Production

API Reference

Notify API Quickstart

Notify API Quickstart

The Notify API manages transaction notifications using webhooks.

Alchemy Notify is used to subscribe to event notifications that occur on your application. You create a webhook to receive notifications on different types of on-chain activity. Here are a few quick-links:

- Webhook Types
- How to Set up Webhooks

Notify webhook types

Below are the Alchemy webhook types and their supported networks.

Webhook Type	Description	Network
Mined Transaction	The Mined Transaction webhook notifies your app when a transaction sent through your app (using your API key) gets mined. This is useful for you to further notify the users of your app about the status of the transaction.	All
Dropped Transactions	The Dropped Transaction webhook notifies your app when a transaction sent through your app (using your API key) gets dropped. This is useful for you to further notify the users of your app about the status of the transaction.	All
Address Activity	Alchemy's Address Activity webhook tracks all ETH, ERC20, ERC721 and ERC1155 transfers. This provides your app with real-time state changes when an address sends/receives tokens or ETH. You can specify the addresses for which you want to track this activity. A maximum of 50,000 addresses can be added to a single webhook.	All
NFT Activity	The NFT Activity webhook allows you to track ERC721 and ERC1155 token contracts for Ethereum NFTs. This provides your app with real-time state changes when an NFT is transferred between addresses.	Ethereum Mainnet & Goerli
NFT Meta Updates	The NFT Metadata Updates webhook allows you to track metadata updates for ERC721 and ERC1155 token contracts for Ethereum and Polygon NFTs. This notifies your app when the metadata for an NFT is updated.	Ethereum Mainnet & Goerli; Polygon Mainnet & Mumbai

V2 Webhook

Field definitions

Field	Description	Value
`webhookId`	Unique ID of the webhook destination.	`wh_octjglnywaupz6th`
`id`	ID of the event.	`whevt_ogrc5v64myey69ux`
`createdAt`	The timestamp when webhook was created.	`2021-12-07T03:52:45.899Z`
`type`	Webhook event type.	`TYPE_STRING`
`event`	Object-mined transaction.	`OBJECT`

Example

V2

```
{
 "webhookId": "wh_octjglnywaupz6th",
 "id": "whevt_ogrc5v64myey69ux",
 "createdAt": "2021-12-07T03:52:45.899Z",
 "type": TYPE_STRING,
 "event": OBJECT
}
```

V1 Webhook Event Object

Field definitions

Field	Description	Value
`app`	Alchemy app name sending the transaction webhook.	`Demo`
`network`	Network for the webhook event.	`MAINNET`

Field	Description	Value
`webhookType`	The type of webhook.	`MINED_TRANSACTION`
`timestamp`	Timestamp when the webhook was created.	`2020-07-29T00:29:18.414Z`
`event name`	Webhook event type.	`OBJECT`

For Notify full dependencies and more code examples, go to the GitHub repo.

Example Response

v1

```
{
  "app": "Demo",
  "network": "MAINNET",
  "webhookType": "MINED_TRANSACTION",
  "timestamp": "2020-07-29T00:29:18.414Z",
  "event name": OBJECT
}
```

How to set-up webhooks

Setting up a webhook is as simple as adding a new URL to your application.

> **!** **NOTE:**
>
> If you need to add over 10 addresses to the address activity webhook, we recommend adding them through an API call.

Set-up webhooks in your dashboard

1. Navigate to the Notify tab in your Alchemy Dashboard.
2. Determine which type of webhook you want to activate.
3. Click the **Create Webhook** button.
4. Select the app to add the webhook notifications.
5. Add in your unique webhook URL. This is the link to receive requests. The webhook payload might not always be compatible for 3rd party integrations.
6. Test your webhook by clicking the **Test Webhook** button.
7. Click **Create Webhook** and your webhook appears in the list.
8. Check your endpoint to see responses.

Set-up webhooks programmatically

Use the `create-webhook` endpoint: https://docs.alchemy.com/reference/create-webhook.

Webhook IP addresses

As an added security measure, you can ensure your webhook notification originates from Alchemy by using one of the following IP addresses:

- `54.236.136.17`
- `34.237.24.169`

Create webhook listeners

Webhook listeners receive requests and process event data.

The listener responds to the Alchemy server with a `200` status code once you've successfully received the webhook event. Your webhook listener can be a simple server or Slack integration to receive the webhook listener data.

