
      "originalPaymentId": "9a2975ce-2466-4c4b-87a0-b2e201330da8"
    }
  ]
}
```

## Check.Payment.Received



```
{
  "id": "6575634b-0f2e-4986-aa9b-b2e20130315d",
  "eventName": "Check.Payment.Received",
  "status": "Pending",
  "partnerId": "1e5d3f04-ae24-4af6-9e30-aecf012b99dd",
  "createdAt": "2025-05-19T14:27:31.927+00:00",
  "resources": [
    "checks/v1/payments/9fd83f35-8f40-4979-9a70-b2e201302e9b"
  ],
  "details": [
    {
      "paymentId": "9fd83f35-8f40-4979-9a70-b2e201302e9b",
      "coreTransactionId": "62ed24fa-f870-48b6-b948-b2e201302e9b",
      "accountNumber": "2375027519",
      "checkNumber": "1005",
      "bofdRoutingNumber": "061000146",
      "payeeName": "Verizon Wireless",
      "checkType": "Unknown",
      "sequenceNumber": "3506582471",
      "clientIdIdentifier": null,
      "positivePayResult": null,
      "positivePayMatchId": null,
      "amount": "5999"
    }
  ]
}
```

## Check.Payment.Rejected



```
{
  "id": "fecaad6b-a2be-40e3-a4ad-b2e20133afcc",
  "eventName": "Check.Payment.Rejected",
  "status": "Pending",
  "partnerId": "1e5d3f04-ae24-4af6-9e30-aecf012b99dd",
  "createdAt": "2025-05-19T14:40:15.167+00:00",
  "resources": [
    "checks/v1/payments/a0a8c33b-9e30-4356-8c13-b2e201338f14"
  ],
  "details": [
    {
      "paymentId": "a0a8c33b-9e30-4356-8c13-b2e201338f14",
      "paymentType": "Forward",
      "coreTransactionId": "2c951cb5-b9f1-4455-b1b9-b2e201338f14",
      "memoPostId": "79c6727e-2fb8-4d8f-a207-b2e201338f14",
      "accountNumber": "2207975570",
      "depositBusinessDate": "250519",
      "postingCode": "OK",
      "amount": "75000",
      "recognizedAmount": "100",
      "payerRoutingNumber": "314074269",
      "payerAccountNumber": "28293886",
      "checkNumber": "1237",
      "checkType": "Standard",
      "sequenceNumber": "6017013176",
      "micr": "1237",
      "purpose": "SKIP_IQA",
      "clientIdentifier": null,
      "schedule": "0,7500,67500",
      "policy": "Standard",
      "rejectionReason": "NotSpecified",
      "isRedeposit": "False",
      "originalPaymentId": "a0a8c33b-9e30-4356-8c13-b2e201338f14"
    }
  ]
}
```

## 6.6.3. Request and response codes

---

### Check Types

The `checkType` attribute in the response to `POST /Checks/v1/payments` describes the type of check the call is dealing with.

Type	Description
Standard	A check drawn from a business or personal account at another bank
OnUs	A check drawn from a CR account
Treasury	A check drawn on the US Treasury and issued by the US Department of the Treasury
MoneyOrder	A pre-paid traceable certificate used to transfer funds in amounts of up to 1000 USD
FedReserve Bank	A check drawn on the Federal Reserve Bank
FedHomeLoan	A check issued by a member Federal Home Loan bank
StateLocalGovt	A check issued by a state or local government
Certified	A check guaranteed by the issuing FI to be valid and that the account it is drawn on has sufficient funds to cover the amount
Unknown	Check type cannot be determined

### Direction

The `direction` attribute in the response to `POST /Checks/v1/payments` describes whether the check funds are going *out of* or *into* the Cross River bank account.

Direction	Description
Inbound	A check drawn from a COS account, deposited at another bank
Outbound	A check deposited into a COS account, drawn from an account at another bank

## Payment Status

The status of the payment is returned in the response to several check calls in the `status` field.

Status	Description
Created	We have received the payment, but have not started processing it yet. This status should only appear briefly under normal circumstances.
Pending	The payment is waiting to be batched and sent to the Federal Reserve
Hold	Payment is being held at the moment and reviewed by our Operations Team
Batched	The payment has been batched and a final review is being done before we send it out in a file to the Federal Reserve
Processing	For inbound payments, we are attempting to post the payment to the receiving account. For outbound payments, the payment has been sent to the Federal Reserve, but has not posted yet.
Completed	The payment has been posted and accepted by the Federal Reserve (in the case of outbound payments). This is a final status.
Rejected	Your check has been rejected for compliance reasons. Applies to Outbound payments only. This is a final status.
Canceled	An outbound payment has been canceled at the request of the partner. A payment may only be canceled while either pending or on hold. This is a final status.

## Positive Pay Status



If Positive Pay is enabled, the positive pay status of the check:

Status	Description
Authorized	The check has been authorized for positive pay
Paid	An authorized check has been presented by the Federal Reserve and paid by the Bank
Revoked	The check's positive pay authorization has been revoked

## Payment Types

The check `paymentType` field, found in responses to checks calls, indicates whether the check has been presented for deposit or has been returned for some reason.

Payment types include:

Type	Description
Forward	A forward item is a check presented for deposit, to be sent to the payer bank
Return	A previous forward item that the payer bank returned

## Availability Policy/Reg CC Policy Types

The policy type supplied in checks responses in the `policy` field, tells you how within how many days funds from a check deposit will be available in the customer's account.

Standard	<b>\$225 or less of the aggregate amount deposited daily to be available next business day.</b> <b>Remainder amount available on the 2nd business day after the deposit date</b>
NewAccount	\$5,525.00 or less of the aggregate amount deposited daily to be available 2nd business day Excess of \$5,525 available on the 7th business day after the deposit date
LargeDeposits	\$225 or less of the aggregate amount deposited daily to be available next business day \$5,300 or less of the aggregate amount deposited daily to be available on the 2nd business day Remainder of the deposit to be available on the 7th business day after the deposit date
RedepositedCheck	Total amount of the deposit to be available on the 7th business day after the deposit date
RepeatedOverdrafts	Total amount of the deposit to be available on the 7th business day after the deposit date
OnUs	Total amount of the deposit to be available on the 2nd business day after the deposit date
RCNoticeOfUnpaidReturn	Total amount of the deposit to be available on the 7th business day after the deposit date
RCSuspectFraud	Total amount of the deposit to be available on the 7th business day after the deposit date
RCFundingAccountOverdrafts	Total amount of the deposit to be available on the 7th business day after the deposit date
RCUnverifiedEndorsement	Total amount of the deposit to be available on the 7th business day after the deposit date
RCInconsistentInformation	Total amount of the deposit to be available on the 7th business day after the deposit date
RCErasuresOrAlterations	Total amount of the deposit to be available on the 7th business

Standard	<b>\$225 or less of the aggregate amount deposited daily to be available next business day.</b> <b>Remainder amount available on the 2nd business day after the deposit date</b>
ns	day after the deposit date
RCOutOfDateRoutingNumber	Total amount of the deposit to be available on the 7th business day after the deposit date
RCPostDatedOrStaleDate	Total amount of the deposit to be available on the 7th business day after the deposit date
RCPayingBankNotPaidIndication	Total amount of the deposit to be available on the 7th business day after the deposit date
RCLostOrDamaged	Total amount of the deposit to be available on the 7th business day after the deposit date
EmergencyConditions	Total amount of the deposit to be available on the 7th business day after the deposit date

## Rejection Reasons

There are many reasons why a check payment might be rejected. These values appear in the `rejectionReason` field of checks responses.

Reason	Description
ImageAnalysisFailure	The software was not able to read the check image
PostingException	The deposit is unable to post (for example, because of NSF or restriction)
AmountMismatch	The written and numeric check amount numbers do not match
MaxItemAmountExceeded	The limit of number of checks deposited has been reached
MaxDepositAmountExceeded	The limit of dollar volume for check deposits has been reached
MaxItemsPerDayExceeded	The limit of amount of checks deposited has been reached for the day
Duplicate	The check has already been deposited
PayerRoutingNumberInvalid	The routing number on the check is invalid
PayerAccountNumberInvalid	The account number on the check is invalid
CheckNumberInvalid	The check number on the check is invalid
NotSpecified	

## Return Codes

A returned check response contains many details, including a `returnCode` that tells you why the check was returned.

Code	Description
A	Not Sufficient Funds
B	Uncollected Funds Hold
C	Stop Payment
D	Closed Account
E	Unable to Locate Account
F	Frozen/Blocked Account
G	Stale Dated
H	Post Dated
I	Endorsement Missing
J	Endorsement Irregular
K	Signature(s) Missing
L	Signature(s) Irregular
M	Non-Cash Item (Non-Negotiable)
N	Altered/Fictitious Item
P	Item Exceeds Dollar Limit
Q	Not Authorized
R	Branch/Account Sold (Wrong Bank)
S	Refer to Maker
T	Stop Payment Suspect
W	Cannot Determine Amount
X	Refer to Image

## 6.6.4. Error codes

---

Code	Description
2000	General exception
2001	Invalid payment status
2002	Invalid posting status
2003	Payment cannot be canceled
2004	Account not found
2005	Payment must be outbound
2006	Payment must be inbound
2007	Payment must be an inbound origination
2008	Payment must be status completed or rejected to be corrected
2009	Invalid change code
2010	Payment must be an inbound return
2011	Payment must be an origination
2012	Payment must be status completed or rejected to be returned
2013	Payment must be status completed or rejected to be dishonored
2014	Invalid return code
2015	Invalid dishonored return code
2016	Original payment not found
2017	Cannot link to same payment ID
2018	Payment type must be return or notification of change
2019	Original payment must be completed
2020	Payment must be on hold to request a rescan
2021	No scan lists are configured for account
2022	Scan already pending
2023	Previous payment not found
2024	Previous payment must be a completed outbound origination

Code	Description
2025	Receiver account not found
2026	Active holds found
2027	Posting account cannot be changed due to current payment status
2028	Posting status must be pending or failed to change posting account
2029	Posting status must be failed to attempt a retry
2030	Originator profile address missing or invalid
2031	Payment image data not available
2032	Invalid front image format
2033	Invalid back image format
2200	Batch not authorizing
2201	Batch requires one or more payments
2202	No filters were found on the request
2300	No active configuration found for account
2301	Deposits not allowed for account type
2306	Max payment amount exceeded
2309	Origination not allowed from a restricted account
2310	Account config change not approved
2311	Account config change not pending
2312	Account config change approval requires dual control
2313	Account configuration not found or is invalid
2314	Global profile not found or is invalid
2315	Global profile change not approved
2316	Global profile change not pending
2317	Product change not pending
2318	Product change not approved



Code	Description
2319	Global config change approval requires dual control
2320	Global config required
2321	Global config accounts must be unique
2323	Inbound Suspense Account not found
2324	Outbound Suspense Account not found
2325	Return Account not found
2326	Operator Account not found
2327	Unposted Account not found
2328	Product config change approval requires dual control
2329	Account config required
2330	Product config required
2400	Distribution must be generating
2401	Distribution already canceled
2402	Distribution must be pending
2403	Distribution must be on hold
2404	Distribution must be verified
2405	Distribution must be released
2406	Distribution must not be released
2407	Distribution must be transmitted
2408	Distribution must be acknowledged
2409	Distribution must be processing
2410	Distribution must have been previously transmitted
2411	Distribution release requires dual control
2412	A distribution is currently generating
2413	No payments to distribute

Code	Description
2414	Distribution is currently processing and can no longer be acknowledged
2415	Distribution is currently complete and can no longer be acknowledged
2416	Cannot verify distribution with exceeded limits
2417	No filters were found on the request
2500	Submission import count cannot exceed total payment count
2501	Submission process count cannot exceed total payment count
2600	Hold is not active
2601	Hold payment not found
2602	No filters were found on the request
2603	This action requires dual control
3000	General exception
3001	Flags not allowed
3002	Clearing hold exceeds user permissions for payment amount
3200	Access denied

## 7. Lending

---

### Lending concepts

Get started with Cross River APIs.

## Loan APIs

The Lending platform includes a robust set of API calls to allow you to create, update, and manage your loans. The platform consists of a set of interconnected micro services that communicates internally using messages. The platform is designed with high fidelity, high performance and eventual consistency in mind.

- **Oversight** UI for policies and rules
- **preApproval** UI and API for Applications
- **Arix-Origination** API for issuing loans
- **Manual Review** UI for working the exception queue detected by **MPLLifeCycle**
- **Arix-funding** API for making payments
- **Selling** UI and API for purchasing loans
- **Contracts** UI for viewing configuration
- **Hooks** UI and API for listening to events

### IMPORTANT

You must have the following IP addresses allowlisted:

Sandbox - 66.206.202.39 , 66.206.202.12

Production -66.206.202.62 , 66.206.202.15

## Lending status page

Sign up for maintenance updates and see the current status of the Arix system at <https://crossriverarix.statuspage.io>

## Where to start

Go to [get API credentials](#) to begin. Where you will find base URLs and Swagger links.

## API reference documentation

Click each of the links for API reference documentation. These calls create loans and update loan information.

- [Create and update a loan](#) POST /Loan
- [Add loan documents](#) POST /Loan/{Id}/Attachments
- [Request new payment rails](#) PUT /Loan/{Id}/FundingInfo
- [Get loan details by ID](#) GET /LoanDetail/{Id}
- [Cancel a loan](#) PUT /Loan/{{loanId}}/Cancel

## 7.1. Application decisioning

---

### Preapproval

MPLs perform due diligence for every loan request they receive. An application can be submitted to the Preapproval API whether or not it has been approved or denied. If it hasn't yet been approved, MPLs can revise the loan application later after a decision was **reached, approved, or denied** (this includes rejected applications), in accordance with the Fair Lending law.

#### IMPORTANT

To access the links below you must have the following IP addresses allowlisted:

Sandbox - 66.206.202.39 , 66.206.202.12

Production -66.206.202.62 , 66.206.202.15

- Swagger: <https://lendingsandbox.crbcos.com/preapproval/swagger>
- Base URL for sandbox: <https://lendingsandbox.crbcos.com/preapproval/v1>
- Base URL for production: <https://lending.crbcos.com/preapproval/v1>
- API scope for sandbox is: CosLending:PreApproval:stg
- API scope for production is: CosLending:PreApproval:prd
- The sandbox UI is <https://lendingappsandbox.crbcos.com/preapproval/applications>
- The production UI is <https://lendingapp.crbcos.com/preapproval/applications>

API	Description
<a href="#"><u>POST /applications/approve</u></a>	Post an application marked as <b>approved</b>
<a href="#"><u>POST /preapproval/v2/applications/{id}/attachments</u></a>	Post file attachments to an existing application
<a href="#"><u>POST /applications/deny</u></a>	Post an application marked as <b>denied</b> , with a declination object attached
<a href="#"><u>POST /application</u></a>	Post an application without the decisioning
<a href="#"><u>POST /applications/{id}/approve</u></a>	Update an existing application to <b>approved</b> status
<a href="#"><u>POST /applications/{id}/deny</u></a>	Update an existing application to <b>denied</b> status

#### Notes:

- During implementation, Cross River and the MPL will agree on additional fields to be passed to CR based on the MPL's unique operating model.
- During implementation, Cross River and the MPL will also agree on the content of the data using the Data Dictionary Monday board. Questions on specific fields should be posted there. If you don't yet have access, contact your Relationship Manager (RM).
- Data types and schemas are available directly in Swagger, which is the "source of truth".
- MPLs should send in their Final Decision using either POST /approve or POST /deny.
- The rules that run when sending in the PreApproval API, are informational. They do not change the status of the application automatically.

## Cross River rules on preapproval

When posting an application using the Preapproval API, Cross River checks the data against a set of regulatory rules and returns the rule results in the API response. The rules run on the Post /Application call (known in the Postman collection as *non-decisioned*). The rules will run on POST /applications/approve as well. The response body includes rule results in the ruleSetResults object, which includes information about the rule and the data

used when running the rule. If one of the rules fails on Preapproval, you can assume it will also fail in Arix.

- Currently, there are no webhooks in preapproval.
- The results of the rules are returned synchronously in the response.
- Cross River Ops cannot manually override the result of a rule in pre-approval.
- Pre-approval rules do not alter the status of an application in pre-approval sent in by an MPL.

The following is an example of a `ruleSetResults` object that contains the Cross River Internal List Rule.

JSON



