

Watch-Drop Handoff SDK

Server-side Node helper (Node 20+, **zero dependencies**; Node 22 LTS recommended) to receive and fulfill FlipKey.gg watch-drop claims — the publisher-side half of a Twitch watch-drop.

The shape of it. Viewers watch your stream → earn a tier → claim (FlipKey.gg provisions them a wallet) → **FlipKey.gg hands the claim off to you** with one signed webhook, so you grant the in-game item on your side.

Install

Copy `flipkey-watchdrop.js` into your backend (or vendor the `/sdk/watchdrop/` folder). No npm package required.

The flow

```
viewer watches stream → crosses a tier → claims (FlipKey: wallet provisioned = the signup)
|
▼
FlipKey POSTs a SIGNED handoff to your webhook → you grant the item → you ack
```

Minimal integration

```
const express = require("express");
const flipkey = require("./flipkey-watchdrop");

app.post(
  "/flipkey/watchdrop",
  express.raw({ type: "application/json" }),    // ← REQUIRED for signature verify
  flipkey.expressHandler({
    secret: process.env.FLIPKEY_WEBHOOK_SECRET, // from dashboard → Webhooks
    apiKey: process.env.FLIPKEY_API_KEY,        // to ack fulfillment
    onClaim: async (claim) => {
      const player = await myDb.playerByFlipkeyAccount(claim.flipkey_account_id);
      await myGame.grantItem(player, claim.reward_drop_id); // idempotent on claim.redemption_id!
      return `granted:${claim.redemption_id}`;             // opaque receipt
    },
  })
);
```

Then set your webhook URL to `https://yourgame.com/flipkey/watchdrop` on the dashboard **Webhooks** page. That's it. See `examples/handoff-server.js` for a complete runnable server.

What you receive (`claim`)

Viewers claim a **soulbound ticket** on FlipKey.gg; the grant fires when they **redeem** it in their library (burn → this webhook). So the event you act on is `item_ticket.redeem`:

field	meaning
event	"item_ticket.redeem" (current rail; the SDK also still accepts legacy "watchdrop.claim" handoffs)
redemption_id	opaque id — use as your idempotency key
reward_drop_id	the reward to grant
item_name	display name of the item
flipkey_account_id	stable account key — map this to your player
wallet_address	same value (the player's FlipKey.gg wallet on Base)
amount	units redeemed (1)
burn_tx_hash	on-chain burn transaction of the consumed ticket
connected	true — connection is a redeem precondition

FlipKey.gg **never** sends a raw game/player id. You hold the `flipkey_account_id → your player` map (stored once at “Connect <Game>”, or matched via `twitch_user_id`).

Security

- Every handoff carries `X-FlipKey-Signature` = `HMAC-SHA256(rawBody, webhook_secret)` (hex). The SDK verifies it for you — **but only if you give it the raw body**, hence `express.raw({ type: "application/json" })` on the route (not `express.json()`).
- Verify on EVERY request; reject mismatches (the SDK returns 401).
- Treat `onClaim` as **idempotent** — FlipKey.gg retries on any non-2xx response.

Don't assume game ownership

Item tickets redeem without owning the game — that's the acquisition funnel working, not an error. A viewer holding your skin is a buyer-to-be. By the time the handoff reaches you the ticket is already **burned**, so never reject a claim because the account doesn't own the game: respond `200`, park the grant in the account's inventory (idempotently), and let the game pick it up whenever they show up. A non-2xx response just makes FlipKey.gg retry the delivery — the player's ticket stays spent either way.

API

- `verifySignature(rawBody, signatureHeader, secret) → boolean`
- `parseClaim(rawBody, signatureHeader, secret) → {ok, claim} | {ok:false, status, error}` — framework-agnostic verify+parse
- `ackFulfilled({ baseUrl?, apiKey, claimId, receipt? })` — mark *fulfilled* + store your receipt
- `expressHandler({ secret, onClaim, apiKey?, baseUrl?, autoAck? })` — the Express convenience wrapper (auto-acks when `apiKey` is set)

Acking

Returning a receipt from `onClaim` (with `apiKey` set) auto-acks fulfillment, so the dashboard shows the claim as **fulfilled**. To ack manually, call `ackFulfilled({ apiKey, claimId, receipt })`. Acking is optional — responding `200` already confirms delivery — but it gives you a clean delivered-vs-fulfilled view and stores your proof-of-delivery receipt.

License

MIT. Source at </sdk/watchdrop/>.

FLIPKEY.GG

© 2026 FlipMyNFT LLC, d/b/a FlipKey

All rights reserved. All trademarks are property of their
respective owners in the US and other countries.

 [X](#)

 [DISCORD](#)

FLIPKEY	PUBLISHERS	CREATORS	LEGAL	MORE
Marketplace	For Publishers	For Creators	Terms	Arcade
About	Register	Apply	Privacy	Your Library
Contact				