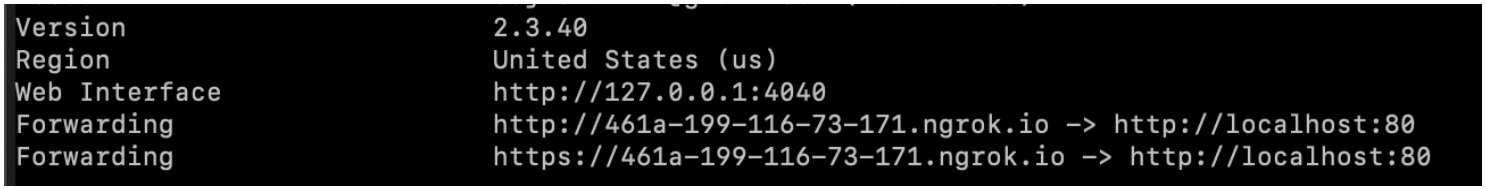
After setting up the webhooks in your Alchemy dashboard (or programmatically) use the starter code in JavaScript, Python, Go, and Rust below. Here's the GitHub repository for the entire code.

JavaScript Python Go Rust

```
  const app = express();

  setDefaultEnvVar("PORT", "8080");
  setDefaultEnvVar("HOST", "127.0.0.1");
  setDefaultEnvVar("SIGNING_KEY", "whsec_test");

  const port = +getRequiredEnvVar("PORT");
  const host = getRequiredEnvVar("HOST");
  const signingKey = getRequiredEnvVar("SIGNING_KEY");

  // Middleware needed to validate the alchemy signature
  app.use(
    express.json({
      verify: addAlchemyContextToRequest,
    })
  );
  app.use(validateAlchemySignature(signingKey));

  // Register handler for Alchemy Notify webhook events
  // TODO: update to your own webhook path
  app.post("/webhook-path", (req, res) => {
    const webhookEvent = req.body as AlchemyWebhookEvent;
    // Do stuff with with webhook event here!
    console.log(`Processing webhook event id: ${webhookEvent.id}`);
    // Be sure to respond with 200 when you successfully process the event
    res.send("Alchemy Notify is the best!");
  });


  // Listen to Alchemy Notify webhook events
  app.listen(port, host, () => {
    console.log(`Example Alchemy Notify app listening at ${host}:${port}`);
  });
}

main();
```

Test webhooks with Ngrok

1. Sign-up for a free Ngrok account.
2. Install Ngrok using the Ngrok guide. On macOS run `brew install ngrok`.
3. Connect your Ngrok account by running `ngrok authtoken YOUR_AUTH_TOKEN`.
4. Start your local forwarding tunnel: `ngrok http 80`.



Once you have a URL to test your webhook (in this case `https://461a-199-116-73-171.ngrok.io` pictured above), follow the steps below:

