

## INTEGRATIONS • FOR IT ADMINS

# Automate Clouddraw with the API

Create API tokens, call the Clouddraw API from scripts, and receive events with signed webhooks.

Last updated 7 October 2026

## Overview

The Clouddraw API is the same API the admin console uses. With it, scripts and tools can read and change your workspace: people, groups, devices, apps, access rules and more.

There are three ways to connect other systems to Clouddraw:

- **API tokens:** your scripts call Clouddraw. See [Step 1](#).
- **Webhooks:** Clouddraw calls your system when something happens. See [Webhooks](#).
- **SCIM:** your identity provider creates and removes people. See [Provisioning people](#).

## You need

- The *Owner* or *Security admin* role to create API tokens and webhooks. An *Auditor* can see them.
- Your **workspace ID**. Every API path contains it.
- A tool that sends HTTPS requests, such as `curl` or PowerShell `Invoke-RestMethod`.

## Step 1. Create an API token

### 1 Open API tokens

In the Cloudraw admin console go to **Settings** → **API tokens** and click **Create token**.

### 2 Name it

Use a name that says who uses it, for example `HR sync script`. The name shows in the audit log for every change the token makes.

### 3 Choose the scope

See the table below. The default is **Read-only**.

### 4 Copy the token

The token starts with `crw_` and is shown **once**. Store it in a password manager or your secrets store. The list later shows only its first characters, when it was created and when it was last used.

| Scope                                       | What the token can do                                                                                                       |
|---------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------|
| <b>Read-only</b> ( <code>read-only</code> ) | <code>GET</code> requests only. Any change returns <code>403 insufficient-scope</code> . Use it for reports and monitoring. |
| <b>Full</b> ( <code>full</code> )           | Read and change everything in this workspace.                                                                               |

#### WARNING

A **full** token is not limited by admin roles. It can change access rules, remove people and create more tokens. Treat it like an owner password. Give each script its own token, so you can revoke one without breaking the others.

A token works only in its own workspace. It can never reach another workspace.

## Rotate and revoke

- **Revoke**: click **Revoke** next to the token. It stops working at once.
- **Rotate**: tokens do not expire on their own. To rotate, create a new token, put it in your script, check the script works, then revoke the old token. The list's *last used* time shows when the old token is no longer in use.

## Step 2. Make requests

### Base URL and paths

All requests go to the Cloudraw API over HTTPS:

```
https://orchestrator.cloudraw.com/v0/tenants/<workspace-id>/...
```

For example, people are at `/v0/tenants/<workspace-id>/people`. Send and receive JSON. Lists come back as `{"items": [ ... ], "next_page": null}`.

### Authentication

Send the token in the `Authorization` header on every request:

```
Authorization: Bearer crw_<your-token>
```

### Safe retries: Idempotency-Key

Send an `Idempotency-Key` header on every `POST`. Use a new random value (for example a UUID) for each new action, and the **same** value when you retry that action.

- Same key and same body within 24 hours: Cloudraw does not do it again. It returns the first answer.
- Same key with a different body: `409 idempotency-key-reuse`.

So a retry after a timeout never creates a second app or a second rule.

### Avoid overwriting changes: If-Match

Many objects have a `version` number that goes up with every change. When you change or delete one, send the version you read in `If-Match`:

```
If-Match: 3
```

- If someone changed the object since you read it, you get `409 version-conflict`. Read it again and redo your change.
- Access rules and trust rules **require** `If-Match` on `PATCH` and `DELETE`. Without it you get `428 if-match-required`. For apps it is optional, but recommended.

### Errors

Errors use the standard problem format (RFC 7807), with content type `application/problem+json`:

```
{
  "type": "https://cloudraw.com/problems/invalid-allowed-apps",
  "code": "invalid-allowed-apps",
  "title": "invalid allowed_apps",
  "status": 400,
  "detail": "strict mode needs the publisher (the signer name, e.g. Microsoft Windows)",
  "errors": [ { "path": "allowed_apps.apps[0].publisher", "message": "strict mode needs the publisher (the signer name, e.g. Microsoft Windows)" } ]
}
```

- `code` is stable. Use it in your scripts to tell errors apart.
- `detail` is a sentence for people.
- `errors[]`, when present, lists each bad field by `path`. Some checks, such as access rules, put the field list in `detail` instead, for example `"match.children[1].value: ..."`.

Long actions such as deletes can answer `202` with a small task object. Its `state` is `done` when the change is complete.

# Webhooks

A webhook sends Clouddraw events to your own HTTPS address as they happen. It carries the same events as the console notifications: for example a connector going offline, a device losing trust, an access request, or an app firewall alert.

## 1 Add the webhook

In **Settings** → **Webhooks**, click **Add webhook**. Enter your HTTPS address and choose the event types: `security`, `connector`, `device`, `access`, `audit`, `jit`, `billing`, or `all`.

## 2 Copy the signing secret

It starts with `whsec_` and is shown **once**. Your receiver needs it to check that requests really come from Clouddraw.

## 3 Send a test

Click **Test**. Clouddraw sends one real signed test event and shows your server's answer. You can test 10 times per hour.