```
"ruleSetResults": [  
  {  
    "status": "Success",  
    "rulesProcessDate": "2023-09-06",  
    "ruleResults": [  
      {  
        "ruleId": "92bc95b5-fa3c-468a-b28b-b06800bdb9d7",  
        "version": "v1",  
        "ruleName": "Internal List Check",  
        "ruleDescription": "Borrower information should not b",  
        "passed": true,  
        "data": "Anita Loan 693952XXX,Scrooge McDuck 01451299",  
        "resultInfo": null  
      }  
    ],  
    "preApprovalLoanId": "693751f7-67ec-401a-b945-b0750068734e"  
  }  
]
```

## Data Dictionary

The Data Dictionary is presented to each MPL during the onboarding process. The content to be sent will be agreed upon between Cross River and the MPL before launch. If you can't access the Data Dictionary, contact your RM.

## POST /applications/approve

Use this API to send in a unique application that hasn't been submitted already, and mark it as **approved**. This is the preferred method for sending in an **approved** application.

- For applications that were approved, but not accepted, you should use this API and mark `AcceptedByCustomer` as **false**.
- For applications that were approved and accepted by the customer, you should use this API and mark `AcceptedByCustomer` as **true**.

The API returns a unique ID that represents this application in our system.

If the decision changes afterwards, you can update it using `POST /applications/{ID}/deny`.

## POST /applications/deny

Use this API to send in a unique application that hasn't been submitted already, and mark it as **denied**. This is the preferred method of sending in an application that has a final status of **denied**. For **denied** applications, an `ApplicationDeniedRequest` Object containing the reasons for denial, must be sent in along with the application information.

The API returns a unique ID that represents this application in our system.

If the decision changes afterwards, you can update it using `POST /applications/{ID}/approve`.

## POST /application

Use this API to send in a unique application that hasn't been submitted already, and where a decision hasn't yet been made yet about the loan application. We *do not* recommend this method of sending in applications unless you have a special use case that requires it.

The API returns a unique ID that represents this application in our system.

When a decision is made, you can update it using `POST /applications/{id}/deny`, `POST /applications/{id}/approve`.



## POST /applications/{id}/approve

Use this API to approve an application that has already been submitted. If the application is missing any required fields for an **approved** application, then send the complete application again.

This API only marks the application as **approved** and does not update other information.

- No Body is used in this call.

## POST /applications/{id}/deny

Use this API to mark as **denied** an application that has already been submitted.

An ApplicationDeniedRequest Object containing the reasons for denial and other denial information must be submitted along with the application information.

### IMPORTANT

Deprecated: CSV File to SFTP folder

For partners who onboarded before 2021, you should continue dropping application tapes into the SFTP folder until you're ready to migrate to API.

- The file format is .csv.
- Naming convention: MplId\_ApplicationTape\_yyyymmdd.csv, where yyyymmdd is the date of file generation.
- Data in the report should be for prior-day applications.
- If a loan changes status (such as from denied to funded), you should include the updated loan record in the tape with the **UpdatedFlag**. The flag indicates that this record updates a previous credit decision for this loan.

## POST/preapproval/v2/applications/{id}/attachments

This new preapproval API allows you to upload supporting files associated with an application and further improves our compliance practices.

### IMPORTANT

The zip file should contain the complete document package. This includes:

- The loan agreement
- AAN
- Credit report
- Relevant documents or responses from vendors used to decision the application according to your credit underwriting policy as well as your BSA-AML processes
- Any data point used as input to your model

For a list of acceptable attachment naming conventions, please refer to [Supporting loan documents](#)

The API is available in the latest Postman collection and in the [Swagger documentation](#).

## 7.2. Create and update a loan (new format)

---

### POST /Loan and PUT /Loan/{Id}

Use these calls to create and update a loan request in Arix.

Request URL `https:// ... Post/Loan`

Request URL `https:// ... Put/Loan/{Id}`

When you update a loan with `PUT/Loan`, use a `LoanUpdate` object. The attributes for a `LoanUpdate` object are the same as those used in the `Loan` object, when you call `POST/Loan`

The table below is the reference information for the API attributes you can use when you create a loan. They are the same attributes that show when you receive a response.

he same attributes are used for the update a loan request: `PUT /Loan/{Id}`

### POST /Loan

Use this call to create a loan request in Arix.

## Endpoint: /Loan

Allows you to create a new loan by making an HTTP POST request to the specified URL. The request should include various details such as loan number, borrower information, loan amount, net funding, interest rate, and other relevant loan parameters. Additionally, the request can include information about different payment fields for rails.

**POST**

https://arixapisandbox.crbnj.net/Loan



**Request**

Response

### BODY PARAMETERS

**LoanNumber**

String

**required**

A customer generated loan number. Up to 100 characters.

**IssuingBankId**

String

**required**

The ID of the issuing bank. CRB, NHB or BRB.

**Platform**

String

**required**

The entity that is the intermediary between the customer and Cross River. 100 characters.

**LoanType**

String

**required**

Select from:

1 - HFS (Hold For Sale)

2 - LTHFS (Long Term Hold For Sale)

4 - CRBRetained (Retained)

**NoteDate**

String

**required**

Date the borrower becomes obligated to pay the loan. Date can be the signing date or the funding date (dependent on the loan documents).

The format must be: yyyy-mm-dd.

**BorrowerLastName** String required

Borrower's last name. 80 characters.

**BorrowerFirstName** String required

Borrower's first name. 80 characters.

**BorrowerDOB** String required

Borrower's date of birth  
The format must be: yyyy-mm-dd.

**BorrowerSSN** String required

Borrower's Social Security Number

**BorrowerAddress** String required

Borrower's address. 160 characters.

**BorrowerCity** String required

Borrower's city of residence. 80 characters.

**BorrowerState** String required

Borrower's state of residence. 40 characters. Letter code.

**BorrowerZip** String required

Borrower's zip code. 20 digits.

**LoanAmount**                      Number                      **required**

Gross dollar amount requested.

This amount represents the sum of the origination fee plus (+) net funding amount.

**NetFunding**                      Number                      **required**

Net dollar amount to be funded

**Rate**                                  Number                      **required**

Interest rate, as a percent

**APR**                                  Number                      **required**

Annual percentage interest rate, as a percent

**ServicingBankId**                      String                      **optional**

ID of the servicing bank

**BorrowerPhone**                      String                      **optional**

Borrower's phone number

**BorrowerEmail**                      String                      **optional**

Borrower's email address. 100 characters.

**Amortization**                      Integer                      **optional**

Number of payments at loan origination to pay off the loan

**Term** Integer optional

Number of months from loan origination to pay off the loan.

**RateType** Integer optional

1 - fixed  
2- floating  
Optional

**PriorLoanFlag** String optional

Whether the borrower took a loan with the platform in the past. 20 characters.  
Y - yes  
N - no

**FICO** Integer optional

Borrower's 3-digit FICO score used for risk assessment

**FICODate** String optional

Date FICO score was obtained  
The format must be: yyyy-mm-dd.

**CreditGrade** String optional

Internal grade for loan or from credit policy. 20 characters.

**AssetClass** Integer optional

Reason for loan:

1 - student

2 - solar

3 - debt consolidation

4 - home improvement

5 - POS (point of service)

6 - medical

7 - auto

99 - other

Optional

**LoanPurpose** String optional

Borrower's stated purpose for the loan. 500 characters.

**Investor** String optional

Name of investor who will be purchasing the loan

Leave blank for loans retained by Cross River.

300 characters.

**Servicer** String optional

The entity that handles the servicing of your account, such as billing statements, notices. 100 characters.

**ApplicationDate** String optional

Date the borrower applied for the loan.

The format must be: yyyy-mm-dd.

**RegBDecisionDate** String optional

Date the loan was approved

The format must be: yyyy-mm-dd.



**PurchaseDate** String optional

Date the lender or investor will purchase the loan from CR  
The format must be: yyyy-mm-dd.

**PaymentDueDate** String optional

Date the first payment is due  
The format must be: yyyy-mm-dd.

**ApprovedLoanAmount** Number optional

Dollar amount the borrower was approved to borrow at the time of application. The amount can change before the final loan amount for reasons related to the borrower or lender.

**PeriodicPayment** Number optional

Dollar amount of monthly payments.

**OriginationFee** Number optional

Charge to the borrower for the loan.

**HomeownerFlag** String optional

Confirm if the borrower is a homeowner. 20 characters.  
Yes or No.

**OpenCreditLines** Integer optional

Number of open credit lines.

**AnnualIncome** Integer optional

Borrower's annual income.

**CreditInquiries12months** Integer optional

Number of credit inquiries made on the borrower in the last 12 months.

**DQPast24months** Integer optional

Number of times the borrower has paid their loan late in the last 24 months on all accounts.

**PublicRecordsOnFile** Integer optional

Number of open public records, such as bankruptcy, tax liens or judgments.

**EmploymentLength** Integer optional

Number of years the borrower has been employed at their current job.

**DebtUtilization** Number optional

Percent of available credit used by borrower.

**TotalRevolvingDebt** Number optional

Dollar amount of all revolving debt on the credit report.

**AccountsOpenedPast24months** Integer optional

Number of accounts the borrower opened in the last 24 months.

**CollectionsExcludingMedical** Integer optional

Number of open collection accounts, not including collections for medical procedures.

**MonthsSinceLastRecord** Integer optional

This field indicates the number of months since the last active public record was on their credit report.

A public record may indicate that a borrower stopped paying bills.

**Employer** String optional

Name of empower. 500 characters.

**DTI** Number optional

Debt to Income Ratio

Ratio of total monthly payment obligations divided by the monthly income as a decimal, not as a percent. It is calculated differently for each credit policy.

It must be submitted as a decimal. The decimal value should be between 0-300.

**MLAFLAG** String optional

If the loan is being made to a covered borrower. 20 characters.

Yes or No.

**MAPR** String optional

20 characters.

A standard calculation used by lenders for loans covered by the Department of Defense's Military Lending Act & Regulation Z.

MAPR is required if MLAFLAG = Yes

For more information, see:

<https://www.fdic.gov/regulations/compliance/manual/5/V-13.1.pdf>

**ReceiverName** String optional

Name of party receiving funds via ACH. 200 characters.

**SameDayFlag** Integer optional

If the loan being funded via ACH or same-day ACH. (Same-day ACH requires set up by Cross River).

**AddendaRecord** String optional

Additional record used for information purposes. 80 characters.

**CoBorrowerFirstName** String optional

Coborrower's first name. 80 characters.

**CoBorrowerLastName** String optional

Coborrower's last name. 80 characters.

**CoBorrowerDOB** String optional

Coborrower's date of birth.

The format must be: yyyy-mm-dd

**CoBorrowerSSN** String optional

Coborrower's Social Security Number.

**CoBorrowerAddress** String optional

Coborrower's address. 160 characters.

**CoBorrowerCity** String optional

Coborrower's city of residence. 80 characters.

**CoBorrowerState**                      String                      optional

Coborrower's State of residence. 40 characters.

**CoBorrowerZip**                      String                      optional

Coborrower's zip code. 20 digits.

**CoBorrowerPhone**                      String                      optional

Coborrower's phone number. 20 digits.

**CoBorrowerEmail**                      String                      optional

Coborrower's email address. 100 characters.

**CoBorrowerFICO**                      String                      optional

Coborrower's FICO score

**MerchantName**                      String                      optional

Merchant's name. 200 characters.

**MerchantFee**                      Number                      optional

Dollar amount paid to merchant.

**RefundAmount**                      String                      optional

Refund amount.

**RefundDate** String optional

Refund date

The format must be: yyyy-mm-dd

**FinalPaymentDate** String optional

Date due for final loan payment

The format must be: yyyy-mm-dd.

**FinalPayment** Number optional

Dollar amount of final payment.

**FinanceCharge** Number optional

Finance charge paid by the borrower.

**TotalPayments** Number optional

Total paid by the borrower after all the payments were made.

**PPY** Number optional

Number of payments per year.

**PaymentFrequency** Integer optional

Frequency of payments:

1 - weekly

2 - semi-monthly

3 - monthly

4 - bi-monthly

5 - other

6 - bi-weekly

**StandardEntryTypeCode**      String      optional

PPD - consumer accounts  
CCD - corporate accounts  
20 characters.

**InterestAccrualMethodDays**      Integer      optional

365 (days)

**SubpoolId**      String      optional

ID of the subpool that the loan belongs to. 100 characters.

**Program**      String      optional

MPL defined program that the loan belongs to. 100 characters.

**Field1**      String      required

Reserved for submitter's IP address  
Up to 5000 characters.

**Field2**      String      required

Reserved for JSON. Up to 5000 characters.

```
"{"  
  ""OFAC decision"":""True|False",  
  ""OFAC decision source"":""""  
}"
```

**Field3**      String      required

Reserved for . Up to 5000 characters.  
See standard JSON format below.

**Field4** String **required**

Reserved for loan agreement version  
Up to 5,000 characters.

**Field5** String optional

Reserved for credit decisioning parameters  
Up to 5,000 characters.  
Required for any credit information requested by Cross River  
See standard JSON format below.

**Field6** String optional

Required for business information  
Up to 5,000 characters.  
See standard JSON format below.

**Field7** String optional

Required for business owner Information  
Collect information on controlling person and business owners who own 25% or more.  
For SBA loans the threshold is 20%.  
Up to 5,000 characters.  
See standard JSON format below.

**Field8** String optional

Open field for text.  
Up to 5,000 characters.

**Field9** String optional

Reserved for fraud vendor raw response  
Up to 5,000 characters.



**Field10**

String

optional

Open field for text.  
Up to 5,000 characters.

**Field11**

String

optional

Open field for text.  
Up to 5,000 characters.

**Field12**

String

optional

Open field for text.  
Up to 5,000 characters.

**Field13**

String

optional

Open field for text.  
Up to 5,000 characters.

**Field14**

String

optional

Open field for text.  
Up to 5,000 characters.

**Field15**

String

optional

Open field for text.  
Up to 5,000 characters.

**Field16**

String

optional

Reserved for second look investors raw response.  
Up to 5,000 characters.

**Field17**

String

optional

Open field for text.  
Up to 5,000 characters.

**Field18**

String

optional

Open field for text.  
Up to 5,000 characters.

**Field19**

String

optional

Open field for text.  
Up to 5,000 characters.

**Field20**

String

optional

Open field for text.  
Up to 5,000 characters.

**Field21**

Array

optional

Optional addition of loan information.  
Byte array in Base 64. byte [ ]. Up to 8,000 characters.

**Rails**

Array

required

List {RailTransaction}.  
For more details about rails: <https://docs.crossriver.com/apis/lending/loan-funding-payment-rails>



```
{
  "Loan": {
    "LoanNumber": {{loanNumber}},
    "IssuingBankId": "CRB",
    "Platform": "string",
    "LoanType": 1,
    "NoteDate": "2019-12-31",
    "BorrowerLastName": Waters,
    "BorrowerFirstName": Therese,
    "BorrowerDOB": "1986-12-31",
    "BorrowerSSN": {{borrowerSSN}},
    "BorrowerAddress": {{borrowerAddress}},
    "BorrowerCity": Davisshire,
    "BorrowerState": {{borrowerState}},
    "BorrowerZip": {{borrowerZip}},
    "LoanAmount": 1500,
    "NetFunding": 1300,
    "Rate": 0.2,
    "APR": "10",
    "BatchId": "string",
    "BorrowerPhone": 959-390-4893,
    "BorrowerEmail": Mara.Howell@hotmail.com,
    "Amortization": 0,
    "Term": 36,
    "RateType": 0,
    "PriorLoanFlag": "string",
    "FICO": 10,
    "FICODate": "2017-12-31",
    "CreditGrade": "string",
    "AssetClass": 0,
    "LoanPurpose": "string",
    "Investor": "string",
    "Servicer": "string",
    "ApplicationDate": {{ApplicationDate}},
    "RegBDecisionDate": {{RegBDecisionDate}},
    "PurchaseDate": {{PurchaseDate}},
    "PaymentDueDate": {{PaymentDueDate}},
    "ApprovedLoanAmount": 0,
    "PeriodicPayment": 0,
    "OriginationFee": 100,
    "HomeownerFlag": "string",
    "OpenCreditLines": 0,
    "AnnualIncome": 0,
    "CreditInquiries12months": 0,
    "DQPast24months": 0,
```

```
"PublicRecordsOnFile": 0,
"EmploymentLength": 0,
"DebtUtilization": 0,
"TotalRevolvingDebt": 0,
"AccountsOpenedPast24months": 0,
"CollectionsExcludingMedical": 0,
"MonthsSinceLastRecord": 0,
"Employer": "string",
"DTI": 0,
"MLAFLAG": "string",
"MAPR": "string",
"ReceiverName": "string",
"SameDayFlag": 0,
"AddendaRecord": "string",
"CoBorrowerFirstName": "string",
"CoBorrowerLastName": "string",
"CoBorrowerDOB": "1986-12-31",
"CoBorrowerSSN": "string",
"CoBorrowerAddress": "string",
"CoBorrowerCity": "string",
"CoBorrowerState": "string",
"CoBorrowerZip": "string",
"CoBorrowerPhone": "string",
"CoBorrowerEmail": "string",
"CoBorrowerFICO": 0,
"MerchantName": "string",
"MerchantFee": 0,
"RefundAmount": 0,
"AdjustedLoanAmount": 0,
"AdjustedNetFunding": 0,
"AdjustedOriginationFee": 0,
"AdjustedCurrentMonthlyPayment": 0,
"RefundDate": "2017-12-31",
"AdjustedMerchantFee": 0,
"FinalPaymentDate": {{FinalPaymentDate}},
"FinalPayment": 0,
"FinanceCharge": 0,
"TotalPayments": 0,
"PPY": 0,
"PaymentFrequency": 0,
"StandardEntryTypeCode": "string",
"InterestAccrualMethodDays": 0,
"SubpoolId": "Moe83",
"Program": "DTM",
"Field1": "string",
"Field2": "string",
```

## Responses

● 200



```
{
  "Id": "bd054ec9-8598-497e-ac2d-ac5400bbbf3e",
  "MPLId": "TST",
  "Date": "/Date(1602674613963-0400)/",
  "LoanNumber": "MZ-1afc220c66ed4ad",
  "IssuingBankId": "CRB",
  "Platform": "string",
  "LoanType": "HFS",
  "NoteDate": "/Date(1577768400000-0500)/",
  "BorrowerLastName": "Emard",
  "BorrowerFirstName": "Darrion",
  "BorrowerDOB": "/Date(536389200000-0500)/",
  "BorrowerSSN": "640517072",
  "BorrowerAddress": "8652_VonRueden_Walks",
  "BorrowerCity": "East",
  "BorrowerState": "LV",
  "BorrowerZip": "30878",
  "LoanAmount": 500,
  "NetFunding": 1300,
  "Rate": 0.05,
  "APR": 10,
  "BatchId": "string",
  "BorrowerPhone": "323-514-0396",
  "BorrowerEmail": "Helga_Hoeger14@gmail.com",
  "Amortization": 0,
  "Term": 36,
  "RateType": 0,
  "PriorLoanFlag": "string",
  "FICO": 10,
  "FICODate": "/Date(1514696400000-0500)/",
  "CreditGrade": "string",
  "AssetClass": 0,
  "LoanPurpose": "string",
  "Investor": "string",
  "Servicer": "string",
  "ApplicationDate": "/Date(1602674613934-0400)/",
  "RegBDecisionDate": "/Date(1602674613934-0400)/",
  "PurchaseDate": "/Date(1602907200000-0400)/",
  "PaymentDueDate": "/Date(1607922000000-0500)/",
  "ApprovedLoanAmount": 0,
  "PeriodicPayment": 0,
  "OriginationFee": 100,
  "HomeownerFlag": "string",
  "OpenCreditLines": 0,
  "AnnualIncome": 0,
```

```
"CreditInquiries12months": 0,
"DQPast24months": 0,
"PublicRecordsOnFile": 0,
"EmploymentLength": 0,
"DebtUtilization": 0,
"TotalRevolvingDebt": 0,
"AccountsOpenedPast24months": 0,
"CollectionsExcludingMedical": 0,
"MonthsSinceLastRecord": 0,
"Employer": "string",
"DTI": 0,
"MLAFLAG": "string",
"MAPR": "string",
"ReceiverName": "string",
"SameDayFlag": 0,
"AddendaRecord": "string",
"CoBorrowerFirstName": "string",
"CoBorrowerLastName": "string",
"CoBorrowerDOB": "/Date(536389200000-0500)/",
"CoBorrowerSSN": "string",
"CoBorrowerAddress": "string",
"CoBorrowerCity": "string",
"CoBorrowerState": "string",
"CoBorrowerZip": "string",
"CoBorrowerPhone": "string",
"CoBorrowerEmail": "string",
"CoBorrowerFICO": 0,
"MerchantName": "string",
"MerchantFee": 0,
"RefundAmount": 0,
"AdjustedLoanAmount": 0,
"AdjustedNetFunding": 0,
"AdjustedOriginationFee": 0,
"AdjustedCurrentMonthlyPayment": 0,
"RefundDate": "/Date(1514696400000-0500)/",
"AdjustedMerchantFee": 0,
"FinalPaymentDate": "/Date(1618372800000-0400)/",
"FinalPayment": 0,
"FinanceCharge": 0,
"TotalPayments": 0,
"PPY": 0,
"PaymentFrequency": 0,
"StandardEntryTypeCode": "string",
"InterestAccrualMethodDays": 0,
"SubpoolId": "Moe83",
"Program": "DTM",
```



```
"Field1": "string",
```

## Field 3 CIP

JSON



```
"{"
  {
    "CIP Decision": "NonDocumentary, Documentary,Other",
    "CIP Vendors used": {
      "CIP KYC vendor": "Socure",
      "CIP bank account vendor": "Plaid"
    },
    "CIP Documents used": {
      "CIP Document": "Driving license",
      "CIP Seconday Document": "Utility Bill"
    },
    "CIP Verification used": {
      "OCR": "Mitek",
      "Fraud": "Experian Fraud",
      "Phone number": "White pages"
    }
  },
  "CIPCollectionMethod": {
    "Name": "FromCustomerviaApplication|FromCustomerviaLeadGenerator|FromExte
    "Address": "FromCustomerviaApplication|FromCustomerviaLeadGenerator|FromE
    "DOB": "FromCustomerviaApplication|FromCustomerviaLeadGenerator|FromExter
    "SSN": "FromCustomerviaApplication|FromCustomerviaLeadGenerator|FromExter
    "CIPCollectionMethodExternalSource": "CreditReport - this field is requir
  }, "TaxNumber":"SSN|ITIN"
}"
```

## Field 5 Credit Decisioning

```
{
  "RevolvingLineFirstAppearance": "yes/no",
  "RevolvingLineGUID": "ID From Origination/First Appearance",
  "RevolvingLineBalanceAsOfWhenCallingCRB": "",
  "FloridaStampTaxCollectedFromCustomer": "Y/N",
  "FloridaStampTaxAmount": "0.00",
  "DMVFeeAmount": "0.00",
  "FloridaStampTaxRateUsed": "0.0035",
  "DiscountedLoan": "Y|N",
  "Purchase discount": "N%",
  "PurchaseDiscountAmount": "0.00",
  "CreditPolicyVersion": "v1.0",
  "CreditModelVersion": "v1.0",
  "CreditBureau": "TU|Experian|Equifax|Innovis|Other",
  "PremiumFee": "0.00",
  "BypassRuleChecking": "True|False"
}
```

## Field 6 Standard Format (Business)

```
{
  "name": "business name",
  "tin": "EIN/SSN",
  "incDate": "2008-01-01",
  "stateIncorp": "AZ",
  "countryDomicile": "US",
  "phone": "4805477526",
  "email": "business email",
  "address": "business address",
    "city": "business city",
  "state": "business state",
  "zipCode": "business zip",
  "physicalAddress": "the complete physical address of the business if diff
    "payroll": "Average monthly payroll",
  "businessType": "Business Entity Type - Sole proprietor, Partnership, C-Co
  "businessPurpose": "Business purpose / description",
  "industry": "xxx",
  "naicsCode": "xxx",
  "franchise": "Yes/No",
  "RegisteredTradingNames": "",
  "ProductServicesOffered": "",
  "InsuranceType": "",
  "AuthorisedSignatory": "",
  "AuthorisedSignatoryVerificationMethod": "",
  "BusinessLicenseInformation": "",
  "BusinessWebSiteVerification": "Yes|No",
  "BusinessAnnualTurnOver": "Gross Annual Revenue, cannot be null",
  "BusinessPricingDetails": {
    "pricingTier2": "",
    "pricingTier3": ""
  },
  "BusinessCreditLimit": {
    "ConsumerCreditLimit": "",
    "DailyCreditLimit": "",
    "BusinessApprovalException": "Y|N"
  }
}
```

## Field 7 Standard Format (Business Owners/Principals)

```
[
  {
    "name": "owner Name",
    "ownershipPer": 24,
    "tin": "SSN|EIN|TIN",
    "address": "owner address",
    "city": "owner city",
    "state": "ownerstate",
    "zipCode": "owner zip",
    "dob": "YYYY-MM-dd",
    "phone": "owner phone",
    "OwnerFico": 700,
    "CIP Decision": "NonDocumentary, Documentary,Other",
    "CIP Vendors used": {
      "CIP KYC vendor": "eg Socure",
      "CIP bank account vendor": "eg Plaid"
    },
    "CIP Documents used": {
      "CIP Document": "eg for a person a Driving license",
      "CIP Secondary Document": "eg for a person Utility Bill"
    },
    "OFAC decision": "True|False",
    "OFAC decision source": "",
    "CIP Verification used": {
      "OCR": "eg Mitek",
      "Fraud": "eg Experian Fraud",
      "Phone number": "eg White pages",
      "type": "Owner | Control person",
      "citizenship": "US",
      "residencyType": "Permanent Resident Alien | Non-Immigrant Alien"
    }
  },
  {}
]
```

Any vendors noted above are provided for illustration only.

### **IMPORTANT**

When you add a date field in a request, the format must be: yyyy-mm-dd.

These fields include BorrowerDOB and FICODate and must be in this format: 1970-12-05

## 7.3. Create a Line-of-Credit (LOC)/ Draws

To operate LOC loans or loans with Multiple Draws, or to enable you to buy **draws**, you need to implement the current process:

### 1 Unique Loan number

- When calling Arix for the first time (LOC opening or first **draw**):  
The field **Loannumber** = `<loan-number>`
- On subsequent loans:  
The field **Loannumber** = `<loanNumber>-yyyymmdd` (add mmss if needed to make more than one draw per day).

### 2 Oversight

We perform oversight checks when the LOC is opened or on the first draw. When calling Arix for the first time, the JSON in `field5` should contain the following:

JSON



```
{
  "RevolvingLineFirstAppearance": "yes",
  "RevolvingLineGUID": null,
  "RevolvingLineBalanceAsOfWhenCallingCRB": null
}
```

The field Rails should be `[]`

For subsequent draws, the JSON in `field5` should contain the following:

