DIGITAL RIVER API REFERENCE GUIDE

API structure

Understand the Digital River API structure.

The Digital River API is a RESTful API. That means we designed the API to allow you to create, read, update, and delete objects with `POST`, `GET`, `PUT` and `DELETE` HTTP methods.

The Digital River API speaks exclusively in JSON. So to ensure the API accepts and processes your requests, always set the `Content-Type` header to `application/json`.

The API key you use to authenticate a request determines whether the request is handled in test mode or production mode. You should use the Digital River API in test mode when you don't want to interact with financial institutions.

All Digital River API requests are sent to `https://api.digitalriver.com`.

We recommend that you include specific HTTP headers for call debugging purposes.

For more information on how Digital River ensures high availability and performance, see our Service Level Agreement.

API keys

Digital River uses your account's API keys to authenticate API requests. We return an error if you fail to include your key in a request or send an outdated or incorrect key. You can always access your keys in Digital River Dashboard.

Digital River provides each account with four standard keys: a public and confidential (secret) pair for test and live modes. In certain scenarios, you may also find it useful to use restricted keys.

Public keys

Your public API key identifies your account with Digital River and allows you to create payment sources. You use public keys with Drop-in payments and DigitalRiver.js.

Confidential keys

Your confidential (secret) API key allows you to send an API request to Digital River without restriction. Keep these secret keys confidential and only store them on your servers. Don't use your confidential key for everything. Instead, consider using restricted keys to limit access to your environment.

Restricted keys

If you want more security, you can create restricted API keys. A restricted key reduces risk when using or building services. It provides the minimum access and permissions a service needs to access specific resources in the Digital River API. Use restricted keys to limit access to services interacting with the Digital River API. You can name and assign a restricted key to the service you want.

Test and production environments

In the test environment, you can test Digital River's features without affecting live data. The production environment allows you to perform actions that affect live data.

In API responses, the `liveMode` boolean attribute indicates whether you are in the test or production environment.

The test and production environments behave similarly with the following exceptions:

- You can only use test payment information in the test environment. Card networks and payment providers do not process payments in a test environment.
- Some payment methods have a different flow when using payment sources in the production environment, and they may require more steps in the test environment.
- In the test environment, Digital River retries webhooks three times over a few hours (as opposed to 72 hours for the production environment) when it does not receive a successful acknowledgment.

Headers for troubleshooting

We highly recommend including the upstream identifier, upstream application identifier, upstream session identifier, and the customer's IP address as HTTP request headers when making calls to the Digital River APIs.

These request headers aid Digital River in performing internal troubleshooting and call debugging. Additionally, these headers will eventually be available to clients in call logs. They will also be searchable on Digital River Dashboard. So, in the future, you can use these headers to search your own logs.

Upstream identifier

The `upstream-id` request header should represent the unique identifier of the upstream request. This identifier should be __generated by the upstream application.

Upstream application identifier

The `upstream-application-id` request header should be an identifier generated by the upstream application that identifies all the calls made by that application. You should use the following format to pass the application name and its version: `<application>/<version>`. For example, the header value might be `EnterpriseApp/1.1. 5`.

By providing this identifier, you allow Digital River to filter call logs by application.

Upstream session identifier

In a request, the `upstream-session-id` header should denote the identifier of a browser session. More specifically, this value identifies a single customer session in the upstream application, thereby allowing us to track that customer's path through the Digital River APIs.

End user's IP address

The `forwarded` request header can designate one or more IP addresses. This header is primarily used to capture a customer's IP address (in this case, the customer represents the end user of the upstream application). If you use `forwarded` to pass more than one IP address, the value farthest to the right should denote the customer's IP address. In the following example, `for=198.51.100.17` is the customer's IP address.

```
...
forwarded: for=192.0.2.43, for=198.51.100.17,
...
```

Dates and times

In the Digital River API, we adhere to ISO-8601, an international standard that provides an unambiguous and well-defined method of representing dates and times.

In every API response, date-time representations follow this ISO-8601 standard. You rarely have to include dates and times in API requests, but when you do, they must also observe this same standard.

The following are the specific date-time rules we use:

- Dates and times are arranged so the largest temporal term (the year) is placed to the left, and each successively smaller term is placed to the right of the previous term. In other words, the values are ordered from the largest to smallest unit of time: year, month, day, hour, minute, and second.
- Each date and time value has a fixed number of digits padded with leading zeros.
- The hyphen separates date values (year, month, and day), while the colon separates time values (hours, minutes, and seconds).
- All numeric values placed to the right of `T` represent times.
- Time values are in Coordinated Universal Time (UTC) with no time zone offset.
- Since the letter `Z` is the zone designator for zero UTC offset, we add it (without a space) after the time values.

Checkout

```
{
    "id": "a088982a-9329-41f2-9061-573b7ecbeaa9",
    "createdTime": "2021-04-07T13:47:13Z",
    ...
    "updatedTime": "2021-04-07T13:47:13Z",
    ...
}
```

Previous
Digital River API reference

Next
Best practices

Last updated 4 months ago