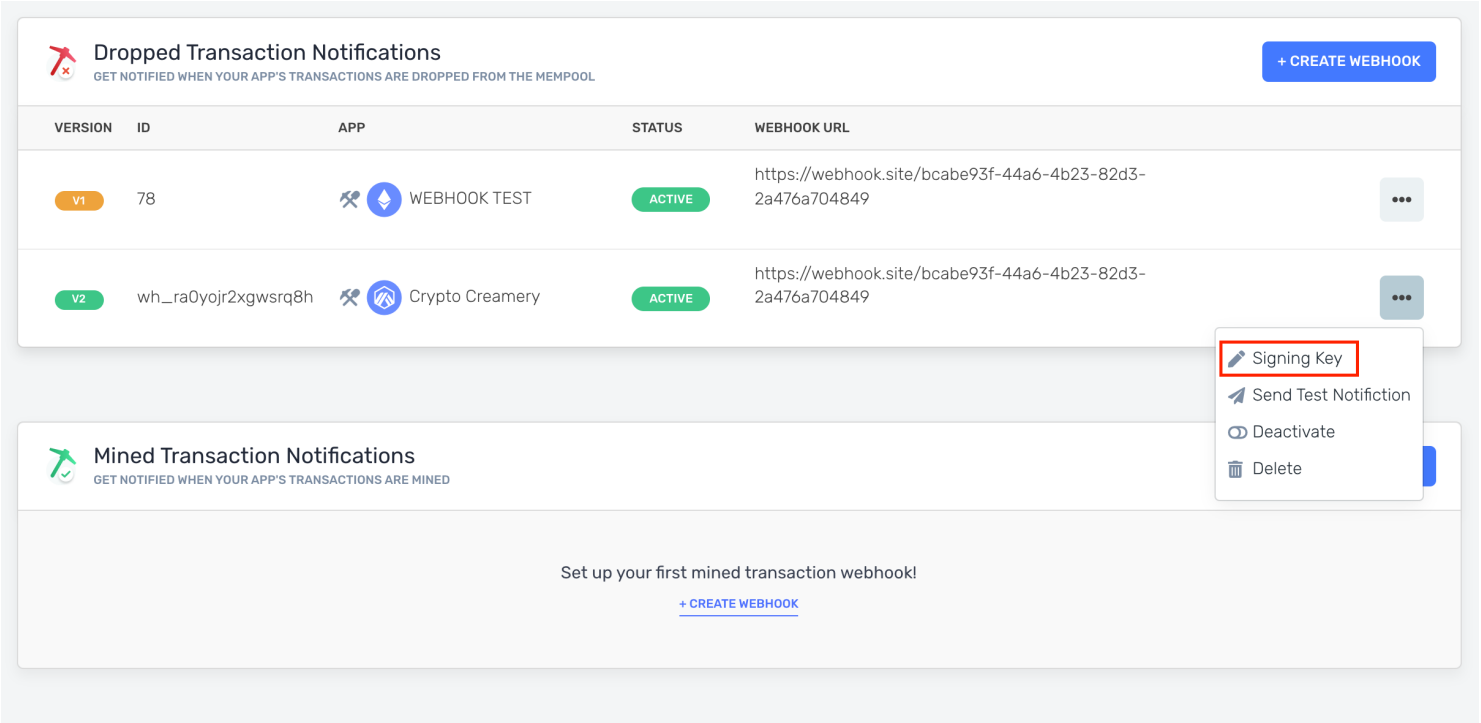
1. Navigate to your Notify dashboard.
2. Click **Create Webhook** on the webhook you want to test.
3. Paste **your unique URL** and hit the **Test Webhook** button.
4. You'll see the webhooks here: `http://localhost:4040/inspect/http`.

Webhook signature & security

To make your webhooks secure, you can verify that they originated from Alchemy by generating a HMAC SHA-256 hash code using your unique webhook signing key.

Find your signing key

Navigate to the Notify page in your dashboard. Next, click on the three dots of the webhook you want to get the signature for and copy the Signing Key.



Validate the signature received

Every outbound request contains a hashed authentication signature in the header. It's computed by concatenating your signing key and request body. Then generates a hash using the HMAC SHA256 hash algorithm.

To verify the signature came from Alchemy, you generate the HMAC SHA256 hash and compare it with the signature received.

Example request header

Request header

```
POST /yourWebhookServer/push HTTP/1.1
Content-Type: application/json;
x-alchemy-signature: your-hashed-signature
```

Example signature validation

JavaScript | Python | Go | Rust

```
import * as crypto from "crypto";

function isValidSignatureForStringBody(
    body: string, // must be raw string body, not json transformed version of the body
    signature: string, // your "x-alchemy-signature" from header
    signingKey: string, // taken from dashboard for specific webhook
  ): boolean {
    const hmac = crypto.createHmac("sha256", signingKey); // Create a HMAC SHA256 hash using th
    hmac.update(body, "utf8"); // Update the token hash with the request body using utf8
    const digest = hmac.digest("hex");
    return signature === digest;
}
```

Webhook retry logic

Alchemy Notify V2 has built-in retry-logic for webhooks. Requests are retried for non-200 response codes in failures to reach your server.

Requests are retried up to 6 times before failing. Below are the request retry intervals.

- 15 seconds
- 1 minute
- 10 minutes
- 1 hour
- 1 day
- 1 day

Updated 5 days ago

← alchemy_getTokenAllowance

Notify API FAQ →

Token API

View all

Overviews

Gwei Calculator

COMPANY

About Us

Customers

Newsroom

Careers

Blog

Press Kit

Terms of Service

CONTACT

General Inquiries

Press

Sales

Discord

Email