```
{
  "RevolvingLineFirstAppearance": "no",
  "RevolvingLineGUID": "GUID from Origination/First Appearance",
  "RevolvingLineBalanceAsOfWhenCallingCRB": "$123"
}
```

Field Rails should contain the payment rails to be funded.

### 3 NetFunding rule

#### Loan Tape:

- NetFunding
- OriginationsFee
- Field5MerchantFee

1. NetFunding=AmountFinanced

2. NetFunding=Loan Amount-Origination Fee-Field5.MerchantFee

Or

3. For draws, NetFunding= sum of the rails being funded

#### Loan Agreement:

- AmountFinanced

### 4 Draw rule

#### Loan Tape

- Field5RevolvingLineFirstAppearance
- Field5RevolvingLineGUID

1. If `RevolvingLineFirstAppearance` = 'No', you can skip all Arix rules if `RevolvingLineGUID` is a valid GUID and it's status is not in (1,101,102,100)

2. If `RevolvingLineGUID` = 'First Appearance', pass this rule

3. Room for additional rules inline with what any checks that you make during draws

### 5 The information provided on subsequent draws has to be current. For example:

- If there is name change, CR expects the draw to have the updated name.
- If there was a material change, such as a credit report was pulled (say on annual cadence), you must provide the updated credit report and treat the loan as a first appearance so that we can check all the details.



## 7.4. Add loan documents

---

Use this call to:

- Post loan documents
- Add loan documents upon Cross River request

### POST /Loan/{loanId}/Attachments

Request URL : <https:// ... /Loan/{Id}/Attachments> (**endpoint below**)

**To add attachments:**

1. Zip all the relevant documents
  - You can send multiple zip files
  - Zip max size is set at 120 MB
  - Files within the zip folder have to follow the naming convention
2. Name the file with the **{Arix GUID}.zip**
3. **POST** the documents

#### **IMPORTANT**

The status of a loan remains **Received** until all the required documents are received by Arix. No webhook is sent to report that docs are incomplete.

### File Naming convention

The naming convention requires that the end of a file name (before the file extension), ends with a dash (-) followed by a document-type code.

#### **Example**

- Loan agreement {JoesCreditReport}-LA.pdf

- Social Security card {JoesCreditReport}-SC.pdf

## **Identification documents**

Type	Suffix
Certificate of Citizenship	-CertificateCitizenship
Court approved name change	-NameChangeCert
Credit card *As a secondary form of ID only	-CC
Credit report	-CR
Debit card * As a secondary form of ID only	-DC
Drivers license	-DL
Insurance card	-InsuranceCard
Marriage Certificate	-MarriageCert
Military ID	-MilitaryID
Miscellaneous ID documents	-MID
Passport	-PP
Permanent resident card	-RC
Proof of address jpeg, jpg, png or pdf provided as POA	-POA
Rental lease agreement	-LeaseAgreement
Selfie Used to verify the ID	-SLF
Social Security card *As a secondary form of ID only	-SC
State Issued ID	-SID
Utility Bill	-UtilityBill
Work visa	-WorkVisa

## Credit documents

Type	Suffix
Bankruptcy Watch	-BankruptcyWatch
Credit memo	-CreditMemo
Credit report	-CR
Credit report (hard pull)	-CR_HP
Credit report (soft pull)	-CR_SP
Credit report( MLA)	-MLA
Experian Instant Prescreen	-EXPInstantPrescreen
Pagaya credit model	-CRM_PGY
Scienaptic.ai credit decisioning	-Scienaptic
Underwriting Checklist	-UWChecklist

## Loan documents and disclosures

Type	Suffix
Adverse Action Notice	-AAN.pdf
Credit Score Disclosure Notice	-CreditScoreDisclosure
Loan agreement	-LA.pdf (Contains Truth in Lending agreement and Promissory Note)
Loan Application	-LApp
Loan Transition Letter	LoanTransition.pdf
Miscellaneous loan documents	-MiscLoanDocs
Notice of Incompleteness	-NOI.pdf
Preauthorized Transfer	-PreAuthTrn (for customer facing authorization requests)
Promissory note	-PN.pdf
Truth in Lending agreement	-TL.pdf

## Income-verification documents

Type	Suffix
Additional pay stub	-PayStubAdditional
Bank statement	-BS
Bank transaction history	-BankTransactionHistory
Business Tax Return	-BusinessTR
Credit card balances	-CreditCardBalances
Credit card statement	-CreditCardStatement
Form 1099	-MI
Customer's payment history	-CustomerPayments History
Form W2	-W2
Pay stub	-PS
Pension/retirement benefit statement	-RS
Plaid Assets	-PlaidAssets
Plaid Income Verification	-PlaidIncome
Point Predictive Income Validation	-PointPredictiveIncome
Previous pay stub	-PayStubPrevious
Social Security Award letter (SSA)	-SS
Tax lien release	-TaxLienRelease
Personal Tax Return	-TR
The Work Number Suffix	-TWN
Truv Income and employment	-truvi
Various income verification documents	-IV

## Other documents for verification

Type	Suffix
Call Center call Log	-CallLog
Carta Borrower Stock Ownership	-CartaBorrowerStockOwnership
Carta Secured Collateral	-CartaSecuredCollateral
Education-related documents	-ED
Voided check	-VC

## Trusted 3rd-party documents

Type	Suffix
Alloy Identity Decisioning Platform	-ALY
Bureau Van Dyke - Review	-BVDRewiew
Clarity	-CTY
Cognito Identity Verification and Watchlist Screening	-CGNTIDWS
Comply Advantage	-CmplyAdvt
DecisionLogic Income Verification	-DcsnLgcIV
Ekata account opening	-EkataAccount
Equifax eID	-EID
Experian PreciseID	-PID
GBG PLC Identity Data Verification	-GBGDataVerification
GIACT	-GCT
GIACT gAuthenticate	-GAuth
GIACT gIdentity	-GI
GIACT gVerify	-GV
idiCORE	-idiCORE
Innovis Identity Verification Solutions	-Innovis
Inscribe fraud document detection	-Inscribe
LexisNexis	-LN
LexisNexis Emailage	-LNEmail
LexisNexis Fraud Intelligence	-LNFraudIntel
LexisNexis FraudPoint	-LNF
LexisNexis InstantID	-LNI
LexisNexis OFAC	-LNO
LexisNexis RiskView Credit Solutions	-LNRiskView
LexisNexis Small Business Credit Report with SBFE Data	-LNSmallBusinessCRSBFE



Type	Suffix
Middesk Verification	-MiddeskVerification
Onfido Facial Similarity	-OFS
Oscilar for ID verification and fraud prevention	-Oscilar
PassFort	-PassFort
Persona Verifications	-Persona
PinWheel Employment Verification	-PEV
Plaid Auth Bank Accounts	-PlaidAuth
Prove Trust Score	-ProveTrustScore
SentiLink eCSBV service	-eCSBV
SentiLink Synthetic-Fraud-Scores	-SNTLF
Socure	-SO
Thomson Reuters Clear ID	-TRCID
TransUnion TLO	-TLO
TransUnion IDVision	-IDV
TransUnion TruValidate	-TVAL
Yodlee	-YOD

## Sample request and response- POST/Loan/{Id}/Attachments

The screenshot displays a REST client interface. At the top, a dropdown menu is set to 'POST' and the URL is 'https://'. A blue 'Send' button is on the right. Below this, tabs for 'Params', 'Authorization', 'Tests', and 'Settings' are visible. Under 'Params', the 'form-data' radio button is selected. A table lists parameters with a checked checkbox, key 'file', and value 'testDocs.zip'. The 'Body' tab is active, showing a JSON response: `{ "response": "File uploaded succesfully." }`. The status bar indicates 'Status: 200 OK'.

KEY	VALUE
<input checked="" type="checkbox"/> file	testDocs.zip X
Key	Value

Body Cookies (3) Headers (19) Test Results Status: 200 OK Save

Pretty Raw Preview Visualize JSON

```
1 {  
2   "response": "File uploaded succesfully."  
3 }
```

## Required parameters

- For multiple types of any single document listed above, add the numerical value to the name.  
**Example:** For multiple paystubs (PS): 179317-PS1 and 179317-PS2
- You can submit an additional 99 files as needed for underwriting, by following the convention, F1 - F99.
- If you want to submit an --F3 file, notify CR ahead of time so we can identify F3 as a security video.  
**Example:** MyRandomFile-F3.mov

## FAQs

### Loan stuck in the status Received

A loan stays in Received for these reasons:

- The file wasn't named correctly based on regex verification
- Some or all required documents weren't received
- There was a problem unzipping the files. If a file is corrupt, Arix won't be able to unzip it. Fix it and resend a POST with the attachments.

## Confirming document status

To confirm the document status for a loan, call [GetLoanDetail](#) and use these collections:

- `RequiredDocs` - to see which documents are required for a loan and if they were received.
- `Attachments` - to see which documents were received by Arix

## Adding loan documents

You can add documents if a loan is in one of the statuses below. If a document is sent after the loan is funded, Arix will store it, but it won't parse it or run it through compliance.

Status	Description
1	Received
2	DocsComplete
3	PassedCompliance
4	Approved
11	AwaitingFunding
101	ComplianceFailed

## 7.5. Get loan details by ID

---

### GET /LoanDetail/{Id}

Use this call to retrieve specific collections for a specific loan (by ID). You can retrieve all of the collections, or narrow down your request to retrieve only specific collections.

Request URL `https:// ... /LoanDetail/{Id}`

The API response always returns the `OriginalLoanInfo` (original details that were entered when the loan was created), and can also return other information. Therefore, if you made changes to a loan, it's easy to view and compare any of the changes you made to the loan.

To check the collections you've called, look at the **Fields** in the path parameter that show after the equal (=) sign.

### Example

In the response below, only results for `FundingAttempts` and `ProcessedRails` are returned, which in this case doesn't return any responses.

```
GET /LoanDetail/{Id}?Fields=FundingAttempts,ProcessedRails
```

### Loan collections

A loan collection contains a specific set of results, in response to a query that you add to your call.

Loan collections	Description
<div>LoanInfoUpdate</div> <div>s</div> <div>array</div>	<p>View all updates made to the loan.</p> <p>To view a full list of the parameters returned, see <div>LoanInfoUpdates</div> below.</p>
<div>FundingInfoUpdate</div> <div>s</div> <div>array</div>	<p>View all funding updates made to the loan.</p> <p>To view a full list of the parameters returned, see <div>FundingInfoUpdates</div> below.</p>
<div>FailedLoanInfoUpdate</div> <div>array</div>	<p>View information about a loan that failed. To view a full list of the parameters returned, see <div>FailedloanInfoUpdate</div> below.</p>
<div>FailedFundingInfoUpdate</div> <div>array</div>	<p>View information about the failed loan funding. To view a full list of the parameters returned, see " <div>FailedFundingInfoUpdate</div> ** below.</p>
<div>ProcessedRails</div> <div>array</div>	<p>View the rails that were processed for a loan. To view a full list of the parameters returned, go to <a href="#">Loan funding - payment rails</a></p>
<div>FundingAttempts</div> <div>array</div>	<p>View the number of times funding was attempted. To view a full list of the parameters returned, see <div>FundingAttempts</div> below.</p>
<div>ReturnedRails</div> <div>array</div>	<p>View the rails that were returned. To view a full list of the parameters returned, go to <a href="#">Loan funding - payment rails</a></p>
<div>StatusUpdates</div> <div>array</div>	<p>View the status of the loan.</p> <p>To view a full list of the parameters returned, go to <a href="#">Status updates</a> .</p>
<div>Attachments</div> <div>array</div>	<p>View the attachments that were submitted, or those that are missing.</p> <p>To view a full list of the parameters returned, go to <a href="#">Document reference</a>.</p>
<div>RequiredDocs</div> <div>array</div>	<p>View the documents that are required for the loan. To view a full list of the parameters returned, go to <a href="#">Document reference</a>.</p>
<div>ComplianceLoanResult</div>	<p>View the compliance status of the loan. To view a full list of the parameters returned, go to <a href="#">Compliance results</a>.</p>

Loan collections	Description
array	
AdjustmentActions array	View updates made to the loan after funding. To view a full list of the parameters returned, go to <a href="#">Adjustment action parameters</a> .

## RequiredDocs collection parameters

This collection returns information about Required Documents and their received status for this loan.

Attribute	Description
DocumentType	One of the documents required for loan processing
Received	Confirms if a particular document was received
ReceivedDate	Date the document was received

## Status Update parameters

Attribute	Description
LoanId string	Unique identifier for a loan
Status string	Loan status. For more information, see <a href="#">Enums</a> .
TimeStamp string	Timestamp for the update
DateInserted string	Timestamp when the status was entered (internal)

## ComplianceLoanResult attributes

This section displays the compliance results of a loan.

Attribute	Description
ResultDate	Date of the result for the compliance check
ComplianceRuleResultList	List of compliance rules, specific to each MPL

## ComplianceRuleResult parameters

These are the fields of a compliance result of a loan for each specific rule. If a rule failed, you can use these query parameters to check the Data against the Rule to see why it failed.

Parameter	Description
RuleName string	Name of the compliance rule. Specific for every MPL.
Rule string	Description of the compliance rule
Data string	The actual data that was received for this loan in regards to this rule.
Result string	Result of the compliance check. True = Passed False = Failed

You can run a loan through compliance a second time with one of these calls:

- [\*\*PUT/Loan{Id}/FundingInfo\*\*](#) (to update funding information)
- [\*\*PUT/Loan/{Id}\*\*](#) (to update loan information)
- [\*\*POST/Loan/{Id}/Attachments\*\*](#) (to add attachments)

## FundingAttempts

This section displays the results of loan funding.

Attribute	Description
<div>Id</div> <div>string</div>	Unique identifier for the loan funding information
<div>LoanId</div> <div>string</div>	Unique identifier for the loan
<div>Time</div> <div>string</div>	Time that the information was returned
<div>TransactionGroupId</div> <div>string</div>	ID for the transaction group
<div>FundingResult</div> <div>string</div>	Indicates if the funding was successful
<div>AmountFunded</div> <div>number</div>	Indicates the amount funded

## LoanInfoUpdate collection parameters

This collection returns information about any updates made to a loan.

Attribute	Description
<div>LoanInfoUpdates</div>	Unique identifier for the loan update
<div>UpdatedBy</div> <div>string</div>	Clerk who updated the loan
<div>UpdatedOn</div> <div>DateTime</div>	The date the loan was updated

## FundingInfoUpdate collection parameters

This collection returns information about any funding updates made to a loan.



Attribute	Description
<div>LoanId</div> <div>string</div>	Unique identifier for the loan
<div>MPLId</div> <div>string</div>	ID of the MPL
<div>UpdatedBy</div> <div>string</div>	Clerk who updated the loan
<div>UpdatedOn</div> <div>DateTime</div>	The date the loan was updated
<div>Rails</div> <div>List{RailTransaction}</div>	Click <a href="#">Loan funding - payment rails</a> for a list of all rail types
<div>TransactionGroupId</div> <div>string</div>	Group ID for the transaction
<div>FundingResult</div>	List of possible funding results. <ul style="list-style-type: none"> <li>- Success</li> <li>- PartialFailure</li> <li>- FundingFailed</li> </ul> See <a href="#">Enums</a> for more information.
<div>AmountFunded</div> <div>decimal</div>	Dollar amount funded

## FailedLoanInfoUpdate collection parameters

This collection returns information about loan funding that failed.


Attribute	Description
FailedLoanInfoUpdateId	Unique identifier of information about the failed loan
UpdatedBy string	Clerk who updated the loan
UpdatedOn DateTime or null	The date the loan was updated
ErrorMessages string	Error message

## FailedFundingInfoUpdate collection parameters

This collection returns information about loan funding that failed.

Attribute	Description
ErrorMessages string	Error message

## Sample request in cURL - GET /LoanDetail/{Id}


Get loan details
Copy icon

```
curl --location --request GET 'https://arixapisandbox.crbnj.net/LoanDetail/f8' \
--header 'Accept: application/json' \
--header 'Authorization: <BearerToken>' \
--header 'Cookie: ss-opt=temp; ss-id=u1Hu7zc2o0fCcYrj041C; ss-pid=64sRjkZ0PyM' \
--data-raw ''
```

## Sample response in JSON - GET /LoanDetail/{Id}





```
{
  "OriginalLoanInfo": {
    "Id": "f83f9c6f-c187-4c53-a1c2-ac5800f1626d",
    "MPLId": "TST",
    "Date": "/Date(1603031931240-0000)/",
    "LoanNumber": "DJ-98dcf5dcb2c5407",
    "IssuingBankId": "CRB",
    "Platform": "string",
    "LoanType": "HFS",
    "NoteDate": "/Date(1577768400000-0000)/",
    "BorrowerLastName": "Jacobi",
    "BorrowerFirstName": "Leonie",
    "BorrowerDOB": "/Date(536389200000-0000)/",
    "BorrowerSSN": "841792974",
    "BorrowerAddress": "703_Kacey_Court",
    "BorrowerCity": "New",
    "BorrowerState": "GU",
    "BorrowerZip": "78383",
    "LoanAmount": 500.000000,
    "NetFunding": 1300.000000,
    "Rate": 0.050000,
    "APR": 10.000000,
    "BatchId": "string",
    "BorrowerPhone": "919-959-8725",
    "BorrowerEmail": "Lilyan_Hackett@gmail.com",
    "Amortization": 0,
    "Term": 36,
    "RateType": 0,
    "PriorLoanFlag": "string",
    "FICO": 10,
    "FICODate": "/Date(1514696400000-0000)/",
    "CreditGrade": "string",
    "AssetClass": 0,
    "LoanPurpose": "string",
    "Investor": "string",
    "Servicer": "string",
    "ApplicationDate": "/Date(1603031931177-0000)/",
    "RegBDecisionDate": "/Date(1603031931177-0000)/",
    "PurchaseDate": "/Date(1603252800000-0000)/",
    "PaymentDueDate": "/Date(1608267600000-0000)/",
    "ApprovedLoanAmount": 0.000000,
    "PeriodicPayment": 0.000000,
    "OriginationFee": 100.000000,
    "HomeownerFlag": "string",
    "OpenCreditLines": 0,
  }
}
```

```
"AnnualIncome": 0,
"CreditInquiries12months": 0,
"DQPast24months": 0,
"PublicRecordsOnFile": 0,
"EmploymentLength": 0,
"DebtUtilization": 0.000000,
"TotalRevolvingDebt": 0.000000,
"AccountsOpenedPast24months": 0,
"CollectionsExcludingMedical": 0,
"MonthsSinceLastRecord": 0,
"Employer": "string",
"DTI": 0.000000,
"MLAFLAG": "string",
"MAPR": "string",
"ReceiverName": "string",
"SameDayFlag": 0,
"AddendaRecord": "string",
"CoBorrowerFirstName": "string",
"CoBorrowerLastName": "string",
"CoBorrowerDOB": "/Date(536389200000-0000)/",
"CoBorrowerSSN": "string",
"CoBorrowerAddress": "string",
"CoBorrowerCity": "string",
"CoBorrowerState": "string",
"CoBorrowerZip": "string",
"CoBorrowerPhone": "string",
"CoBorrowerEmail": "string",
"CoBorrowerFICO": 0,
"MerchantName": "string",
"MerchantFee": 0.000000,
"RefundAmount": 0.000000,
"AdjustedLoanAmount": 0.000000,
"AdjustedNetFunding": 0.000000,
"AdjustedOriginationFee": 0.000000,
"AdjustedCurrentMonthlyPayment": 0.000000,
"RefundDate": "/Date(1514696400000-0000)/",
"AdjustedMerchantFee": 0.000000,
"FinalPaymentDate": "/Date(1618718400000-0000)/",
"FinalPayment": 0.000000,
"FinanceCharge": 0.000000,
"TotalPayments": 0.000000,
"PPY": 0,
"PaymentFrequency": 0,
"StandardEntryTypeCode": "string",
"InterestAccrualMethodDays": 0,
"SubpoolId": "Moe83",
```

```
"Program": "DTM",
"Field1": "string",
"Field2": "string",
"Field3": "string",
"Field4": "string",
"Field5": "string",
"Field6": "string",
"Field7": "string",
"Field8": "string",
"Field9": "string",
"Field10": "string",
"Field11": "string",
"Field12": "string",
"Field13": "string",
"Field14": "string",
"Field15": "string",
"Field16": "string",
"Field17": "string",
"Field18": "string",
"Field19": "string",
"Field20": "string",
"Rails": [
  {
    "ACHFields": {
      "StandardEntryClassCode": "PPD",
      "TransactionType": "push",
      "ToRoutingNumber": "021214891",
      "ToAccountType": "GL",
      "ToAccountName": "Test",
      "ToAccountNumber": "9999999999",
      "Description": "Test"
    },
    "Id": "e6901e4f-75e3-4e86-a94f-ac5800f1626d",
    "RailType": "COSACH",
    "Priority": 1,
    "Amount": 300.0,
    "LoanId": "f83f9c6f-c187-4c53-a1c2-ac5800f1626d",
    "MPLId": "TST",
    "IssuingBankId": "CRB",
    "IsFailed": false,
    "LoanType": "Undefined"
  }
],
"LoanInfoUpdates": [],
"ProcessedRails": [],
```

```
"FundingAttempts": [],
"ReturnedRails": [],
"StatusUpdates": [
  {
    "Id": 1635439,
    "LoanId": "f83f9c6f-c187-4c53-a1c2-ac5800f1626d",
    "Status": "Received",
    "TimeStamp": "/Date(1603031931243-0000)/",
    "DateInserted": "/Date(1603031931327-0000)/"
  }
],
"Attachments": [],
"RequiredDocs": [
  {
    "DocumentType": "LoanDocument",
    "Received": false
  }
],
"ComplianceLoanResult": {
  "ResultDate": "/Date(-621355968000000-0000)/",
  "ComplianceRuleResultList": []
}
```



## 7.6. Loan funding - payment rails

---

A payment rail is the method by which Arix transfers funds. Arix uses these rails:

- [ACH / Same Day ACH](#)
- [Physical checks](#)
- [RPPS](#)
- [Internal transfer \(XPay\)](#)
- [RTP](#)
- [Card](#)
- [Wire](#)

### IMPORTANT

To track rails, use the RailId you receive back after the API call.

Funding is an automated process.

It is your responsibility to review your records before you create any loans or requests for funding.

## PUT /Loan/{Id}/FundingInfo

Use this call to add new payment rails for a particular loan. This API call replaces rails that haven't been successfully funded.

Request URL `https:// .../Loan/{Id}/FundingInfo`

You can use this call to:

- Attempt disbursements to the borrower in case of returns or failures
- Make multiple disbursements, one or more rails, as needed
- Add rails

Funds can be sent as one net sum or in multiple disbursements. Arix automatically calculates the sum of the loan and adjusts the amount down (if necessary) so that no funds that exceed the approved loan amount are sent.

### IMPORTANT

When you send a PUT call to update loan information, already processed rails and rails that are in the process of being funded aren't affected.

## Sample request in JSON - PUT /Loan/{Id}/FundingInfo

In this sample request, we used this API to request one disbursement of \$7,000 using the RPPS rail.

JSON