Each event is a `POST` with a JSON body:

```
{ "id": "evt...", "tenant": "<workspace-id>", "category": "device", "at": "2026-10-07T09:15:00.000Z",  
  "event": { "action": "device.trust.untrusted", "severity": "warn", "message":  
    "Device LAPTOP-12 is now UNTRUSTED (was trusted)", "data": { ... } } }
```

## Check the signature

1. Read the headers `X-Clouddraw-Timestamp` (Unix seconds) and `X-Clouddraw-Signature-V1` (`sha256=<hex>`).
2. Compute HMAC-SHA256 with your signing secret over `<timestamp>.<raw request body>`, as hex.
3. Compare it to the header value. Reject the request if it does not match, or if the timestamp is more than 300 seconds from your clock.
4. Use the event `id` to drop duplicates. A retried event keeps the same `id`.

An older header, `X-Clouddraw-Signature` (HMAC of the body only), is still sent for now. Use the V1 header for new receivers.

- **Delivery:** failed deliveries are retried with growing delays. The webhook list shows the last delivery result.
- **Receivers:** the address must be public HTTPS with a valid certificate. Self-signed certificates and private addresses are refused.
- **Change:** you can change the event types or pause a webhook. To change the address, delete it and add a new one. It gets a new secret.
- **Rotate the secret:** **Rotate secret** shows a new secret once. The old one stops at once, so update your receiver right away.

- **App firewall alerts** reach webhooks set to `all`. The "program spreading" alert is a `security` event.

#### TIP

For a complete record (every audit entry, every app firewall decision) send it to your SIEM or log platform instead. See [SIEM and log export: HTTP destination](#).

## Provisioning people (SCIM)

To create, update and suspend people from your identity provider, use SCIM, not API tokens. SCIM has its own token, which starts with `crwscim_`. Set it up with:

- [Microsoft Entra ID: SCIM provisioning](#)
- [Okta and other providers: SCIM provisioning](#)

## Examples

Replace everything in `<angle brackets>` with your own values. The examples use `curl` on Linux or macOS. In Windows PowerShell, use `curl.exe` and put the JSON body in a file ( `-d @body.json` ).

### List people

```
curl -s "https://orchestrator.cloudraw.com/v0/tenants/<workspace-id>/people?q=<name-or-email>" \
-H "Authorization: Bearer <api-token>"
```

Leave out `?q=` to list everyone. Add `?group=<group-id>` for one group's members. Each person has `id`, `email`, `name`, `groups`, `state` (for example `active`, `invited` or `suspended`) and `device_count`. Read-only tokens can do this.

### Publish an app

This publishes a remote desktop server through one of your connectors. Find the connector ID with `GET .../connectors`.

```
curl -s -X POST "https://orchestrator.cloudraw.com/v0/tenants/<workspace-id>/connections" \
-H "Authorization: Bearer <api-token>" \
-H "Content-Type: application/json" \
-H "Idempotency-Key: <new-random-uuid>" \
-d '{"name": "finance-rdp", "type": "rdp", "host": "<server-name-or-ip>", "port": 3389, "connector_id": "<connector-id>"}'
```

The answer ( `201` ) is the new app. Keep its `id` for the next example. Nobody can reach the app until an access rule allows it.

### Add an access rule

This lets one group open the app, only from a computer with BitLocker on. Find group IDs with `GET .../groups`.

```
curl -s -X POST "https://orchestrator.cloudraw.com/v0/tenants/<workspace-id>/policy-rules" \
-H "Authorization: Bearer <api-token>" \
-H "Content-Type: application/json" \
-H "Idempotency-Key: <new-random-uuid>" \
-d '{
  "name": "Finance can use finance-rdp",
  "effect": "allow",
  "target": { "service_id": "<app-id>" },
  "match": { "op": "and", "children": [
    { "type": "group", "operator": "in", "value": ["<group-id>"] },
    { "type": "posture", "field": "disk_encrypted", "operator": "eq", "value": true
  ] }
}'
```

Check `enforced` in the answer. It must be `true`. If it is `false`, `enforcement_reason` says why, and the rule grants nothing. For the health checks you can use, see [Require healthy devices](#).

**TIP**

To see the exact conditions and operators your workspace supports, call `GET .../policy-rule-catalog`.

# Troubleshooting

| Status and code               | Cause and fix                                                                                         |
|-------------------------------|-------------------------------------------------------------------------------------------------------|
| 401 unauthorized              | No <code>Authorization</code> header, or it is not in the form <code>Bearer crw_...</code> .          |
| 401 invalid-token             | The token is wrong or revoked. Create a new one.                                                      |
| 403 insufficient-scope        | A read-only token tried to change something. Use a full token for this script.                        |
| 403 tenant-assertion-mismatch | The workspace ID in the path is not the token's workspace.                                            |
| 404 tenant-not-found          | The workspace ID in the path is wrong.                                                                |
| 409 idempotency-key-reuse     | You sent an <code>Idempotency-Key</code> again with a different body. Use a new key for a new action. |
|                               |                                                                                                       |

| Status and code                             | Cause and fix                                                                                                                                                   |
|---------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 409 version-conflict                        | The object changed since you read it. Read it again, then send the new <code>version</code> in <code>If-Match</code> .                                          |
| 428 if-match-required                       | Access rules and trust rules need <code>If-Match</code> on <code>PATCH</code> and <code>DELETE</code> .                                                         |
| 400 invalid-policy-rule                     | Something in the rule is wrong. <code>detail</code> names the field, for example <code>match.children[1].value</code> .                                         |
| Rule saved but <code>enforced: false</code> | The rule names no group, person or device, or uses a condition that is not enforced in access rules yet. Read <code>enforcement_reason</code> .                 |
| Webhook test fails                          | Your address must be public HTTPS with a valid certificate. Check the answer the test shows, and that your receiver accepts <code>POST</code> with a JSON body. |