

FlipKey.gg Redeem SDK

Drop-in JS + CSS for publisher redemption pages. Wallet-signed redemption flow with safe-by-default key handling — keys never enter the page DOM.

Licenses vs. items. This SDK is the *license-key* redemption path — the buyer proves ownership on your page and your backend releases a key. *In-game items* (giveaway rewards, watch-drop rewards, purchased items) redeem on FlipKey.gg's own library page instead: the soulbound ticket burns and FlipKey.gg POSTs a signed `item_ticket.redeem` webhook to your backend to grant the item — no key, no widget on your side. That half is covered in the [Watch-Drop Handoff SDK guide](#).

Architectural note. The SDK runs on *your* redemption page, on *your* domain. FlipKey.gg itself never holds, transits, or sees the keys you issue. The widget collects a wallet signature and POSTs it to *your* backend; your backend verifies on-chain ownership, claims a key from your own inventory, and returns either a redirect URL, an entitlement-granted notice, or a short-lived show-key payload. The key never appears in HTML attributes.

Quick start

```
<link rel="stylesheet" href="https://flipkey.gg/sdk/redeem/flipkey-redeem.css">
<div class="fk-redeem"
  data-platform="steam"
  data-token-id="42"
  data-redeem-endpoint="/api/redeem"
  data-publisher="Acme Games"></div>

<!-- Nothing else to load. The widget lazy-imports the FlipKey wallet
connector itself, so buyers can sign in with a passkey (no wallet
extension, no account setup). Add data-chain-id="8453" for a
Base-mainnet drop; the default is Base Sepolia testnet (84532). -->
<script src="https://flipkey.gg/sdk/redeem/flipkey-redeem.js"></script>
<script>FlipKeyRedeem.mountAll();</script>
```

The widget renders a **Sign in & Redeem** button. On click it connects the user's wallet, asks for a signature on a domain-bound message that includes the token ID and a fresh nonce, then POSTs `{ tokenId, wallet, signature, nonce, issuedAt }` to `data-redeem-endpoint`.

Base Account is the default (passkey/email/social, the wallet FlipKey.gg buyers have). If the buyer has a browser-extension wallet, a small "Use a browser wallet (Coinbase, MetaMask, ...)" link appears beneath the button as a fallback.

Set `data-platform` to one of: `steam`, `xbox`, `playstation`, `nintendo`, `epic`, `inhouse` (your own launcher: no store page, no account gate; the key is shown, and a `redeemUrl` in your response adds a Continue button). `data-publisher` is optional and is rendered in the widget heading.

Identity gate (Steam & Epic)

When `data-platform` is `steam` or `epic`, the widget adds an identity-confirmation step *before* the wallet flow. On mount it calls `GET {base}/api/{platform}/me` (same origin as your redeem endpoint). If the buyer isn't signed in, the widget renders a "Sign in with Steam/Epic" screen pointing at `{base}/api/{platform}/auth`; after the OAuth/OpenID round-trip the buyer returns to the page authenticated. The confirmed account id then rides along in the redeem POST as `steamId` / `epicAccountId`.

`xbox`, `playstation`, and `nintendo` have no public OAuth, so they skip the gate entirely — the widget goes straight to wallet connect.

Endpoints your backend must expose for a gated platform (relative to the redeem endpoint's origin):

Endpoint	Method	Returns / does
<code>/api/{platform}/me</code>	GET	Profile (<code>steamId</code> / <code>accountId</code> , display name, avatar) if signed in; <code>401</code> or empty otherwise. Reads a same-origin session cookie — the widget sends <code>credentials: 'include'</code> .
<code>/api/{platform}/auth</code>	GET	Starts the OAuth/OpenID redirect and returns the buyer to the redeem page authenticated.
<code>/api/{platform}/logout</code>	POST	Clears the identity session (powers the "use a different account" link).

Why gate at all? For Steam, keys redeem to whichever account is logged into the browser — confirming first prevents activating on the wrong account. For both platforms, a verified platform identity is your input for refund-eligibility and bad-actor screening.