```
{
  "FundingInfoUpdate": {
    "Rails": [
      {
        "RailType": "RPPS",
        "Priority": 1,
        "Amount": 7000,
        "RPPSFields": {
          "BillerID": "132654654632135423452",
          "BillerName": "Anita Loan",
          "ConsumerNameAtBiller": "TestNameAtBiller",
          "ConsumerAccountNumberAtBiller": "5454546498195454"
        }
      }
    ]
  }
}
```

## Sample response in JSON - PUT /Loan/{Id}/FundingInfo

```
"Rails": [  
  {  
    "RPPSFields": {  
      "BillerID": "132654654632135423452",  
      "BillerName": "Anita Loan",  
      "ConsumerNameAtBiller": "TestNameAtBiller",  
      "ConsumerAccountNumberAtBiller": "5454543427795454"  
    },  
    "Id": "a974ce4a-e7b3-4310-bc2b-ac6900e79b1a",  
    "RailType": "RPPS",  
    "Priority": 1,  
    "Amount": 7000,  
    "LoanId": "ca485557-20c3-4a5b-b123-ac6900da4728",  
    "MPLId": "TST",  
    "IsFailed": false,  
    "LoanType": "Undefined"  
  }  
]
```

## RailTransaction attributes

These properties are returned for rail transactions.

Every rail includes a **GUID**, which is a unique identifier for the rail which you can use to match the rail request to the rail response.

Attribute	Description
<div>RailId</div> GUID	Unique identifier for a rail
<div>LoanId</div> string	Unique identifier for a loan
<div>RailType</div> <u>RailType</u> or null	Type of rail for payment Select the RailType Enum: <ul style="list-style-type: none"> <li>• <u>COSACH</u></li> <li>• <u>COSACHSD</u></li> <li>• <u>AdjustmentAction</u> (Not a payment rail)</li> <li>• <u>Check</u></li> <li>• <u>RPPS</u></li> <li>• <u>Internal Transfer (XPay)</u></li> <li>• <u>RTP (Instant Payments)</u></li> <li>• <u>Wire</u></li> <li>• <u>Card</u></li> </ul>
Priority integer or null	Order the rails should be processed - starts with number 1. If all the rails have the same number, Arix processes them in the order they were received.
Amount decimal	Amount attempted to process on this rail. If left blank, Arix attempts to process the full balance.
<div>MPLId</div> string	ID of the MPL
<div>IssuingBankId</div> string	ID of the issuing bank
<div>IsFailed</div> boolean	Indicates if the loan transmission failed. True or false.

Attribute	Description
<div>LoanType</div> <div>string</div>	Type of loan
<div>AmountFunded</div> <div>decimal</div>	Dollar amount funded
<div>RailTransactionId</div> <div>string</div>	Unique identifier for a rail transaction
<div>FundResultRailResultType</div> <div>Result of the funding. See <a href="#">Enums</a>.</div>	
<div>Message</div> <div>string</div>	Message returned
<div>ProcessedAt</div> <div>DateTime</div>	Date the rail was processed
<div>TransactionGroupId</div> <div>GUID</div>	Unique identifier of the transaction group

## Rail cutoff times

The bank reserves 1.5 - 2 hours for pre-processing of funding before the network cutoff times below. Only loans that completed all underwriting checks are funded. Loans that miss a cutoff will auto-roll to the next processing window.

Rail type	Cutoff times
ACH	Cutoff is 16:00 EST. The Fed accepts submissions until 18:00 EST.
ACH SD (Same Day) Limited to \$1,000,000 per transaction	Cutoff is 13:00 EST. The Fed accepts submissions until 14:45 EST. Extended Same Day cut off is 14:30 EST. The Fed accepts submissions until 16:45 EST.
Cards	24/7 (no cutoff times)
Checks	Cutoff is 11:00 EST
Mastercard RPPS	Support for 3 MC RPPS processing cycles: 03:00 EST – 05:00 EST 13:00 EST – 15:00 EST 21:00 EST – 23:00 EST To have a transaction included in a processing cycle, it has to be submitted no later than 50 minutes before the cutoff. Any transactions submitted later than 50 minutes before the cutoff time, will flow to the next cycle.
Internal Transfer (XPay)	24/7 (no cutoff times)
RTP (Real Time Payment) Limited to \$1,000,000 per transaction	24/7 (no cutoff times)
Wires	Cutoff is 17:00 EST. Wires are released to the Fed once an hour. The last release is at the cutoff time.

## ACH return cutoff times

ACH returns are received in batches, with the following cutoff times:

## **ACH returns**

- 1. First Batch: 12:00 AM EST**
- 2. Second Batch: 3:00 AM EST**
- 3. Third Batch: 11:30 AM EST**
- 4. Fourth Batch: 3:30 PM EST**
- 5. Fifth Batch: 4:30 PM EST**

## 7.6.1. ACH parameters

---

Arix uses ACH to push money into the account of a borrower, vendor, or creditor. CR must approve every ACH payment that is sent.

There are two types of ACH rails:

### **COSACH**

- Via Nacha
- Funded next day

### **COSACHSD**

- Via Nacha
- Same day funding



```
{
  "LoanId": "bbedd717-8589-4411-b6da-ac6e0097bf51",
  "Rails": [
    {
      "ACHFields": {
        "StandardEntryClassCode": "PPD",
        "TransactionType": "push",
        "ToRoutingNumber": "021214891",
        "ToAccountType": "Checking",
        "ToAccountName": "Anita Lon",
        "ToAccountNumber": "9999999999",
        "Description": "Loan"
      },
      "Id": "23b5fedb-bc20-41a0-a4ae-ac9300b4530e",
      "RailType": "COSACH",
      "Priority": 1,
      "Amount": 100
    }
  ]
}
```

## Request parameters

Attributes	Description
<div>ToRoutingNumber</div> <div>string</div>	<p>Receiving bank's routing number.</p> <p>The value entered for this field is validated up front and it must be 9 digits long.</p> <p><b>Required</b></p>
<div>ToAccountName</div> <div>string</div>	<p>Name on the customer account.</p> <p>Limited to 22 characters.</p> <p><b>Required</b></p>
<div>ToAccountNumber</div> <div>string</div>	<p>Account number.</p> <p>Limited to 17 digits.</p> <p><b>Required</b></p>
<div>Description</div> <div>string</div>	<p>Mapped to identification field in the NACHA file.</p> <p>Limited to 15 characters.</p> <p><b>Required</b></p>
<div>StandardEntryClassCode</div> <div>string</div>	<p>The type of account:</p> <ul style="list-style-type: none"> <li>- PPD - consumer accounts</li> <li>- CCD - corporate accounts</li> </ul> <p><b>Required</b></p>
<div>TransactionType</div> <div>string</div>	<p>Push</p> <p><b>Required</b></p>
<div>ToAccountType</div> <div>string</div>	<p>Checking account or savings account.</p> <p><b>Required</b></p>

## 7.6.2. ACH returns

---

### Returned rail

ACH credits aren't final as are wires. When Arix uses an ACH rail to process a loan, the rail can be returned from the Fed at any point in the . Returns typically occur within several days of funding, but they can be generated up to 180 days from the funding date.

Arix uses webhooks to update the funding status of a loan.

### What happens in Arix when a rail is returned

#### Rail status

When a rail is returned, the status of the rail is automatically updated to **Returned**.

#### Loan status

The loan status changes based on these conditions:

- **Returned**

If all rails were returned, then the loan automatically moves into the status, `Returned`, regardless of the previous loan status.

- **NotFullyFunded**

If at least one rail was returned, the loan moves into the status, `NotFullyFunded`. See the [enums](#) reference document for a list of the loan and rail statuses.

### Webhooks

These webhooks are triggered for the rail and loan statuses:

- `LoanStatusUpdated` (if the return causes the loan status to change)
- `RailUpdated`

## Loan funding

The loan funding information is updated accordingly in these fields:

- `ProcessedRails`
- `ReturnedRails`

## Investigating returned rails

You can call [Get/LoanDetails](#) to view the details on a returned rail by including returned rails in the field request. The update includes the NACHA return reason code, the return trace number, and the original trace number.

## Resubmitting a returned rail

To resubmit a returned rail, call PUT/FundingInfo with the correct details.

### 7.6.3. Adjustment action parameters

---

You can update the loan information after a loan was partially funded. Update the relevant parameters from the list below. Any loan funding information that you changed, shows with the word Adjusted before it.

Attributes	Description
<div>AdjustmentActionType</div> <div>LoanAdjustmentAction</div>	Type of update made to the loan <b>Required</b>
<div>ActionAmount</div> <div>decimal</div>	Dollar amount of the change to the loan <b>Required</b>
<div>Description</div> <div>string</div>	Description of the reason for the loan adjustment. 1 - 30 characters. <b>Required</b>
<div>UpdatedBy</div> <div>string</div>	By whom the information was updated Optional
<div>UpdatedOn</div> <div>string</div>	Date/ time the information was updated Optional
<div>LoanId</div> <div>string</div>	Unique identifier of the loan Optional
<div>AdjustedLoanAmount</div> <div>decimal <i>or null</i></div>	Adjusted gross dollar amount requested. Optional
<div>AdjustedNetFunding</div> <div>decimal <i>or null</i></div>	Adjusted net dollar amount to fund Optional
<div>AdjustedOriginationFee</div> <div>decimal <i>or null</i></div>	Adjusted charge to the borrower for the loan Optional
Optional <div>AdjustedCurrentMonthlyPayment</div> <div>decimal <i>or null</i></div>	Adjusted monthly payment amount Optional
<div>AdjustedFinalPaymentDate</div> <div>DateTime <i>or null</i></div>	Adjusted date due for final loan payment. The format must be: yyyy-mm-dd. Optional
<div>AdjustedFinalPayment</div> <div>decimal <i>or null</i></div>	Adjusted dollar amount of final payment Optional

Attributes	Description
AdjustedFinanceCharge decimal <i>or null</i>	Adjusted finance charge paid by the borrower Optional
AdjustedTotalPayments decimal <i>or null</i>	Adjusted total paid by the borrower after all the payments have been made Optional
AdjustedAPR decimal <i>or null</i>	Adjusted annual percentage interest rate, as a percent Optional

JSON
<pre>{   "FundingInfoUpdate":{     "Rails":[       {         "AdjustmentActionFields":{           "AdjustmentActionType":"Prepayment",           "ActionAmount":300,           "Description":"customer payment"         },         "RailType":"AdjustmentAction",         "Priority":1,         "Amount":300.0       }     ]   } }</pre>

## 7.6.4. Check parameters

---

To pay down loan proceeds, you can transfer funds by sending a physical check to the borrower or a biller (merchant or business). This is useful when you have to pay billers that don't accept payments via Mastercard RPPS.

You can add an optional Base64 encoded attachment to accompany the check, and it will be mailed as a PDF.

### Check delivery

After a check is issued, it takes about 5 - 7 business days for receipt (CR uses USPS first class mail). If a check isn't cashed after 15 calendar days, it is canceled.

You can contact MPL Ops to stop payment on a check, and for any questions regarding check delivery or receipt.

Webhooks are reported when a check is processed, mailed, cashed, or expired. For more information on webhooks, see [View Webhook events available](#) .

#### IMPORTANT

When you add a date field in a request, the format must be: yyyy-mm-dd, such as 1970-12-05.

### Request parameters



```
{
{
  "FundingInfoUpdate": {
    "Rails": [
      {
        "CheckFields": {
          "PhysicalAddressOfRecipient": {
            "Line_1": "Anita Loan",
            "Line_2": "Strawberry Fields Lane",
            "City": "Paradise City",
            "State": "NY",
            "Zip": "11252",
            "Country": "USA"
          },
          "Name": "Anita",
          "Attachment": "{Encoded64 text here}",
          "Description": "Loan from Cross River",
          "Remittance_advice": "",
          "IsPrintedandMailed": "0",
          "CheckEmail": "anita@gmail.com"
        },
        "RailType": "CHECK",
        "Priority": 1,
        "Amount": 200.0
      }
    ]
  }
},
```

## Request parameters

Attribute	Description
<div>Rails</div> array	Rail selected, in this case <div>Check</div>
<div>Line_1</div> string	Check delivery address, line 1 60 character limit if line_2 is provided 120 characters max length, if line_2 is not provided <b>Required</b>
<div>Line_2</div> string	Check delivery address, line 2 60 character limit Optional
<div>City</div> string	City of business for check delivery <b>Required</b>
<div>State</div> string	State of business for check delivery. 2 letters. <b>Required</b>
<div>Zip</div> string	ZIP code of business for check delivery <b>Required</b>
<div>Country</div> string	Recipient's country. Always US. 2-letter country code. Optional
<div>Name</div> string	Name of the borrower 48 character limit <b>Required</b>
<div>Description</div> string	The name of the loan being paid. This is printed on the check and looks best if under 100 chars. Optional
<div>Attachment</div> string	Base64 encoded (see below 1 page PDF) It must be encoded and inserted here as text. Optional
<div>Remittance_advice</div>	Base64 encoded text.

Attribute	Description
array	Up to 700 chars before encoding. Appears on the top third of the page. The check is located on the bottom third of the page. Optional
<code>IsPrintedandMailed</code> boolean	Whether the check is being printed and mailed. True if check is being printed and mailed, otherwise, false. <b>Required</b>
<code>CheckEmail</code> string	Recipient's email Optional
<code>Priority</code> integer	Rail priority <b>Required</b>
<code>Amount</code> integer	Amount being funded <b>Required</b>

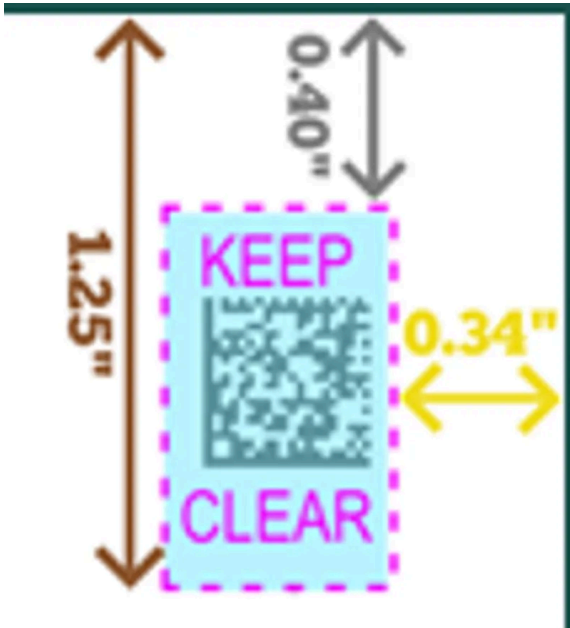
## Attachments

If your team utilizes our inserts with checks feature, please make sure you follow the requirements listed below (see `Attachment` attribute). Inserts must be in PDF format.

### These are the requirements for creating your PDF:

- **TrueType fonts must be used.**
- 8.5 × 11 margins are required. Any margins that are smaller or larger than this will not run properly through our programs.
- PDF must be created by Adobe Version 1.4 or better.
- PDF must be in portrait orientation. Inserts oriented in landscape may not print correctly.
- All security on PDF must be removed. PDF can not have edit restriction locks and can not be password protected.

- PDF must be able to display the full page on “actual size” print settings. If an insert is too large to print on the “actual size” setting, the insert will not print properly and the page may be cut off.
- The insert must have all settings embedded/flattened. Fillable form PDF’s that are not embedded cannot be processed correctly and will be rendered blank.
- The blue shaded area shown below should be kept clear. This area is where 2D barcodes are placed.



## 7.6.5. RPPS parameters

You can use the RPPS rail in Arix to electronically transfer funds to pay loan proceeds directly to billers. Borrowers who select to use this rail, can have funds sent directly to billers to pay down their outstanding bills, most typically used to pay down outstanding credit card bills.

The rail uses Mastercard's RPPS rail system. Arix sends an API call to the CR RPPS gateway, which in turn sends a request to Mastercard to transfer funds. Mastercard then sends a credit to the biller.

RPPS requests are batched daily by the CR RPPS gateway when the system sends the batched requests to Mastercard for processing.

The borrower's credit card doesn't have to be a Mastercard. It can be any card issued by a biller that signed up to receive bill payments via the RPPS service.

JSON



```
{
  "Rails": [
    {
      "RailType": "RPPS",
      "Priority": 1,
      "Amount": 900,
      "RPPSFields": {
        "BillerID": "1234567890",
        "BillerName": "Anita Loan",
        "ConsumerNameAtBiller": "TestNameAtBiller",
        "ConsumerAccountNumberAtBiller": "5454546498195454"
      }
    }
  ]
}
```

## Request parameters

Attribute	Description
<div>BillerID</div> string	The 10 character ID number of the credit card issuer (biller). If length(BillerID) < 10, you can pad with zeroes. Example: 0000012345 <b>Required</b>
<div>ConsumerNameAtBiller</div> string	The borrower's cardholder name on the credit card account <b>Required</b>
<div>ConsumerAccountNumberAtBiller</div> string	The borrower's credit card account number on the biller's credit card <b>Required</b>
<div>BillerName</div> string	The name of the credit card company that is making the disbursement Optional

## Biller directory

There are 2 ways to do directory lookup:

1. CR can deliver a daily CSV of the RPPS biller directory into your SFTP folder.
2. MC RPPS biller validator API <https://developer.mastercard.com/product/bill-payment-validator/>. Cross River can set up credentials for you.

## RPPS sandbox simulation

You can simulate rails in the sandbox. The table indicates the responses that you should receive for the transaction amounts listed.

Amount	Result
Up to \$3	Result is submitted
Above \$3 to \$4	Result is rejected
Above \$4 to \$5	Result is approved After 30 seconds it becomes returned
Above \$5	Result is approved

## PCI compliance

Payment Card Industry (PCI) standards ensure that sensitive credit card information is protected from fraudulent use. Arix processing complies with PCI standards.

When an RPPS transaction is initiated, Arix receives a request that includes the borrower's credit card number. This number is sent to a tokenization service that is Level 1 PCI-compliant. They send a token back to Arix, which stores the token.

When processing the RPPS batch, Arix calls to detokenize all the credit card numbers, and sends the credit card numbers in a file to Mastercard.

Arix logs contain only masked credit card numbers.

## 7.6.6. RPPS returns

---

RPPS rails are approved immediately after funding and the rail is marked Success.

If however, an RPPS is returned, these are the guidelines:

- If, within 5 days of the rail attempt RPPS returns a result of Rejected or Failed, Arix marks the rail as Rejected.
- If, within 5 days of the rail attempt, RPPS returns a result of Returned, Arix marks the rail as Returned.
- If 5 days pass without a Reject, **Failed** or Returned result from RPPS, Arix marks the rail as Success.



## 7.6.7. RPPS transaction reversals

---

When requesting a reversal of an RPPS transaction, follow this process:

1. Check whether the biller allows reversal using the **reqAdndaRevm** tag that you can find in the XML biller directory.
2. If the biller allows reversal, then send an email to [iprops@crossriver.com](mailto:iprops@crossriver.com) and [mplops@crossriverbank.com](mailto:mplops@crossriverbank.com) as follows:
  - Subject: RPPS REVERSAL REQUEST
  - Content:
    - Biller ID
    - Account with biller (first 6 digits & last 4 digits)
    - Name on account
    - Reversal amount

### Note:

### Sending reversals

In accordance with Mastercard RPPS Rules and Obligations, the following rules pertain to sending reversals: Participant may send a reversal transaction to a Biller that accepts reversals within five Mastercard Remote Payment and Presentment Service® (RPPS) processing days of the payment that was previously sent. Participant may initiate a reversal only for actual payment transactions previously sent from the Participant to the Biller through the Mastercard Remote Payment and Presentment Service® (RPPS) network. Participant shall indemnify the affected Participant and Mastercard Remote Payment and Presentment Service® (RPPS) from and against any and all claims, demands, losses, liabilities, or expenses, including attorney fees and costs that result directly or indirectly from the debiting or crediting of the reversal transaction.

Participant shall notify the Receiver prior to submitting a reversal transaction and advise them of the reason for such reversal prior to submitting reversals.

## **RPPS credit and debit cap management**

Debit caps limit the daily amount of reversals that a Concentrator/Biller will accept from an Originator. For example, a concentrator/biller may only reverse \$2500 per day even if the original payment was \$10,000. These caps minimize risk to Mastercard and the RPPS settlement banks by containing exposure

with respect to daily settlement positions. Debit caps limit the daily debit amount an Originator may send to a Receiver to reverse payments.

## **Good faith request**

Good Faith requests are for items that have expired for reversals past the five day time frame or for billers who accept reversals more than the daily debit cap as set by the biller/concentrator. Ops will contact the biller to make a sincere request to repay the transaction's proceeds. When processing reversals, many institutions adhere to their own internal procedure. Reversals may take six months or longer to recover, though this is not always the case. Submitting good faith request returns are not guaranteed as the decision solely lies on the receiving institution. To submit a good faith request – send an email to [IPROps.support@crossriver.com](mailto:IPROps.support@crossriver.com) with the name, amount and disbursement date.

RPPS requests are batched daily by the Cross River RPPS gateway when the system sends the batched requests to Mastercard for processing. If duplicate transactions are detected at Mastercard, the system automatically rejects the duplications.

## 7.6.8. Internal transfer (XPay) parameters

---

Internal transfers funds between Cross River and an MPL account within the Cross River Operating System (COS). These transfers are executed internally within CR and are immediate.

Once funds are in the MPL account, they can use the funds in any of these use cases:

- Fund 3rd parties to make the final payment to the borrower
- ACH to merchants (Point-of-Sale payments)
- Recycle a loan for a repeat customer

JSON



```
{
  "LoanId": "bbedd717-8589-4411-b6da-ac6e0097bf51",
  "Rails": [
    {
      "XPayFields": {
        "ToAccountName": "",
        "ToAccountNumber": "2392433617",
        "Description": "A1B2C3"
      },
      "RailType": "COSXPay",
      "Priority": 1,
      "Amount": 7111.12
    }
  ]
}
```

## Request parameters

Attribute	Description
<div>ToAccountNumber</div> <div>string</div>	<p>The unique identifier of the receiver. This defaults to the COS account number.</p> <p><b>Required</b></p>
<div>Description</div> <div>string</div>	<p>A description of the transaction to be used by the partner to identify the transaction.</p> <p>Minimum length of 1, maximum length of 30.</p> <p><b>Required</b></p>
<div>ToAccountName</div> <div>string</div>	<p>Name of the account receiving the funds</p> <p><b>Required</b></p>

## 7.6.9. Instant payments parameters

### IMPORTANT

The RTP rail in Cross River is now called Instant Payments.

In Arix, the `RailType` Enum is still called RTP.

Instant Payments uses one of the 2 network platforms to send a payment directly to the account of a borrower, vendor, or creditor.

**Instant Payments** is a way for MPLs to send immediate fund transfers at any time of the day, including weekends and holidays.

Cross River uses two different Instant Payments network platforms: RTP via The Clearing House (TCH), and the Federal Reserve's FedNow platform.

- **RTP® via The Clearing House**. The Clearing House (TCH) provides Instant Payments using its RTP® system. All federally insured US depository institutions that participate in the RTP network can send payments from and receive payments to their accounts 24/7.
- **FedNow®**. The Federal Reserve's instant payment infrastructure allows financial institutions of every size across the U.S. to provide safe and efficient 24/7 instant payment services.

Arix uses **network interoperability** to deliver the payment to the creditor as efficiently and effectively as possible. The network used for sending the payment depends on the routing number specified, and on availability. The payment response includes the network platform the payment has been processed through.

All transactions occur in real-time and the payments are irrevocable.

## RTP directory

Although more than 65% of U.S. bank account holders are eligible to receive RTP funds disbursements, some are not. In order to avoid unnecessary charges, we recommend that

our partners use the [Get service info for specific FI \(directory\)](#) API endpoint to see which financial institutions can receive RTP credit transfers (CRDT). It's important to note, that even if the financial institution can receive RTP transfers, it does not guarantee that the receiver's account is eligible.

### Credentials for Directory

To use the RTP directory, you will need credentials for COS. These are separate credentials from the Arix API credentials. Contact your Relationship Manager if you need more information.

## Request body example

JSON



```
{
  "FundingInfoUpdate": {
    "Rails": [
      {
        "RailType": "RTP",
        "Priority": 1,
        "Amount": 900,
        "RTPFields": {
          "SubjectRouting": "021214891",
          "SubjectAccount": "9999999999",
          "SubjectName": "Test"
        }
      }
    ]
  }
}
```

## Request parameters

Attribute	Description
<div>SubjectRouting</div> <div>string</div>	<div>Receiving bank's routing number</div> <div>The value entered for this field is validated up front and it must be 9 digits long</div> <div>Required</div>
<div>SubjectName</div> <div>string</div>	<div>Name on the <div>SubjectAccount</div></div> <div>Limited to 22 characters</div> <div>Required</div>
<div>SubjectAccount</div> <div>string</div>	<div>Account number of the <div>SubjectAccount</div></div> <div>Limited to 17 digits</div> <div>Required</div>
<div>AdditionalDescription</div> <div>ion</div>	<div>A description of the payment</div> <div>Limited to 140 characters</div> <div>Optional</div>

## RTP payment queuing for offline participants

Although RTP participating banks have an SLA of 99.9% uptime, from time to time, a receiving bank will be unavailable to transact. This can be due to routine maintenance or unknown technical issues that are resolvable with time.

CR will queue the payment until the receiving bank becomes responsive or for up to 3 days. The RTP rail will be rejected if the payment hasn't been completed within 3 days.

If you'd like to cancel the rail before that, our operations team can cancel the payment for you.

## Reject simulation on sandbox

You can simulate receiving a reject response on the RTP rail.

Add the **reject:prefix** to the `SubjectName` as seen on line 11.

JSON



```
{
  "FundingInfoUpdate":{
    "Rails":[
      {
        "RailType":"RTP",
        "Priority":1,
        "Amount":900,
        "RTPFields":{
          "SubjectRouting":"021214891",
          "SubjectAccount":"9999999999",
          "SubjectName":"reject:Penny Pinchington"
        }
      }
    ]
  }
}
```



## 7.6.10. Card parameters

You can use the Card rail in Arix to pay out funds to the borrower's credit card. This rail is used to supplement the RPPS rail to get a wider coverage of credit card networks.

JSON



```
{
  "Rails": [
    {
      "RailType": "Card",
      "Priority": 1,
      "Amount": 1500,
      "CardFields": {
        "FirstName": "Chayav",
        "LastName": "Kesef",
        "CardNumber": "5454546498195454",
        "Cvv": "544",
        "ExpirationMonth": 6,
        "ExpirationYear": 26,
        "SubjectPhoneNumber": "19175554444",
        "SubjectEmail": "CK@noribbit.com" ,
        "SubjectAddress": {
          "Street": "Strawberry Fields",
          "City": "Liverpool",
          "State": "NY",
          "PostalCode": "11694",
          "Country": "US"
        }
      }
    }
  ]
}
```

## Request parameters

Attribute	Description
<div>CardNumber</div> <div>string</div>	<div>The credit card number found on the front or back of a credit card</div> <div>Required</div>
<div>FirstName</div> <div>string</div>	<div>The borrower cardholder first name on the credit card account</div> <div>Required</div>
<div>LastName</div> <div>string</div>	<div>The borrower cardholder last name on the credit card account</div> <div>Required</div>
<div>CVV</div> <div>string</div>	<div>The card verification value on the card</div> <div>Optional</div>
<div>ExpirationMonth</div> <div>string</div>	<div>Card expiration month 2-digits</div> <div>Required</div>
<div>ExpirationYear</div> <div>string</div>	<div>Card expiration year 2-digits</div> <div>Required</div>
<div>SubjectPhoneNumber</div> <div>string</div>	<div>Cardholder phone number 10 digits (for example 2015551234 no dashes)</div> <div>Optional</div>
<div>SubjectEmail</div> <div>string</div>	<div>Cardholder email address (for example me@mailprovider.com)</div> <div>Optional</div>
<div>SubjectAddress</div> <div>Address</div>	<div>Primary location details of cardholder</div> <div>Optional</div>

## PCI compliance

Payment Card Industry (PCI) standards ensure that sensitive credit card information is protected from fraudulent use. Arix processing complies with PCI standards.

Arix receives a request that includes the borrower's credit card number. This number is sent to a tokenization service that is Level 1 PCI-compliant. A token is sent back, which Arix stores.

When processing a card payment , Arix calls to detokenize the credit card numbers, and sends the credit card information to our Push to Card processor.

Arix logs contain only masked credit card numbers.

## 7.6.11. Wire parameters

You can use the Wire rail in Arix to push money into the account of a borrower, vendor or creditor.

Wire transfers go through **Fedwire**<sup>®</sup>, operated by the US Federal Reserve.

CRB releases wires to the Fed, multiple times a business day during business hours. Subsequent rails of a loan, will not be sent until the wire is completed or rejected.

JSON



```
{
  "Rails": [
    {
      "RailType": "Wire",
      "Priority": 1,
      "Amount": 80,
      "WireFields": {
        "SubjectName": "King Kong",
        "Description": "Smash",
        "AdditionalDescription": "Me want to buy present for Anne",
        "SubjectRouting": "021000021",
        "SubjectAccount": "20020020022",
        "SubjectAddress": {
          "Street": "350 Fifth Avenue",
          "AdditionalLine": "Empire",
          "BuildingNumber": "6",
          "City": "New York City",
          "State": "NY",
          "PostalCode": "10118",
          "Country": "US"
        }
      }
    }
  ]
}
```

# Request parameters

Attribute	Description
<div>SubjectName</div> <div>string</div>	Beneficiary name. Maximum 35 characters.
<div>SubjectRouting</div> <div>string</div>	Beneficiary bank's routing number Required
<div>SubjectAccount</div> <div>string</div>	Beneficiary account number Required
<div>Description</div> <div>string</div>	<i>For internal CR use.</i> Reason for the wire transfer. The data is not included with the outgoing FedWire message Required
<div>AdditionalDescription</div> <div>string</div>	Information from the originator to the beneficiary
<div>SubjectAddress</div> <div>Address</div>	The Address of the Beneficiary When sent like the example in the above JSON, the address will look like this: King Kong 350 Fifth Avenue 6 NY New York City 10118 Empire

## 7.6.12. Pull funding from merchant

---

The PullFunding API provides a method for MPLs to retrieve funds from merchant accounts back into the bank, using [Nacha Debits](#). This is especially useful when a merchant initiates a return or refund, as it reduces the need for manual reconciliation or additional requests, by reusing previously funded loan.

It is best practice to wait three days before sending out funds after a successful pull.

According to Nacha standards, a pull can be returned to the merchant. Most returns tend to occur within the first three days. To reduce the likelihood of overfunding, it is advisable to wait before sending out the next `FundingInfo` request.

### Flow of Funds

#### When this API can be called

Before calling the PullFunding API, ensure the loan is eligible for using this API according to `Loan Status` and `Amount` being requested

Loan status should be one of the following:

- `NotFullyFunded`
- `Funded`
- `ReadyToSell`
- `Sold`

`Amount` being requested in the API should be greater than 0 and less than the sum of rails that have been successfully sent and haven't been returned.

- $0 < \text{Amount} \leq \text{loan.amountFunded}$

### Flow of PullFunding

When this API succeeds, you receive an ID in the response. Track this Rail ID using the `RailUpdated` [webhook event](#).

The full list of FundResults of the rail is in the RailResultType [Enums](#) .

Using the PullFunding API, is essentially asking for a return of funds to the loan.

Unlike the success of a standard push rail, a successful pull will be in a **returned** status and *not* in a **success** status.

These statuses have different meanings than other rail statuses:

Action	FundResult ( <code>RailResultType</code> e <a href="#">Enums</a> )	Description
Put /PullFunding succeeds	Requested	Pull Funds has been requested
The Pull Succeeds	Returned	The pull funds request has <b>been successfully</b> received by NACHA, and the bank has received the funds. The pull amount has been returned to the loan.
Pull is returned	PullReturned	The ACH Pull, has been returned. The amount available in the loan for funding, will be deducted by the rail amount.

## PUT /{LoanID}/PullFunding

## Pull Funding

PUT

https://arixapisandbox.crbnj.net/Loan/{LoanID}/PullFunding



Request

Response

### PATH PARAMS

**LoanID**

String

required

The Unique ID of the loan returned from the POST /Loan API call

### BODY PARAMETERS

**FundingPull** ▶

Object

required



```
curl --location --request PUT 'https://arixapisandbox.crbnj.net/Loan/215eb
--header 'Authorization: Bearer <token>' \
--header 'Content-Type: application/json' \
--data '{
  "FundingPull": {
    "Rail": {
      {
        "ACHFields": {
          "StandardEntryClassCode": "PPD",
          "TransactionType": "pull",
          "ToRoutingNumber": "021214891",
          "ToAccountType": "GL",
          "ToAccountName": "Ari",
          "ToAccountNumber": "9999999999",
          "Description": "Pulling from Ari"
        },
        "RailType": "COSACH",
        "Priority": 1,
        "Amount": 100
      }
    }
  }
}'
```

## Responses

● 200

● 400



```
{
  "ResponseStatus": {
    "ErrorCode": "BadRequest",
    "Message": "PullFunding amount should be less than or equal to cur
    "Errors": []
  }
}
```

## 7.7. Cancel a loan

---

Use this call to cancel a loan by using its ID.

### PUT /Loan/{{loanId}}/Cancel

If you want to cancel a loan the amount funded has to be zero, **amountfunded** = 0. If any of the rails have funded, it can no longer be canceled via API until all rails have been returned.

Once you cancel a loan, it goes into a terminal state of canceled and it can no longer be changed.

#### IMPORTANT

Once a loan is canceled, it stays forever canceled and cannot be changed. Loans that were canceled, can continue to write status updates, but they won't change the loan.

Confirmation of the cancellation is received when you receive a [LoanStatusUpdated](#) webhook, with the status 100 (canceled).

#### Request URL

https:// ... /Loan/{{loanId}}/Cancel

A body should not be submitted with this API call.

### Loan statuses in which the Cancel API can be called successfully

Loan Status	Condition needed in order to call Cancel API
Received	
DocsComplete	
ComplianceFailed	
Approved	
AwaitingFunding	
InFunding	amountfunded = 0 AND The only rail requested is ACH/ACHSD. (This will cancel the ACH request)
NotFullyFunded	amountfunded = 0
Returned	
Funded	Should not call the API; contact MPL Ops
ReadyToSell	Should not call the API; contact MPL Ops
Sold (Not shown in Arix)	Should not call the API; contact MPL Ops

## 7.8. Purchasing a loan

---

### Lending Selling APIs v2

Cross River provides you (or your investor) three options to purchase your loans using the Selling system (part of Lending).

- **Recommended:** API and Webhook integration for full automation and loan level monitoring as the loan moves from "Pending" to "In Seasoning" to "Ready For Purchase" to "Purchased".
- **Low touch:** Log in to the UI, review and click to Purchase
- **No touch:** Auto Debit your account with Cross River

Regardless of the option, you can configure Hooks to receive a purchase confirmation via email or Slack.

HoldForSale ("HFS") and Long Term Hold For Sale ("LTHFS") are the loan types managed in Selling. You can also view loans that Cross River Retains ("RET").

#### UI

- The sandbox UI is <https://lendingappsandbox.crbcos.com/loans/selling>
- The production UI is <https://lendingapp.crbcos.com/loans/selling>

#### API

- Swagger: <https://lendingsandbox.crbcos.com/selling/swagger/index.html>
- URL for sandbox: <https://lendingappsandbox.crbcos.com/loans/selling>
- Base URL for production: <https://lendingapp.crbcos.com/loans/selling>
- API scope for sandbox is: **CosLending:Selling:stg**
- API scope for production is: **CosLending:Selling:prd**

## IMPORTANT

To access the links below you must have the following IP addresses allowlisted:

Sandbox - 66.206.202.39 , 66.206.202.12

Production -66.206.202.62 , 66.206.202.15

API	Description
<a href="#"><u>GET /selling/api/loans/{loanType}</u></a>	Get all loans by loan type
<a href="#"><u>POST /selling/api/loans/purchase</u></a>	Purchase loans
PUT/selling/api/loans/{id}/loanType	Request to change a loan type
<a href="#"><u>GET /selling/api/sofr</u></a>	Get the latest SOFR value

## GET/selling/api/loans/{loanType}

This API retrieves loan details based on **loan type**, which must be **HFS, LTHFS, or RET**

The loan type is specified in the URL path (e.g., /selling/api/loans/HFS). Users can refine their search with multiple **query parameters** including issuing bank, investor, loan status, search filters, pagination controls, sorting options, and date filters (funding date, purchase date, seasoning start date, etc.).

Additionally, results can be **exported as a CSV file** for easy data handling.

```
{
  "loans": [
    {
      "id": "735dfae0-c26a-4f75-9afa-b26c00468706",
      "number": "00b153ef-e8cb-4b76-aea7-c29879a69ad3",
      "originationStatus": "Funded",
      "saleStatus": "ReadyForPurchase",
      "mplId": "TST",
      "issuer": "CRB",
      "originationId": "",
      "seasoningStart": "2025-01-21",
      "amount": 7001.8,
      "adjustedAmount": 7000.5,
      "interestRate": 0.3,
      "interest": 4.02,
      "fees": 70.01,
      "total": 7075.83
    },
  ],
}
```

## POST/selling/api/loans/purchase

This API **executes the purchase of loans** based on loan type and purchase date. The request body includes key details such as the **date, loan type**, (1 for HFS, 2 for LTHFS), **issuing bank, MPL ID**, and **investor ID** (optional). The date parameter is required but does **not impact the purchase process**. If the **investor ID** is left empty, loans will be purchased **without investor allocation**.

For example: Purchase HFS loans for MPL TST for issuing bank CRB.

When the purchase is complete, the user can receive a webhook "batchpurchasecomplete".

## GET /selling/api/sofr

This API returns the SOFR 30 day average from the NY Federal Reserve. This rate is needed if your contract with Cross River is using SOFR as the basis for Interest Accrual during the seasoning period.



## 7.9. Hook notifications

---

### Overview

*Hooks* is a notification system, used in Lending to update you on the status of your loan. *Hooks* report to your system with real-time notifications when an event happens in Lending. You have to register to receive hooks for each relevant event type by configuring the destination and delivery method, most commonly webhooks. When that event occurs, a Hook is triggered and reports the updates to your system via the chosen delivery method.

For example, if a rail status changes to *returned*, a `RailUpdated` hook event would be triggered, and reported to the partner notifying them of the event. This notification will only be sent out, if you have previously registered to receive this type of event ( `RailUpdated` ), and the registration is active.

#### IMPORTANT

To access the links below you must have the following IP addresses allowlisted:

Sandbox - 66.206.202.39 , 66.206.202.12

Production -66.206.202.62 , 66.206.202.15

- Swagger: <https://lendingsandbox.crbcos.com/hooks/swagger/index.html>
- Base URL for sandbox: <https://lendingsandbox.crbcos.com/>
- Sandbox UI: <https://lendingappsandbox.crbcos.com/hooks/registrations>
- Production UI: <https://lendingapp.crbcos.com/hooks/registrations>
- API Scope:
  - CosLending:Hooks:stg for sandbox
  - CosLending:Hooks:prd for production



By default, a webhook that fails will retry 3 times with a delay interval of 5 minutes. Different Intervals can be configured for a registration by the Cross River support team.

Events can also be resent manually via the hooks UI, although **there is a retention policy of 60 days for events in the system.**

## Payload Delivery Wrapper

When an event is sent in the form of a webhook, there will always be a wrapper around the content. The actual event will be in the JSON's *content* section.

Example: LoanStatusUpdated Event inside Wrapper



```
{
  "MPLId": "TST",
  "Ignore": false,
  "content": {
    "Id": 0,
    "LoanId": "c1ee19e4-791c-4f38-b581-ae6d007692a6",
    "Status": 5,
    "TimeStamp": "2022-04-04T07:13:08.5754408+00:00",
    "DateInserted": "0001-01-01T00:00:00+00:00"
  }
}
```

Attribute	Description
MPLId string	A 3-letter ID for the partner
Ignore boolean	True or False
Content JSON	This is the actual content of the event being sent

## Sample Events

## LoanStatusUpdate

This event is reported every time the status of a loan is changed:

JSON



```
{
  "Id": 0,
  "LoanId": "3f07773b-5ba5-43e6-bf30-aea200361dad",
  "Status": 104,
  "TimeStamp": "2022-05-30T03:17:43.6233616+00:00",
  "DateInserted": "0001-01-01T00:00:00+00:00"
}
```

## ComplianceLoanFailed

This event is reported every time the loan fails to pass all compliance checks:

JSON



```
{
  "MPLId": "xxx",
  "LoanId": "...guid...",
  "CreateDate": "date",
  "FailedRulesReasons": [
    {
      "RuleName": "rule name",
      "Rule": "some string",
      "Data": "some string",
      "Result": true,
      "FailedComplianceID": 67
    }
  ]
}
```

## LoanNoteAdded

This event is reported when a bank operator wants to send you a message about a specific loan. This is a message from a human user:

```
{
  "Subject": "Your Loan 11112652056066 has additional comments",
  "Message": "This is a note sent from the manual review UI",
  "CreatedBy": "apollack, CRB",
  "CreatedOn": "2024-09-29T10:16:00.2828498",
  "LoanId": "e9ad35a8-a9fa-4334-995d-b1e800eb3e17",
  "LoanNumber": "11112652056066",
  "MplDetails": "TST - Testing MPL",
  "Uri": "https://lendingappsandbox.crbcos.com/loans/underwrite?query=eyJsb2F",
  "RuleName": null,
  "Data": null
}
```

Attribute	Description
Subject string	The title of the note being added
Message string	Message sent from a human user
CreatedBy string	The username that sent the message, along with the company they work for.
CreatedOn DateTime	When the note was created.
LoanId string	Unique Cross River identifier for the loan
LoanNumber string	Unique identifier for the loan, as sent by the partner
MplDetails string	Mplid and name of the Partner
Uri string	URI that links to the loan that was commented on, in the COS Lending Manual Review UI. This can only be opened in a whitelisted workstation.
RuleName string	<i>Internal Cross River information</i>
Data string	<i>Internal Cross River information</i>

## RailUpdated

This event is reported when there is a status update to a rail that was requested:

JSON



```
{
  "Id": 0,
  "RailId": "...ID of rail that is received when rail created...",
  "LoanId": "fd331267-6ac2-4818-9d0f-ab2f00747b61",
  "RailType": 8,
  "AmountFunded": 8500.0,
  "RailTransactionId": "...railid from cos...",
  "FundResult": int(this is enum RailResultType),
  "Message": "Description of rail created if no error returned",
  "ProcessedAt": "2019-12-26T07:05:09Z",
  "TransactionGroupId": "9dde776d-0c04-4bc8-9c9c-e0ad9a83c447"
}
```

## LoanSaleStatusUpdated

This event is reported when a loan sale status has changed:

JSON



```
{
  "MPLId": "TST",
  "Investor": "RotzehRibit",
  "LoanId": "048ef0f5-1c08-4f58-b3db-b26e00b59d59",
  "Status": 1, //This is the LoanSaleStatus enum
  "TimeStamp": "2025-05-04T18:15:29.242132+00:00",
  "LoanType": "HFS",
  "IssuingBank": "CRB",
  "PartnerLoanNumber": "18848835471438",
  "Principal": 5100,
  "Interest": 0,
  "VolumeFee": 0,
  "Fees": []
}
```

## BatchPurchaseCompleted

This message is sent when the purchase of a batch of loans is completed in selling:

```
{
  "MplId": "TST",
  "BatchId": "b5f018a8-331f-4762-8418-b2dd00ad65b8",
  "LoanType": null,
  "IssuingBank": "CRB",
  "Investor": "RotzehRibit",
  "TotalPrincipalCharged": 176500,
  "TotalInterestCharged": 26537.16,
  "TotalVolumeFeeCharged": 458.9,
  "TotalAchFeeCharged": 0,
  "NumOfLoansPurchased": 15,
  "NumOfLoansPartiallyPurchased": 0,
  "TimeStamp": "2025-05-14T10:36:07.4395183+00:00"
}
```

## POST /Hooks/v2/Registrations

Use this call to:

- Create a registration to hook events
- Edit a hook registration
- Restart a registration if it is suspended

Request URL [https:// .../Hooks/v2/Registrations](https://.../Hooks/v2/Registrations)

## Editing a Registration

To **edit** an existing registration, use the same API endpoint as for creating a registration.

### How It Works

- Include the **ID of the registration** in the POST request to modify an existing entry.
- The API **automatically determines** whether to create or update a registration based on:
  - **MPLID**

- **hookName**
- If a registration **already exists** for the given MPLID/hookName combination, the API will update it instead of creating a new one.
- If no matching registration is found, a **new** one will be created.

When editing a registration via the **UI**, any stored **passwords or tokens** will be shown **hashed on the UI**. For security reasons, You will need to **re-enter** these credentials when modifying the registration.

## Sample request in cURL - POST /Hooks/v2/Registrations

In the request below, `hookName` shows the event registered as `LoanStatusUpdated` :

 Curl



```
curl --location --request POST 'https://lendingsandbox.crbcos.com/Hooks/v2/Re
--header 'Content-Type: application/json' \
--header 'Authorization: Bearer <Token>' \
--data-raw '{
  "id": null,
  "applicationName": "Cos.Lending",
  "hookName": "LoanStatusUpdated",
  "hookCorrelationId": "Your 3 letter MPL ID",
  "options": {
    "hookType": "Web",
    "httpMethod": "POST",
    "uri": "Add your URL here, that is specific to this event type",
    "headers": {
      "headerName": [
        "LoanStatusUpdated"
      ]
    },
    "authenticationOptions": {
      "authenticationType": "None"
    }
  },
  "authenticationOptions": {
    "authenticationType": "None"
  },
  "suspended": false
}'
```

## Duplication Registrations

The default behavior for hooks registration, is to either create or update the registration based on the MPLID and the hookName, assuming there is only one registration per combination of Mplid/hookName.

In certain use cases, you might want to create two registrations for the same event. For example, one web hook, and one email hook. To do this, you need to supply a new Guid for the ID field of the registration.

## Webhook attributes



Attribute	Description
Id string	This field can be null or populated with the registration GUID for the registration. <b>null</b> - when creating a new registration <b>GUID</b> - when updating or unsuspending an existing registration
applicationName string	Should always be <b>Cos.Lending</b>
hookName string	The name of the hook you want to register for: - LoanStatusUpdated - ComplianceLoanFailed - RailUpdated
hookCorrelationId string	3-letter MPL ID
suspended boolean	True or false. True if the registration should be suspended and not send webhooks. False if the registration should be active. <b>If you edit a suspended registration, false will restart the registration.</b>
options object	Select the relevant values for the fields below.
options.hookType	Delivery method for the event (enum). Select from: - Web (for webhooks) - Email - Slack - Message (coming soon) - RabbitMq (coming soon) - Sqs (coming soon)
options.httpMethod string	POST
options.uri string	The URL to receive webhook events/

Attribute	Description
options.headers string array	Key/Value pairs of Headers and values. Each header can have an array of values: <pre>"Headers": {   "FirstHeader": [     "test1",     "test2"   ],   "secondHeader":["single value"] }</pre>
options.authenticationOptions object	Authentication options for web hooks. Select from: <ul style="list-style-type: none"> <li>- None</li> <li>- Basic</li> <li>- oidc</li> </ul>

## Sample response in JSON - POST /Hooks/v2/Registrations

Use the `id` received in the response to update or restart the registration.

JSON
<pre>{   "Version": "2.0",   "IsSuccessful": true,   "StatusCode": 200,   "Result": {     "id": "eb62ffbd-7ed4-4146-be48-7f2a46551c0e",     "applicationName": "cos.lending",     "hookName": "LoanStatusUpdated",     ...     ...     ..   } }</pre>

## Webhooks without Authentication

JSON



```
"options": {
  "hookType": "Web",
  "httpMethod": "POST",
  "uri": "Add your URL here, that is specific to this event type",
  "headers": {
    "headerName": [
      "Any Header you want to add to all events"
    ]
  },
  "authenticationOptions": {
    "authenticationType": "None"
  }
}
```

## Webhooks with Basic Authentication

JSON



```
"options": {
  "hookType": "Web",
  "httpMethod": "POST",
  "uri": "Add your URL here, that is specific to this event type",
  "headers": {
    "headerName": [
      "Any Header you want to add to all events"
    ]
  },
  "authenticationOptions": {
    "authenticationType": "Basic",
    "username": "yyy",
    "password": "1234"
  }
}
```

## Webhooks with OIDC Authentication

OpenID Connect using your own authentication authority.

### OIDC setup must be implemented first

Your OIDC must be fully implemented in order to use Hooks notifications with OIDC authentication.

This includes setting up a discovery endpoint `/.well-known/openid-configuration`.

JSON



```
"options": {
  "hookType": "Web",
  "httpMethod": "POST",
  "uri": "Add your URL here, that is specific to this event type",
  "headers": {
    "headerName": [
      "headerValue"
    ]
  },
  "authenticationOptions": {
    "authenticationType": "Oidc",
    "authorityUrl": "",
    "clientId": "",
    "clientSecret": "",
    "scope": ""
  }
}
```

## Slack hook registration

Hook events can also be sent to your slack channels. To do this you must:

1. Create a slack bot with permissions to write to a channel in your workspace.
2. Retrieve the auth token of the slackbot.
3. Add the slackbot to the slack channel that you wish to register.
4. Register the channel to receive events on the Cos.Lending.Hooks UI, or via the API payload below.

### IMPORTANT

Registering to a slack channel is not recommended if automation is needed. For complete automation, it is recommended to use "web" hooks instead.

JSON



```
{
  "id": null,
  "applicationName": "Cos.Lending",
  "hookName": "LoanNoteAdded",
  "hookCorrelationId": "Your 3 letter MPL ID",
  "options": {
    "hookType": "Slack",
    "authToken": "Your Slack API bot token.",
    "channel": "The channel name. You must also add your slack bot to this ch
  },
  "suspended": false
}
```

## Email hook registration

Hook events can also be sent to your Email address, or group email address.

1. Register the channel to receive events on the Cos.Lending.Hooks UI, or via the API payload below.

### IMPORTANT

Registering to receive events via email is not recommended if automation is needed. For complete automation, it is recommended to use "web" hooks instead.

```
{
  "id": null,
  "applicationName": "Cos.Lending",
  "hookName": "LoanNoteAdded",
  "hookCorrelationId": "Your 3 letter MPL ID",
  "options": {
    "hookType": "Email",
    "to": [
      "email address"
    ],
    "cc": [
      null
    ],
    "bcc": [
      null
    ]
  },
  "suspended": false
}
```

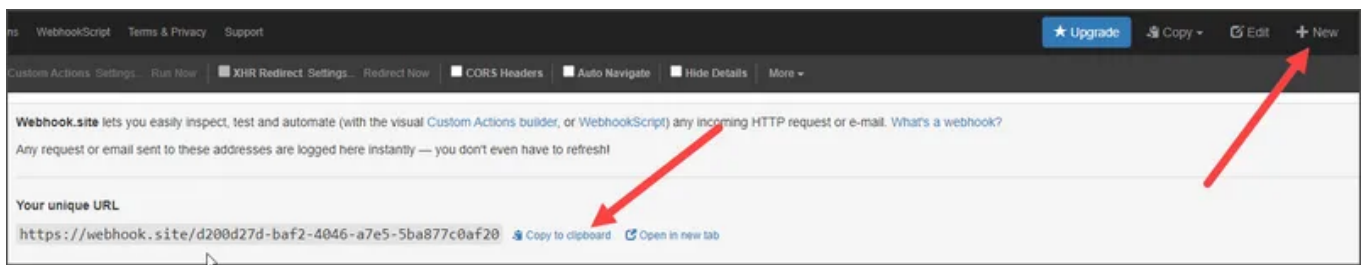
## Testing webhook registration

To test registering webhooks, configure a URL for the specific event you are registering to. If a URL is not yet available, then you can use the [webhook.site](https://webhook.site) as a temporary test solution. Follow these instructions to test the webhooks.

### To register your URL to receive webhook events:

1. Go to [https://webhook.site/](https://webhook.site)
2. Click **New** on the top menu. The site returns **Your unique URL**.
3. **Copy to clipboard** your **unique URL**.

If you already have other webhooks registered, the browser might automatically direct you to the URL you've used.



4. Paste the copied URL into the `options.uri` field of the JSON body.

5. Send the Registration API call with hook type "web" to register for webhooks.

The response field `Result.id` is the `RegistrationId` of the event. You can find the registration ID with the call `GET Hooks/v2/Registrations`.

Repeat the process for each event that you want to register for.

## DELETE /Hooks/v2/Registrations

Request URL `https://.../Hooks/v2/Registrations`

Use this call to unregister any hook event you are registered to.

## Sample request in cURL - DEL /Hooks/v2/Registrations

Curl



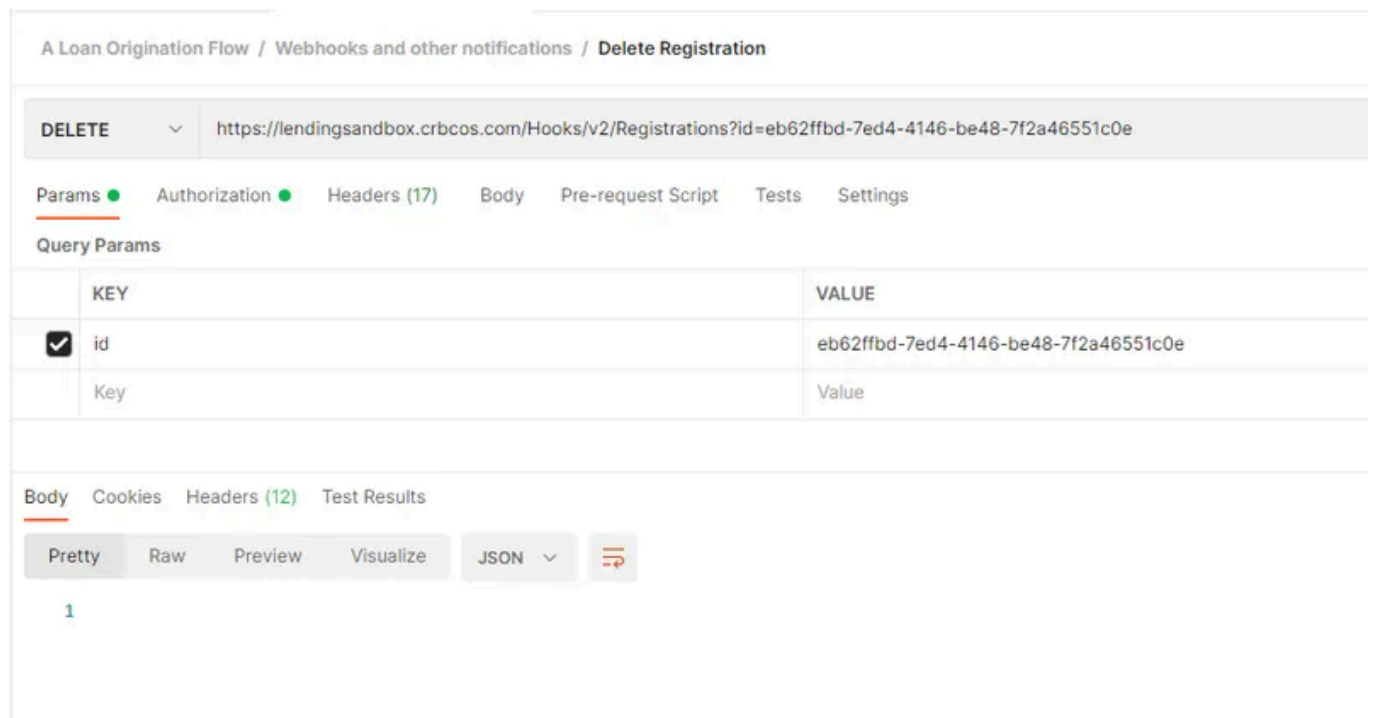
```
curl --location --request DELETE 'https://lendingsandbox.crbcos.com/Hooks/v2/
--header 'Accept: application/json' \
--header 'Authorization: Bearer ew0KICAiYWxnIjogIl*****' \
--header 'Cookie: ss-opt=temp; ss-id=QVhFixRMpD92y7HQ7b74; ss-pid=De9QBzZCTdh'
```

Required query parameter

Parameter name	Parameter type	Data type	Description
id	query	long	The registration ID of hook registration to remove

## Sample response in JSON - DEL /Hooks/v2/Registrations

There is no response in the image below because the hook registration was deleted successfully.



You can use the Arix test collection in Postman to practice unregistering from an event:

1. In the Postman collection, **Delete Registration** click the tab, **Params**.
2. In the **KEY** field, enter id .
3. In the **VALUE** field, enter the registration ID number that you received when you registered to the event.
4. Click **Send**.



## 7.10. Enums

---

### **LoanStatusEnum**

These enums communicate the status of a loan.

Enums	Enum number	Description
Received	1	Arix received a new loan application. Send attachments as the next step in the process.
DocsComplete	2	All required documents were received
PassedCompliance	3	Loan successfully passed compliance
Approved	4	Loan was approved for funding
InFunding	5	At least 1 rail was submitted for funding
Funded	6	Full loan amount was funded
AccountingEntriesComplete	7	An internal CR process. Confirms that account entries are complete.
ReadyToSell	8	Loan finished the seasoning period and is ready to be sold
Sold	9	Loan was sold (not in use) Sale info is on the post-purchase report in SFTP
Settled	10	Loan was paid back by end user (not in use)
AwaitingFunding	11	All compliance rules were passed and the system is waiting for the funding information. Once in this status, only a PUT /FundingInfo will get the loan funded.
InReverse	80	A return/ cancel has been requested, and the funding is being returned to the originating account.
AccountingEntriesReversed	90	<i>Internal status</i> Loan was canceled or fully returned and accounting entries were reversed.
Cancelled	100	Issuing bank canceled the loan. <b>Terminal State</b>
ComplianceFailed	101	Loan failed the compliance check

Enums	Enum number	Description
Rejected	102	Issuing bank didn't approve - possible only prior to funding. <b>Terminal State.</b>
Returned	103	All rails that were successfully funded came back as returned
NotFullyFunded	104	Funding was attempted, but a balance remains to be funded

## FundingResult enums

These enums communicate the results of a funding attempt.

Enums	Enum number	Description
Success	0	All funded
PartialFailure	1	Some funds were sent successfully, but some failed
FundingFailed	2	Funding failed and nothing was funded

## RailResultType enums

These enums communicate the results of a rail transaction as part of the funding attempt.

Enums	Enum number	Description
NotAttemptd	0	Rail wasn't attempted
Requested	1	API call was successful and the funds are pending transmission
Success	2	<p>Full requested amount has been funded and released on the rail.</p> <p><b>Note:</b> There is lag time between the time an ACH is sent and when it is received (by the receiving bank), usually 1 business day (unless it was sent as a same-day ACH).</p> <p>CR considers the rail a <b>success</b> if and until we receive a notice from the receiving bank that the rail failed. At that time, the status of the rail changes from <b>Success</b> to <b>Returned</b>.</p>
RequestFailure	3	The request to fund failed immediately upon request (usually due to a mistake in the API instructions.)
Rejected	4	The rail was rejected
Returned	5	The rail that was sent and marked as <b>success</b> was returned.
Skipped	6	A rail was skipped. No need to reattempt the funding.
ValidationFail	7	The rail failed because it couldn't be validated
CosAchPaymentRejected	8	<p>Arix received a notification that a specific ACH rail was rejected</p> <p><i>(Internal Ops)</i></p>
CosAchPaymentCanceled	9	<p>CR Ops canceled the ACH payment</p> <p><i>(Internal Ops)</i></p>
CosAchNocReceived	10	<p>A Notice of Change (NOC) was received. Relevant for recurrent disbursements.</p> <p><i>(Internal Ops)</i></p> <p>Contact customer for updated instructions.</p>

Enums	Enum number	Description
CosAchHoldEscalated	11	CR Ops put the ACH transaction on hold (Internal Ops)
CosCoreTransactionCompleted	12	COS Core (part of XPay) transaction was completed successfully (Internal Ops)
CosCoreTransactionRejected	13	COS Core (part of XPay) transaction was unable to complete (Internal Ops)
CosXPaymentReceived	15	Inbound payment request has been received and is awaiting approval (Internal Ops)
CosXPaymentCanceled	16	Payment canceled at the sender's request or has expired (Internal Ops)
CosXPaymentRejected	17	Payment was rejected by the receiver (Internal Ops)
CosXPaymentFailed	18	Payment failed to post to either the sender or receiver's account. Most commonly due to insufficient funds (Internal Ops)
CosXPaymentCompleted	19	Payment has been posted to both the sender and receiver's accounts (Internal Ops)
CosCoreTransactionCompletedReverse	20	COS Core (part of XPay) reversal transaction has been completed (Internal Ops)
CosXPaymentRejectedReverse	24	COS XPay requested reversal hasn't been approved (Internal Ops)

Enums	Enum number	Description
<code>CosXPaymentFailedReverse</code>	25	COS XPay requested reversal was unable to be reversed (Internal Ops)
<code>CosXPaymentCanceledReverse</code>	26	COS XPay reversal was canceled ( <i>Internal Ops</i> )
<code>CheckCashed</code>	27	Check was cashed
<code>CheckPrinted</code>	28	Check was printed
<code>DigitalCheckPrinted</code>	29	A digital check was printed
<code>CheckMailed</code>	30	The check was mailed
<code>Failed</code>	31	The check wasn't able to be cashed
<code>Cancelled</code>	32	Rail was cancelled

## RailType enums

These enums communicate the type of rail used to make the transactions.

Enums	Enum number	Description
RPPS	4	RPPS rail
NF	7	No funding
COSACH	8	Next day funding via ACH
COSACHSD	9	Same day funding via ACH
COSXPay	10	Internal transfer between CR accounts
AdjustmentAction	20	Adjustments to a loan after it was funded
Check	30	Physical check
RTP	40	Real Time Payments
Card	50	Card
Wire	60	Wire

## RPPS enums

RPPS uses enums to return funding attempt results.

Enum name	Enum number	Description
Received	1	The funding attempt was received by RPPS
Submitted	2	The funding attempt was submitted by RPPS
Pending	3	The funding attempt is pending
Approved	4	The funding attempt was approved by RPPS
Rejected	5	The funding attempt was rejected by RPPS
Failed	6	The funding attempt failed in RPPS
Returned	7	The rail was returned by RPPS

## 7.11. Servicing reports

---

CR requires that an MPL, or the Servicer used by the MPL, provide daily all servicing events for the loans issued by CR.

The servicing report contains a record for every loan every day. It is a snapshot of each loan. The headers are the same as those listed in the data dictionary.

The table below is the reference information for servicing.

### File conventions

- File format: CSV.
- File transport mechanism: SFTP (SSH; supported public key algorithms RSA, sha2 521, sha2 384, sha 256)
- File Name: MplId\_ServicingTape\_yyyymmdd.csv, where yyyymmdd is the date of file generation
  - MplId: This is internal to CRB. Please check with the data team [dataengineers@crossriver.com](mailto:dataengineers@crossriver.com) for the value to be used in the filename
- File location: \Prod\MPL\LoanServicingTapes\MplId\_ServicingTape\_yyyymmdd.csv
- Report date: should match the date of the filename
- Report data: should be from the day prior to the report date

#### Example

The following shows the relationship between these three data points:

- A file with name : MplId\_ServicingTape\_20231225.csv. File is generated on the 12/25 for the previous day's data.
- Report Date: Should be 2023-12-25
- Report data: Loan positions/standing as of end of 2023-12-24



# Dictionary updates

Attributes	Action	Date
IssuingBank	Added	October 21, 2021
ForbearanceDuration ForbearanceDurationType ExtraFields	Added	June 16, 2021
ForbearanceDurationMonths	Removed	June 16, 2021

# Identification

Field	Description
Platform alphanumeric	<b>Required</b> MPL name
MplAcctID alphanumeric	<b>Required</b> Unique loan number
IssuingBank alphanumeric	<b>Required</b> Name of issuing bank of the loan. For CR originated loans, it should say "CRB".
Servicer alphanumeric	<b>Required</b> Servicer name
ServicerLoanNum alphanumeric	<b>Required</b> Servicer loan number

# Sale/Transfers

Field	Description
<div>BeginningSubpoolId</div> alphanumeric	<b>Required</b> On transfer day, this field will show the SubpoolId prior to the transfer. For subpool transfers or sales.
<div>EndingSubpoolId</div> alphanumeric	<b>Required</b> On transfer day, this field will show the SubpoolId after the transfer. For subpool transfers or sales.
<div>PriorOwner</div> alphanumeric	<b>Required</b> On transfer day/sale day, this field shows the owner prior to the transfer. <ul style="list-style-type: none"> <li>• CRB-HFS: Loans bought back by CR after the 3 day seasoning period</li> <li>• CRB-LTHFS: Loans owned by CR with a longer seasoning period</li> <li>• CRB-Retained: Loans retained by CR</li> </ul>
<div>CurrentOwner</div> alphanumeric	<b>Required</b> On transfer day, this field shows the owner after the transfer <ul style="list-style-type: none"> <li>• CRB-HFS : Loans bought back by CR after the 3 day seasoning period</li> <li>• CRB-LTHFS: Loans owned by CR with a longer seasoning period</li> <li>• CRB-Retained: Loans retained by CR</li> </ul>
<div>CurrentTransferredInterest</div> number up to 2 decimals	<b>Required for subpool transfers and sales</b> Interest transferred in the sale or in the subpool transfer
<div>CurrentTransferredPrincipal</div> number up to 2 decimals	<b>Required for subpool transfers and sales</b> Principal transferred in the sale or in the subpool transfer
<div>CurrentTransferDate</div> date	<b>Required for subpool transfers and sales</b> Date loan is sold/transferred to the new owner or subpool.

Field	Description
	<p>This field should only be populated on the day after transfer.</p> <p>Format: yyyy-mm-dd</p>

# Loan terms

Field	Description
<div>ContractDate</div> <div>date</div>	<b>Required</b> Note date. Format: yyyy-mm-dd
<div>FirstFundedDate</div> <div>date</div>	<b>Required</b> Funding date for loans that are disbursed on multiple days: FirstFundedDate is the date of the first disbursement MostRecentFundedDate is the date of the most recent disbursement. For loans funded in a single disbursement, the  FirstFundedDate is the same as the MostRecentFundedDate Format: yyyy-mm-dd
<div>MostRecentFundedDate</div> <div>date</div>	<b>Required</b> Date of most recent disbursement. If the loan is funded in a single disbursement, this is the same as the FirstFundedDate. Format: yyyy-mm-dd
<div>Boarding Date</div> <div>date</div>	<b>Required</b> Date of most recent disbursement. If the loan is funded in a single disbursement, this is the same as the FirstFundedDate. Format: yyyy-mm-dd
<div>OriginalMaturityDate</div> <div>date</div>	<b>Required</b> Date of the final loan maturity per the loan agreement. Format: yyyy-mm-dd
<div>CurrentMaturityDate</div> <div>date</div>	<b>Required</b> Current maturity date. Format: yyyy-mm-dd
<div>Vintage</div> <div>date-number</div>	<b>Required</b>

Field	Description
	Year and month of note. Format: yyyy-mm-dd
<code>OriginalLoanAmount</code> number up to 2 decimals	<b>Required</b> Gross loan amount
<code>OriginalFundedAmount</code> number up to 2 decimals	<b>Required</b> Net funding to borrower
<code>CurrentDisbursementAmount</code> number up to 2 decimals	<b>Required</b> Aggregated total of all disbursed amounts so far for this loan. For a single disbursement, this equals the loan amount.
<code>CurrentDisbursementCount</code> integer	<b>Required</b> The (number) count of disbursements funded for this loan. For a single disbursement, the number is 1.
<code>OriginalScheduledPeriodicPmt</code> number up to 2 decimals	<b>Required</b> Borrower's required periodic payment per the original amortization schedule.
<code>CurrentScheduledPeriodicPmt</code> number up to 2 decimals	<b>Required</b> Borrower's required periodic payment per the current amortization schedule.
<code>OriginalScheduledFinalPayment</code> number up to 2 decimals	<b>Required</b> Borrower's required final payment per the original amortization schedule.
<code>CurrentScheduledFinalPayment</code> number up to 2 decimals	<b>Required</b> Borrower's required final payment per the current amortization schedule.
<code>PaymentFrequency</code> alphanumeric	<b>Required</b> Frequency of regularly scheduled payments per the amortization schedule - monthly, bimonthly, weekly, biweekly.

Field	Description
OriginalInterestRate number up to 2 decimals, formatted as a percent	<b>Required</b> Original note interest rate. Format: Percent (%)
CurrentInterestRate number up to 2 decimals, formatted as a percent	<b>Required</b> Current interest rate
InterestAccrualMethod alphanumeric	<b>Required</b> Method of interest accrual
OriginalBorrowerState alpha - 2 letters	<b>Required</b> State in which the loan was originated
CurrentBorrowerState alpha - 2 letters	<b>Required</b> State where the borrower currently resides
OriginalFicoScore integer	<b>Required</b> FICO score at origination.
OriginalFicoScoreDate date	<b>Required</b> Date the FICO score was pulled at origination. Format: yyyy-mm-dd
OriginalVantageScore integer	<b>Required</b> VantageScore at origination
OriginalVantageScoreDate date	<b>Required</b> Date the VantageScore was pulled at origination. Original internally assigned credit rating (loan grade). Format: yyyy-mm-dd
UpdatedFicoScore integer	Required if an updated credit pull was made Last FICO score pulled - should only be populated if a new FICO was pulled.
UpdatedFicoScoreDate date	Required if an updated credit pull was made Date of last FICO score pulled - should only be populated if a new FICO was pulled. Format: yyyy-mm-dd

Field	Description
<div>UpdatedVantageScore</div> integer	<p>Required if an updated credit pull was made</p> <p>Last VantageScore pulled - should only be populated if a new VantageScore was pulled.</p>
<div>UpdatedVantageScoreDate</div> date	<p>Required if an updated credit pull was made</p> <p>Date of last VantageScore pull - should only be populated if a new VantageScore was pulled.</p> <p>Format: yyyy-mm-dd</p>
<div>PreDTIExMortgage</div> number up to 2 decimals, formatted as a percent	<p><b>Required</b></p> <p>DTI excluding mortgage pre-loan.</p> <p>Format: Percent (%)</p>
<div>PreDTIInMortgage</div> number up to 2 decimals, formatted as a percent	<p><b>Required</b></p> <p>DTI including mortgage pre-loan.</p> <p>Format: Percent (%)</p>
<div>PreDTIDate</div> date	<p><b>Required</b></p> <p>DTI date pulled pre-loan.</p> <p>Format: yyyy-mm-dd</p>
<div>PostDTIDate</div> date	<p><b>Required</b></p> <p>DTI date pulled post-loan.</p> <p>Format: yyyy-mm-dd</p>
<div>PostDTIExMortgage</div> number up to 2 decimals, formatted as a percent	<p><b>Required</b></p> <p>DTI excluding mortgage post-loan.</p> <p>Format: Percent (%)</p>
<div>PostDTIInMortgage</div> number up to 2 decimals, formatted as a percent	<p><b>Required</b></p> <p>DTI including mortgage post-loan.</p> <p>Format: Percent (%)</p>
<div>OriginalCreditGradeInternal</div> number or alphanumeric	<p><b>Required</b></p> <p>Original internally assigned credit rating (loan grade)</p>

Field	Description
CurrentCreditGradeInternal number or alphanumeric	<b>Required</b> Current internally assigned credit rating (loan grade)
OriginationFeePct number up to 2 decimals	<b>Required</b> Origination fee by percent
OriginationFeeAmt number up to 2 decimals	<b>Required</b> Origination fee by dollar amount
APR number up to 2 decimals, formatted as a percent	<b>Required</b> Annual percentage rate
OriginalTerm number up to 2 decimals	<b>Required</b> Number of months to pay loan before maturity at origination
CurrentTerm number up to 2 decimals	<b>Required</b> Number of months remaining to maturity
OriginalNumScheduledPayments <b>integer</b>	<b>Required</b> Number of periodic payments to maturity at origination
CurrentNumPaymentsScheduled <b>integer</b>	<b>Required</b> Number of remaining periodic payments to maturity

## Activity



Field	Description
<div>ReportDate</div> <div>date</div>	<p><b>Required</b></p> <p>This should be the current date, and all balances and running monthly totals should be as of the prior day.</p> <p>Format: yyyy-mm-dd</p> <p>Example</p> <p>Reporting date should be 5/1/2016 for the April loan summary.</p>
<div>BeginningPrincipalBalance</div> <div>number up to 2 decimals</div>	<p><b>Required</b></p> <p>Beginning principal loan balance for prior day. Should equal previous day ending principal balance.</p>
<div>BeginningTotalBalance</div> <div>number up to 2 decimals</div>	<p><b>Required</b></p> <p>Beginning principal + interest + fees loan balance for prior day</p>
<div>BeginningBalancePostChargeOff</div> <div>number up to 2 decimals</div>	<p><b>Required</b></p> <p>Beginning principal balance for prior day after charge off. This balance should exclude the charged off amount (if any).</p>
<div>EndingPrincipalBalance</div> <div>number up to 2 decimals</div>	<p><b>Required</b></p> <p>Ending principal loan balance for prior day</p>
<div>EndingTotalBalance</div> <div>number up to 2 decimals</div>	<p><b>Required</b></p> <p>Ending principal + interest + fees loan balance for prior day</p>
<div>EndingBalancePostChargeOff</div> <div>number up to 2 decimals</div>	<p><b>Required</b></p> <p>Ending principal balance for prior day after charge off. This balance should exclude the charged off amount (if any).</p>

Field	Description
<b>PeriodPrincipalPmt</b> number up to 2 decimals	<b>Required</b> Principal payments received during the current month - running monthly total.
<b>PeriodPrincipalAdj</b> number up to 2 decimals	<b>Required</b> Small balance adjustments made to the loan balance during the current month - running monthly total.
<b>PeriodPrincipalPmtScheduled</b> number up to 2 decimals	<b>Required</b> Scheduled principal payments received during the current month - running monthly total.
<b>PeriodPrincipalPmtPrepaid</b> number up to 2 decimals	<b>Required</b> Prepay (non-scheduled) principal received during the current month - running monthly total.
<b>PeriodInterestPmt</b> number up to 2 decimals	<b>Required</b> Interest payments received during the current month - running monthly total.
<b>PeriodInterestPmtScheduled</b> number up to 2 decimals	<b>Required</b> Scheduled interest payments received during the current month - running monthly total.
<b>PeriodLateFeeAssessed</b> number up to 2 decimals	<b>Required</b> Amount of late fees charged - running monthly total.
<b>PeriodOtherFeesAssessed</b> number up to 2 decimals	<b>Required</b> Amount of other fees assessed (excluding NSF or late fees) - running monthly total.
<b>PeriodOtherFeesPaid</b>	<b>Required</b>

Field	Description
number up to 2 decimals	Amount of other fees (excluding NSF or late fees) paid - running monthly total.
<b>PeriodNsfAssessed</b> number up to 2 decimals	<b>Required</b> Amount of insufficient funds charged - running monthly total.
<b>PeriodNsfPaid</b> number up to 2 decimals	<b>Required</b> Amount paid for insufficient funds - running monthly total.
<b>LastPaymentReceivedDate</b> date	<b>Required</b> Day of month the last borrower payment was received. Format: yyyy-mm-dd
<b>NextPaymentDueDate</b> date	<b>Required</b> Next payment due date. Format: yyyy-mm-dd
<b>NextPrincipalDueDate</b> date	<b>Required</b> Next principal payment due date. Format: yyyy-mm-dd
<b>NextInterestDueDate</b> date	<b>Required</b> Next interest due date. Format: yyyy-mm-dd
<b>ChargeOffDate</b> date	<b>Required</b> Date the loan is charged off book. Format: yyyy-mm-dd
<b>ChargedOffPrincipalAmount</b> number up to 2 decimals	*Required if a charge off occurred Total principal charged off during current month. This should only be populated during the month that the charge off occurs.
<b>ChargedOffInterestAmount</b>	*Required if a charge off occurred Total interest charged off/ reversed during the current month.

Field	Description
number up to 2 decimals	This should only be populated during the month that the charge off occurs.
<b>InterestAccruedTotal</b> 1 number up to 2 decimals	<b>Required</b> Interest accrued but not collected as of prior day
<b>CumulInterestPmtLTD</b> number up to 2 decimals	<b>Required</b> Total interest payment received from the borrower since inception
<b>CumulPrincipalPmtLTD</b> D number up to 2 decimals	<b>Required</b> Total principal payment received from the borrower since inception
<b>CumulPrincipalPmtPrepaidLTD</b> number up to 2 decimals	<b>Required</b> Total prepay principal received from the borrower since inception
<b>PrincipalRecoveredAmount</b> t number up to 2 decimals	<b>Required</b> Principal amount recovered after charge off - life to date
<b>InterestRecoveredAmount</b> t number up to 2 decimals	<b>Required</b> Interest amount recovered after charge off - life to date
<b>LateFeeRecovered</b> number up to 2 decimals	<b>Required</b> Late fee amount recovered after charge off - life to date.
<b>NsfFeeRecovered</b> number up to 2 decimals	<b>Required</b> NSF fee amount recovered after charge off - life to date

Field	Description
<div>OtherFeesRecovered</div> number up to 2 decimals	<b>Required</b> Other fees (excluding late fees or NSF fees) amount recovered after charge off - life to date.
<div>AverageDailyBalance</div> number up to 2 decimals	<b>Required</b> Average loan outstanding for the month

## Loan status

Field	Description
<div>LoanStatus</div> alphanumeric	<b>Required</b> Payment status of the loan: Current- 0 days past due InGracePeriod - 1 thru grace period days past due Delinquent - grace period + 1 thru charge off days past due ChargedOff - loan charged off Forebearance - loan is in forbearance/deferment status Bankruptcy - loan in bankruptcy Matured - loan paid off on normal amortization schedule PaidOff - loan paid off early
<div>DaysPastDue</div> integer	<b>Required</b> Number of days past due - total
<div>DelinquencyStartDate</div> <b>date</b>	<b>Required</b> Date the loan entered the delinquent status. Format: yyyy-mm-dd
<div>PastDuePrincipalAmt</div> number up to 2 decimals	<b>Required</b> Amount of principal payment past due
<div>PastDueInterestAmount</div> number up to 2 decimals	<b>Required</b> Amount of interest payment past due
<div>PastDueTotalAmount</div> number up to 2 decimals	<b>Required</b> Amount of principal + interest + fees past due
<div>Times1to5DPD</div> integer	<b>Required</b> Number of times loan was $0 < \text{DPD} \leq 5$ .
<div>Times6to15DPD</div> integer	<b>Required</b> Number of times loan was $5 < \text{DPD} \leq 15$ .
<div>Times16to30DPD</div> integer	<b>Required</b> Number of times loan was $15 < \text{DPD} \leq 30$ .
<div>Times31to60DPD</div> integer	<b>Required</b> Number of times loan was $30 < \text{DPD} \leq 60$ .

Field	Description
<div>Times61to90DPD</div> integer	<b>Required</b> Number of times loan was $60 < \text{DPD} \leq 90$ .
<div>Times91to120DPD</div> integer	<b>Required</b> Number of times loan was $90 < \text{DPD} \leq 120$ .
<div>Times121plusDPD</div> integer	<b>Required</b> Number of times loan was $120 < \text{DPD}$ .
<div>ConfirmOfFraudDate</div> date	<b>Required</b> Date the loan was confirmed as fraudulent. Format: yyyy-mm-dd
<div>FraudAmt</div> number up to 2 decimals	<b>Required</b> Total amount lost to fraud
<div>ConfirmOfDeathDate</div> date	<b>Required</b> Date the borrower's death was confirmed. Format: yyyy-mm-dd

## Bankruptcy

Field	Description
<div>BankruptcyStatus</div> <div>alphanumeric</div>	<b>Required</b> Current status of bankruptcy as of prior day
<div>BankruptcyChapter</div> <div>alphanumeric</div>	<b>Required</b> Bankruptcy chapter
<div>BankruptcyNoticeDate</div> <div>date</div>	<b>Required</b> Date of the bankruptcy notice. Format: yyyy-mm-dd
<div>BankruptcyFilingDate</div> <div>date</div>	<b>Required</b> Date the bankruptcy was filed. Format: yyyy-mm-dd
<div>BankruptcyCloseDate</div> <div>date</div>	<b>Required</b> Date the bankruptcy was closed. Format: yyyy-mm-dd
<div>BankruptcyCloseReason</div> <div>alphanumeric</div>	<b>Required</b> Reason the bankruptcy was closed.
<div>BankruptcyProcessDate</div> <div>date</div>	<b>Required</b> Date the bankruptcy was processed. Format: yyyy-mm-dd
<div>BankruptcyDischargeDate</div> <div>date</div>	<b>Required</b> Date the bankruptcy was discharged. Format: yyyy-mm-dd
<div>BankruptcyDismissalDate</div> <div>date</div>	<b>Required</b> Date the bankruptcy was dismissed. Format: yyyy-mm-dd
<div>BankruptcyWithdrawalDate</div> <div>date</div>	<b>Required</b> Date the bankruptcy was withdrawn. Format: yyyy-mm-dd



Field	Description
<b>ForbearanceStartDate</b> date date	<b>Required</b> Date the loan entered the forbearance status. Format: yyyy-mm-dd
<b>ForbearanceEndDate</b> e date	<b>Required</b> Date the loan exited the forbearance status. Format: yyyy-mm-dd
<b>ForbearanceDuration</b> on integer	<b>Required</b> Duration of the forbearance period. Duration type has to be indicated in the ForebearanceDurationType field.
<b>ForebearanceDurationType</b> ionType integer	<b>Required</b> Indicator that identifies the duration type reported in the ForbearanceDuration field 1 - Days 2 - Months 3 - Years
<b>ForbearanceReason</b> n alphanumeric	<b>Required</b> Reason for the forbearance

## Modifications

Field	Description
<div>ScraFlag</div> alphanumeric	<b>Required</b> Flag indicating that SCRA applies to this loan. Y or N only.
<div>ScraRate</div> number up to 2 decimals	<b>Required</b> Modified rate during the SCRA period - Y or N.
<div>ScraBeginDate</div> date	<b>Required</b> Date the SCRA period begins. Format: yyyy-mm-dd
<div>ScraEndDate</div> date	<b>Required</b> Date the SCRA period ends. Format: yyyy-mm-dd
<div>ModFlag</div> alphanumeric	<b>Required</b> Flag indicating the loan was modified. Only the most recent modification should be provided. Y or N only.
<div>ModStatus</div> alphanumeric	<b>Required</b> Flag indicating whether the modification is active. Only the most recent modification should be provided. Y or N only.
<div>ModType</div> number up to 2 decimals	<b>Required</b> Code indicating the type of loan modification. Only the most recent modification should be provided. 1 - Rate 2 - Payment deferral 3 - Principal reduction 4 - Term adjustment 5 - Payment adjustment 6 - Other

Field	Description
<div>ModDescr</div> alphanumeric	<b>Required</b> Description of the specific modification made. Only the most recent modification should be provided.
<div>ModRate</div> number up to 2 decimals	<b>Required</b> The new rate, post modification. Only most recent modification should be provided.
<div>ModPmt</div> number up to 2 decimals	Required The new payment, post modification. Only the most recent modification should be provided.
<div>ModTerm</div> integer	<b>Required</b> The new term, post modification. Only the most recent modification should be provided.
<div>ModBa1</div> number up to 2 decimals	<b>Required</b> The new principal balance, post modification. Only the most recent modification should be provided.
<div>ModFinalPmt</div> number up to 2 decimals	<b>Required</b> The new final payment, if applicable, post modification. Only the most recent modification should be provided.
<div>ModMaturityDate</div> date	<b>Required</b> The new maturity, post modification. Only the most recent modification should be provided. Format: yyyy-mm-dd

## Settlement

Field	Description
SettlementDate date	<b>Required</b> Date the loan was settled with the borrower. Format: yyyy-mm-dd
SettlementAmount numeric	<b>Required</b> The portion of the balance agreed upon that WILL BE PAID by the borrower.
SettlementAmountPaid numeric	<b>Required</b> The portion of the balance agreed upon that HAS ALREADY BEEN PAID by the borrower.
ForgivenAmount numeric	<b>Required</b> The portion of the balance that won't be collected due to the settlement agreement (as of the agreement date).
UnpaidSettlementBalance numeric	<b>Required</b> The outstanding amount remaining of the agreed upon settlement amount.
ExtraFields alphanumeric	<i>Optional</i> Json blob allowing for all extra fields. The blob must EXCLUDE text delimiters and line feeds. Any changes/additions to this field must go through change management.

## 7.12. Transaction reports

---

Transaction data enables Cross River to collect interest based on an adjusted loan balance. The table below is the transaction reference information for the files.

### File conventions

- File format: CSV.
- File transport mechanism: SFTP (SSH; supported public key algorithms RSA, sha2 521, sha2 384, sha 256).
- File location: \Prod\MPL\TransactionReports\MplId\_TransactionTape\_yyyymmdd.csv
- Naming convention: MplId\_TransactionTape\_yyyymmdd.csv, where yyyymmdd is the date of file generation.
- Report date: should match the date of the filename.
- Report data: should be from the day prior to the report date.

- Make sure to provide a list of possible `TransactionCodes` and `SourceCodes`
- This file should include all borrower payments, payment reversals, cancellations, refunds, charge offs, loan modifications, and fee assessments.

Transaction code mapping table is below.

### Dictionary updates

Attribute	Action	Date
<div>Amount</div> <div>BalanceImpactCode</div>	Added	October 21, 2021
<div>TransactionAmount</div> <div>BalancePrior</div> <div>PrincipalAmount</div> <div>InterestAmount</div> <div>LateFeeAmount</div> <div>BalanceAfter</div> <div>NextDueDate</div> <div>Notes</div>	Removed	October 21, 2021

Attribute	Description	Sample Record
TransactionId alphanumeric	<p>Unique loan transaction ID. Use the ID from the servicing system.</p> <p>Note that the TransactionId and TransactionCode must be unique.</p>	123ABC456
LoanId alphanumeric	Unique loan number	777777-AAAAA
TransactionDate date	Date the transaction was processed by the servicing system	Format: yyyy-mm-dd 2014-11-10
SourceCode alphanumeric	An identifier used to describe the method of the transaction	AAA
SourceCodeDesc alphanumeric	A description that correlates to the SourceCode identifier such as personal check, money order, AutoPay, etc.	ACH
TransactionCode alphanumeric	<p>An identifier used to describe the type of transaction - use the transaction code from the servicing system.</p> <p>Note that TransactionId and TransactionCode must be unique.</p> <p>Use the ID in your servicing system - please provide mappings as outlined below.</p> <p>Reversed transactions should have a different transaction code from the original transaction.</p>	BBB
TransactionCodeDesc alphanumeric	A description that correlates to the TransCode identifier such as boarding, scheduled payment, AutoPay fee payment, refund, cancellation, fee assessment, charged off principal, loan modification, etc.	Principal payment

Attribute	Description	Sample Record
<div>Amount</div> <div>numeric</div>	<p>Reflects the amount of this portion - TransactionId + TransactionCode.</p> <p>---</p> <p><b>Note on operators:</b> A transaction that increases the balance should be positive (+), and a transaction that decreases the balance should be negative (-). You should be able to <b>SUM</b> the transactions with the same balance impact code to arrive at the current balance.</p>	277.85
<div>BalanceImpactCode</div> <div>integer</div>	<p>Code that indicates the type of balance impacted by this TransactionId + TransactionCode</p> <p>---</p> <p>0 = No balance impact 1 = Principal 2 = Interest 3 = Fee</p>	1
<div>NonCashFlag</div> <div>bit</div>	<p>A code that indicates if the transaction was cash or not.</p> <p>---</p> <p>0 - Cash transaction 1 - Other non-cash transaction</p>	0
<div>ReversalIndicator</div> <div>bit</div>	<p>A code that indicates the reason for a transaction reversal.</p> <p>---</p> <p>0 - Original transaction 1 - Reversal of original transaction</p>	0
<div>ReversalReason</div> <div>alphanumeric</div>	<p>A description that indicates the reason for reversal such as ACH return, bounced check, etc.</p>	ACH



Attribute	Description	Sample Record
Effective Date date	The date the transaction affected the loan	Format: yyyy-mm-dd e.g. 2014-11-10
RemittedDate ate date	The date the funds are remitted to CR	Format: yyyy-mm-dd e.g. 2014-11-10
PaymentOwner ner alphanumeric	Payment owner	CR

TransactionCode	TransactionCodeDefinition	Principal Impact (y/n)	Interest Impact (y/n)	Fee Impact (y/n)	Cash Flow	Transaction Amount Sign
123	Principal paid by borrower	y	n	n	1	-
456	Principal adjustment (small \$ write off)	y	n	n	0	-
789	Interest paid by borrower	n	y	n	1	-
120	Refund principal	y	n	n	1	-
450	Principal payment reversal	y	n	n	1	+
111	Late fee assessed	n	n	n	0	+
222	Late fee paid	n	n	y	1	-

## 7.13. [Deprecated] Purchase reports

---

After the loan is funded, it is either sold back to the MPL, sold to an investor (directly), sold to the MPL on behalf of an investor (indirectly), or retained by Cross River.

When the loan completes the seasoning period it is **Ready to Sell** to the MPL or an investor. Cross River produces a pre-purchase report, which is an invoice.

### Pre-purchase report (invoice)

The pre-purchase reports are sent to the MPL or investors before the sale.

The invoice reports:

- The list of the loans ready to be purchased
- The loans retained by the bank
- Accrued interest
- The volume fee (if relevant)
- The sum total of funds to be paid to Cross River

The purchaser sends the funds to Cross River on loan origination day, or in the case of an ACH transfer, the day before.

The sum of the funds equals the total amount of the loans + the accrued interest + any bank fees (if relevant).

The pre-purchase reports are available at 0600 Eastern on the SFTP location provisioned by Cross River at the following location:

```
\Prod\MPL\DailyReports\PreLoanSaleReports\PreSale_LoanSales_PlatformName_yyyy-mm-dd.csv (comma delimiter)
```

### Post-purchase report (receipt)

After the MPL or the investor funds Cross River for the purchase, a post-purchase report, a receipt, is sent. The report confirms receipt of the funds and that the account has been reconciled.

The post-purchase reports contains:

- A list of the loans sold to them
- Fees and Interest paid to the bank

The post-purchase reports are made available immediately post sell on the SFTP location provisioned by CR at the following location:

`\Prod\MPL\DailyReports\PostLoanSaleReports\PostSale_LoanSales_PlatformName_yy  
yy-mm-dd.csv` (comma delimiter)

## Accrued interest

Accrued interest is the interest that accumulated on the loan since the loan was funded. The approach to accrued interest and how to calculate it is negotiated with all partners.

## Volume fee

The volume fee (or origination fee), is calculated based on the cost charged for loan origination and the volume of loans made over a specified amount of time. The volume fee is negotiated with all partners.

Cross River collects volume fee at the time of funding.

## Definitions

- At funding - estimated volume fee is collected
- Purchase = Par + accrued Interest
- Day - Calendar day. Purchases take place on the 1st business day following a 3-day seasoning period.
- If ACH is used for purchases, settlement is T+1. Wires settle the same day.

- Purchase account – A CR account to be provided, post board approval. The Routing # is 021214891.
- For MPLs - File name format: LoanSales\_PlatformName\_yyyymmdd.xlsx
- For Investors - File name format: LoanSales\_InvestorName\_yyyymmdd.xlsx

## 8. COS reference codes

---

These codes are not relevant for Card payments or Lending.

### Transaction codes

Code	Description
achIncoming	Incoming ACH payment
achIncomingReturn	Incoming ACH returned payment
achIncomingNoc	Incoming ACH NOC
achOutgoing	Outgoing ACH payment
achOutgoingReturn	Outgoing ACH returned payment
achOutgoingNoc	Outgoing ACH NOC
accountTransfer	Internal book transfer
deposit	Deposit
withdrawal	Withdrawal
interestPayment	Interest payment
interestReversal	Interest Reversal
miscellaneousCredit	Misc Credit
miscellaneousDebit	Misc Debit
wireIncoming	Incoming Wire payment
wireOutgoing	Outgoing Wire payment
check	Check
cash	Cash
reversal	Reversal
adjustment	Adjustment
sale	Sale
cardAuth	Card Auth
payment	Payment
refund	Refund
atmTransaction	ATM Transaction
rtpDebit	RTP Debit

Code	Description
rtpCredit	RTP Credit
xPayDebit	XPAY Debit
xPayCredit	XPAY Credit
withholding	Withholding
sweepInterestPayment	Sweep Interest
chargeback	Chargeback
bonus	Bonus
reward	Reward
cryptoPurchase	Crypto Purchase
cryptoSale	Crypto Sale
cryptoOutgoing	Crypto Outgoing
cryptoIncoming	Crypto Incoming
cryptoTransfer	Crypto Transfer
cryptoReturn	Crypto Return
internationalOutgoing	INTL Outgoing
internationalReturn	INTL Return

## Rejection codes



Code	Description
ERR	General Error
RES	Account Restriction
NSF	Insufficient Funds
ANF	Account Not Found
CLS	Account Closed
INA	Account Inactive
DRM	Account Dormant
ESC	Account in escheatment
CHR	Account in charge-off status
STP	Stop payment active

# Transaction flags

Flag	Description
FRC	Force Posting. Overrides most authorization rules.
NSF	Overrides sufficient funds rule
RES	Overrides all account and product restrictions
STP	Overrides all stop payments
PRE	Indicates transaction is part of pre-settlement phase of daily processing.
STS	Override account status rule
RGD	Exclude transaction from counting towards Reg D debit limit
AUTH	Indicates the transaction is for authorization purposes only. No money will move between accounts as a result of this transaction. No account activity record will be written as a result either.
CID	Company ID
RTN	Routing number
CHK	Check number
ACC	Account number
TYP	Primary type
SUB	Sub type
DIR	Direction
RES	Reason code
MEM	Memo
DAT	Data
SRC	Source
REF	Reference
EXP	Expiration
SVC	Service type
TRC	Trace number

Flag	Description
NID	Network ID
TID	Transaction ID
RID	Receiver ID
OID	Originator ID
BID	Batch ID

## Account sources

Source	Description
Transaction	Activity from transactions will first soft post, remaining as status pending for the day, and later hard post during the nightly settlement process. Once hard posted a sequence number and running balance are assigned.
MemoPosts	Activity from memo posts are a temporary debit or credit on the account. Commonly they will be removed and then posted as a transaction.

## Activity status

Status	Description
Pending	Memo posts will remain pending until they are removed (commonly a day or more). Transactions will always start as status pending when they first soft post.
Posted	Transactions will always transition to status posted during the nightly settlement process for the account. Once hard posted, a sequence number and running balance are inserted.
Removed	Only memo posts can be removed

## Activity types

Type	Description
Internal	Activity is between two sub accounts under the same master account
External	Activity is between different master accounts

## Closure reason

Reason returned
CustomerInitiatedNoComplaint
CustomerComplaintAccountClosing
CustomerComplaintAddOnProducts
CustomerComplaintApplicationProcessing
CustomerComplaintInterestRate
CustomerComplaintBillingDispute
CustomerComplaintBillingStatement
CustomerComplaintCollectionPractices
CustomerComplaintCreditDetermination
CustomerComplaintCreditReporting
CustomerComplaintCustomerService
CustomerComplaintFraud
CustomerComplaintFundAvailability
CustomerComplaintLateFees
CustomerComplaintMerchantDispute
CustomerComplaintOnlineAccountManagement
CustomerComplaintOtherFees
CustomerComplaintBalanceIssue
CustomerComplaintPromoDispute
CustomerComplaintMarketing
CustomerComplaintSettlementHardshipPrograms
BankPartnerInitiatedNoBalance
PartnerProductClosure
FirstPartyFraud
ThirdPartyFraud

### Reason returned

FinancialCrimeNonFraud

HighOverdraftAmount

ExceededTransactionLimits

AccountInactivityDormancyEscheatment

TermsOfServiceBreach

BulkClosureExcerciseOther

ChargeOff

## Transaction status

### Status

### Description

Pending

Accepted, but not yet processed

Complete

Processed successfully. Money has been moved between the two accounts

Rejected

Processing failed due to one or both sides of the transaction being rejected. See rejection codes for details.

Review

Transaction waiting for approval from Ops team before it can be processed.

Declined

After review by Ops team the transaction was declined to be processed

Canceled

While a transaction is pending it may be canceled.

## Hold reasons

Reason	Description
Availability	Relates to the availability of funds for check deposits and ACH payments
Management	Bank management has placed a hold
Collateral	
Safeguard	
CardAuth	Debit card authorization
Other	

## Hold status

- Active
- Inactive

## Hold types

- Internal
- Partner

## Identification types

Type
DriversLicense
Passport
IdCard
SocialSecurity
Other

## Memo transaction types

# OFAC/PEP status

Status	Description
Pending	A scan is being performed on the customer.
Clear	A scan has been completed and the customer has been cleared.
Failed	A scan has been completed but the customer must be reviewed by our BSA/AML Team.
Reviewing	The customer is currently being reviewed by our BSA/AML Team.

# Ownership type

Type
Exempt
Excluded
NonProfit
LegalEntity

# Rail type

Type	Description
ACH	ACH Network
Wires	Wire Network
CreditCard	Credit Card
DebitCard	Debit Card
StoredValue	Stored value card network (e.g. gift cards)
ICL	Image Cash Letter (i.e. checks)
XPay	XPay rail which enables instant transfers between COS accounts
Internal	Internal book transfer



# Relationship status

- Active
- Inactive

# Relationship type

Type
Primary
AuthSigner
Joint

# Restriction applies to

Type	Description
All	Master and sub accounts
Account	Master account only
SubAccounts	Sub accounts only

# Restriction status

- Active
- Inactive

# Risk ratings

- Low
- Medium
- High

# Customer entity types

Entity Type
Limited Liability Company
Sole Proprietorship
Corporation
Not for Profit
Investment/Hedge/Private Equity Fund
Trust
Special Purpose Vehicle/Special Purpose Entity
Private Investment Company/Private Holding Company

# Statement status

Status	Description
Processing	Statement is generating
Completed	Statement data is fully available and all associated pdf documents have been generated

# Stop payment status

- Active
- Inactive

# Subledger type

Type	Description
Passthrough	Subledger balance is informational only and will not be considered when making transaction decisions for the account. Only the master balance is used for transaction decisions. Subledgers can go negative if needed, as long as the funds are in the master account.
Direct	Subledger balance is used for transaction decisions. Implicit subledger cannot go negative.

## Tax ID type

Type	Description
SSN	Social security number
EIN	Employer identification number
ITIN	Individual tax ID number
VAT	International VAT number

## Transaction type

- Debit
- Credit

## Error codes

Code	Description
1000	General exception
1200	DateFormed required
1201	DateFormed cannot be in the future
1202	BirthDate required
1203	BirthDate cannot be in the future
1204	EntityName required
1205	FirstName not allowed for classification
1206	MiddleName not allowed for classification
1207	LastName not allowed for classification
1208	Prefix not allowed for classification
1209	Suffix not allowed for classification
1210	PreferredName not allowed for classification
1211	FirstName required
1212	LastName required
1213	EntityName not allowed for classification
1214	Name required
1215	PoliticallyExposedPerson required
1216	PoliticallyExposedPerson not allowed for classification
1217	BirthDate not allowed for classification
1218	DateFormed not allowed for classification
1219	RegO required
1220	RegO not allowed for classification
1221	PrimaryOwnerCustomerId required
1222	PrimaryOwnerCustomerId not allowed for classification
1223	EntityType not allowed for classification

Code	Description
1224	OwnershipType not allowed for classification
1225	TaxIdType not allowed for classification
2000	General exception
2001	Account not active
2002	Status cannot be set to closed manually
2003	Account not open
2004	Account is closed
2005	Account is not closed
2006	Account cannot be closed until active holds removed
2007	Account cannot be closed until active memo posts removed
2008	Account cannot be closed until all completed transactions have been applied and balance is zero
2009	Account cannot be closed until sub accounts are closed
2010	Max total accounts exceeded for product
2011	Max active accounts exceeded for product
2012	Max daily accounts exceeded for product
2013	Account already exists
2014	Account number cannot be generated
2015	Account generation attempts exceeded
2016	Invalid account number prefix
2017	Invalid account number length
2018	Customer not assigned to same partner as product
2019	Customer classification not allowed for product
2020	Customer must have an address associated
2021	Customer must have a phone number associated

Code	Description
2022	Customer must be cleared for OFAC prior to account opening
2100	Sub account not found
2101	Sub account not active
2102	Cannot modify implicit sub account
2103	Sub account already exists
2104	Sub account is closed
2105	Sub account is not closed
2106	Master account required
2107	Master account not open
2108	Master account not active
2109	Master account not found
2110	Beneficiary required
2111	Sub account total limit has been reached for master account
2112	Sub accounts are not enabled for product
2113	Sub account number cannot be generated
2114	Sub account generation attempts exceeded
2115	Invalid sub account number prefix
2116	Invalid sub account number length
2117	Sub account cannot be closed until all completed transactions have been applied and balance is zero
2118	Sub account daily limit has been reached for master account
2119	Sub account active limit has been reached for master account
2200	Customer not found
2201	Customer not active
2202	Entity not active

Code	Description
2203	Address not active
2204	Cannot remove primary address
2205	DateFormed required
2206	DateFormed cannot be in the future
2207	BirthDate required
2208	BirthDate cannot be in the future
2209	Primary owner customer not found
2210	Primary owner customer not active
2211	Primary owner customer must be a person
2212	Entity and primary owner must be under same partner
2213	Entity and primary owner must be different customers
2214	Entity customer must be a business
2215	Owner customer not found
2216	Owner customer not active
2217	Owner customer must be a person
2218	Entity and owner customers must be under same partner
2219	Entity and owner must be different customers
2220	Owner relationship already exists for entity customer
2221	Unable to deactivate customer due to established account relationship(s)
2222	Partner not found or is inactive
2223	An active customer with this tax ID already exists for partner
2224	Cannot remove primary identification
2225	Cannot remove primary phone
2226	Invalid customer entity type
2350	Hold not found

Code	Description
2351	Hold is inactive
2352	Invalid hold amount
2353	Expiration date cannot be in the past
2375	Restriction not found
2376	Restriction not active
2377	Restriction policy in request must match restriction policy associated with change
2378	Restriction policy approval requires dual control
2379	Restriction policy not found
2380	Restriction policy not active
2381	Restriction policy max restrictions exceeded
2382	Restriction policy name already exists
2383	Pending restriction policy change with name already exists
2384	Restriction already inactive
2385	Restriction policy change not approved
2386	Restriction policy change not pending
2425	Interest accrual not found
2426	Start date must be before end date
2427	Amount must be greater than or equal to zero
2428	Payments not equal to zero must be posted to core
2450	Memo post not found
2451	Memo post not active
2452	TransactionCode is invalid
2475	Product not found
2476	Product not active



Code	Description
2477	Product change not approved
2478	Product change not pending
2479	Product in request must match product associated with change
2480	Classification not allowed for product type
2481	Product type invalid
2482	Product name already exists
2483	Unknown account type for product
2484	Pending product change with name already exists
2485	Interest Accrual not allowed for account type
2487	Statements not allowed for account type
2488	Product change approval requires dual control
2525	Relationship not found
2526	Relationship not active
2527	Primary relationship cannot be removed
2528	Relationship not allowed for account type
2529	RelationshipType not allowed for account classification
2530	IsTaxReportingOwner cannot be enabled for relationship type
2531	Customer must be classified as a person
2532	Customer relationship already exists
2533	Customer must be under same partner as account
2534	Customer must be cleared for OFAC prior to creating relationship
2550	Settlement not found
2551	Daily settlement must be pending
2552	Daily settlement must not be complete
2553	Daily settlement must be pre-settling and all interest must be processed

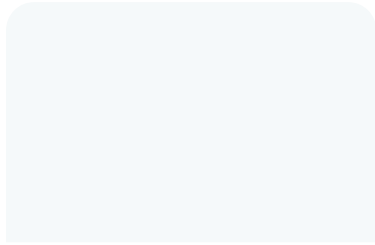
Code	Description
2554	All accounts must be settled before daily settlement can be completed
2575	Snapshot not found
2576	Snapshot account number does not match
2577	Snapshot partner does not match
2600	Statement not found
2601	Statement not completed
2602	Statement is closed
2603	AccountNumber must match statement
2604	Unable to generate statement for account type
2625	Transaction not found
2626	Debit account not found
2627	Debit account not active
2628	Credit account not found
2629	Credit account not active
2630	Debit and credit accounts must be different
2631	Debit account type not allowed
2632	Credit account type not allowed
2633	General ledger transfers require auto approval permission
2634	Transaction must be pending
2635	Transaction not effective
2636	Transaction must be under review
2637	Transaction authorization not found
2638	Transaction approval requires dual control
2639	Invalid transaction status
2640	PartnerId not accessible

Code	Description
2641	PartnerId is required for user type
2642	TransactionCode is invalid
2643	Max transaction flags exceeded
2644	Invalid transaction flag
2645	Transaction flags not allowed
2646	Transaction filter required
2647	Transaction trace number required
2648	Transaction proposal required
2649	Transaction amount must be greater than zero
2650	Transaction auth status invalid
2700	Accounting export is not processing

## 9. Status page

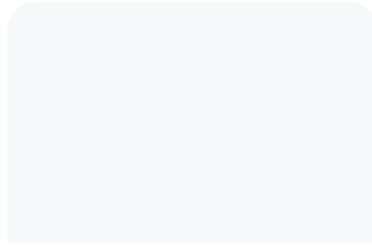
---

These pages contain the system status of our Sandbox and Production environments.



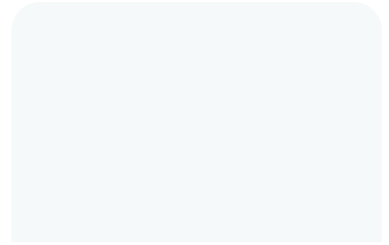
### **COS**

Real-time information  
covering Bank Rails,  
Accounts & Card Issuing



### **Card Payments**

Real-time information  
covering Push and Pull  
to card



### **Lending**

Real-time information  
covering digital lending

# 10. Changelog

The information below contains a list of new features by release.

Please reach out to your Implementation or Relationship Manager with any questions.

You can subscribe at <https://crossriverbank.statuspage.io/> for updates about releases.

**Next release is scheduled for December 1, 2024**

## Nov 3, 2024 Release

Feature	Description
Card Payments - Multiple Business Types	CRB Card Payments merchant partners are now able to send different business types (BAI codes) on each transaction by optionally specifying the <code>businessType</code> on each request. We now also return the <code>businessType</code> on every payment

## Sep 29, 2024 Release

### Deployed to Production

Feature	Description
ACH Later Settlement	Support for later settlement window: CR can now receive standard ACH payments until 1:45 AM EST to meet the 2:15 AM EST batch distribution deadline to be settled by 8:30 AM EST the same day.
International Payments	International Payments live! Cross River will utilize local Rails and SWIFT to send money to tier 1 countries. International Payments can be accessed via API or COS Explorer.

# Sept 22, 2024 Release

## Deployed to Sandbox

Feature	Description
ACH Later Settlement	Support for later settlement window: CR can now receive standard ACH payments until 1:45 AM EST to meet the 2:15 AM EST batch distribution deadline to be settled by 8:30 AM EST the same day.
International Payments	International Payments live! Cross River will utilize local Rails and SWIFT to send money to tier 1 countries. International Payments can be accessed via API or COS Explorer.

# July 14, 2024 Release

## Deployed to Production

Feature	Description
Wire Drawdowns	You can now initiate an outbound wire drawdown request, also called a reverse wire, to ask someone at a different bank to send funds back to you, the requester.
COS Explorer	Wire Authorization: 2 authorized partner users can now approve wires up to \$5M on COS Explorer, eliminating the need for callbacks from the CR Wires Operations team.

# June 30, 2024 Release

## Deployed to Sandbox

Feature	Description
Wires	You can now initiate an outbound wire drawdown request, also called a reverse wire, to ask someone at a different bank to send funds back to you, the requester.
COS Explorer	In Explorer, you can now initiate an outbound wire drawdown request, also called a reverse wire, to ask someone at a different bank to send funds back to you, the requester.

## June 16, 2024 Release

### Deployed to Production

Feature	Description
Wires	The Wire.Payment.Received webhook has been updated. When you receive the webhook for a wire that posted, you can now see the SenderRoutingNumber, Originating FI Identifier, and Originating FI name.
COS Explorer	We're updating tab behavior and now Explorer remembers the last tab you had open.
COS Explorer	We've made updates to the login screen. This is part of ongoing improvement efforts.
Instant Payments	You can now return RTP payments in COS Explorer.

## June 2, 2024 Release

### Deployed to Sandbox

Feature	Description
Wires	The Wire.Payment.Received webhook has been updated. When you receive the webhook for a wire that posted, you can now see the SenderRoutingNumber, Originating FI Identifier, and Originating FI name.
COS Explorer	We're updating tab behavior and now Explorer remembers the last tab you had open.
COS Explorer	We've made updates to the login screen. This is part of ongoing improvement efforts.
Instant Payments	You can now return RTP payments in COS Explorer.

## May 19, 2024 release

### Deployed to Production

Feature	Description
COS Explorer	You now have the ability to filter Deposit Subledger accounts by Entity Name.

## April 14, 2024 release

### Deployed to Production



Feature	Description
Network Interoperability	RTP payments now use network interoperability to deliver your payments to the creditor as efficiently and effectively as possible. When you originate an Instant Payment using POST /rtp/v1/payments, we'll take care of the routing for you – accessing any endpoint on either the RTP® or FedNow® networks. <a href="#">Check out how it works</a>
International Payments	Our International Payment offering and endpoints are live! Check out the <a href="#">documentation and tutorials</a> where you can easily send to 20+ countries using 9 different currencies – with just a few simple API calls.
COS Explorer	Any partners who haven't yet migrated to Cross River's <b>Identity Server</b> will automatically be prompted to the migration flow when they login. <a href="#">Check out the COS Explorer login information</a> .
COS Explorer	For <b>ACH transactions</b> , you can now search in COS Explorer by Client ID for the last 180 days.

