Backend handler contract

Your `data-redeem-endpoint` receives:

```
{
  "tokenId": "42",
  "wallet": "0xabc...",
  "signature": "0x...",
  "nonce": "deadbeef...",
  "issuedAt": 1730000000000,
  "contract": "0x...",           // present only for multi-contract publishers
  "steamId": "7656119..."      // OR "epicAccountId" – present only on gated platforms
}
```

Your handler must:

1. Verify `signature` was produced by `wallet` over the SDK's message format, using a **smart-wallet-aware** verifier — viem's action-level `verifyMessage` (handles EOA, ERC-1271, and ERC-6492). An ecrecover-only check (`ethers.verifyMessage`) rejects Base Account, the SDK's *default* smart-contract wallet. See [examples/redeem-handler.js](#).
2. Verify `nonce` is fresh and within ~5 minutes of `issuedAt` — replay protection.
3. On gated platforms (Steam/Epic), confirm `steamId` / `epicAccountId` is present and record it — it is the second factor and your input for refund-eligibility and bad-actor screening.

4. Verify on-chain that `wallet` owns `tokenId` in your FlipKey-deployed contract.
5. Atomically claim a key from your inventory (`UPDATE keys SET status='redeemed' WHERE status='unused' LIMIT 1`).
6. Report the activation back to FlipKey.gg, signed with your `webhook_secret` .
7. Set `Cache-Control: no-store` on the response and return one of the three result shapes:

The signed message includes your publisher name. The SDK signs `data-publisher` as part of the message, so your handler must rebuild the message with the *exact* same publisher string or signature verification will fail.

```
{ "result": "redirect", "url": "https://store.steampowered.com/account/registerkey?key=..." }
{ "result": "entitlement_granted" }
{ "result": "show_key", "key": "XXXXX-XXXXX-XXXXX", "redeemUrl": "https://...", "expiresInSec": 90 }
```

Use `redirect` for platforms that accept pre-fill (Steam, PSN). Use `entitlement_granted` if you used a direct platform API to grant the game without exposing a key (Steamworks `GrantPackage`, etc.) — preferred when available. Use `show_key` only for platforms that don't support pre-fill (Xbox, Nintendo, Epic).

Since v0.5.0 the widget does **not** auto-navigate on a `redirect` result — it shows the buyer a button to open the pre-filled URL. This is deliberate: auto-redirecting mid-flow could trigger a platform re-login that swallowed the key before the buyer was ready.

Link-test handler

Before FlipKey.gg lets you list a game, every redemption URL on the listing must pass a live link test. FlipKey.gg calls your URL with `?fk_test=1&nonce=<random>` and expects:

```
{ "nonce": "<echoed>", "sig": "<hex hmac_sha256(your_webhook_secret, 'fk_link_test:' + nonce)>" }
```

This proves the URL exists, runs your handler, and that you control the FlipKey-issued `webhook_secret` . A drop-in implementation lives in [examples/link-test-handler.js](#).

Per-platform notes

Platform	Delivery	Code format	Notes
<code>steam</code>	Redirect (pre-fill)	5x5 with hyphens	Cleanest UX; <code>?key=</code> attaches the code. Direct entitlement also possible via Steamworks <code>GrantPackage</code>
<code>xbox</code>	Show-key (paste)	25-char with hyphens	Microsoft's <code>redeem.microsoft.com</code> doesn't accept pre-fill
<code>playstation</code>	Redirect (pre-fill)	12-digit (often 4-4-4)	Sony's <code>?voucherCode=</code> opens the redemption modal pre-filled
<code>nintendo</code>	Show-key (paste)	16-digit	Account must have accessed Switch eShop at least once
<code>epic</code>	Show-key (paste)	varies	Epic's pre-fill OAuth is a separate gated integration

Security model

The SDK enforces these properties; your backend handler is responsible for the rest.

What the SDK guarantees

- The key is **never** written to a `data-*` attribute, `value` attribute, `aria-label`, or any other HTML attribute. Browser extensions, third-party scripts (analytics, ads, support widgets, session-replay), and corporate proxies that snapshot DOM state cannot scrape what isn't there.
- The key is **never** written to `localStorage`, `sessionStorage`, or cookies.
- The key, when shown (paste-flow platforms only), lives in a single closure variable, set into the DOM with `.textContent`, and is auto-redacted after 5 minutes (extended from 90s so buyers have time for Steam Guard / a mid-flow re-login), on tab-hidden, or on navigate-away.
- The key never appears in a URL or query string. The redirect URL the SDK navigates to is built by *your* backend and returned in a JSON response.

What you (the publisher) are responsible for

- **Don't render keys server-side into HTML.** Even with HTTPS, server-rendered keys land in the DOM, get scraped by extensions and session-replay tools, and leak into logging / monitoring services that snapshot the page.
- **Set `Cache-Control: no-store` on every response that may contain key material** — both your `/api/redeem` JSON response and the redemption page HTML. CDN caches, corporate proxies, and browser caches will otherwise persist key material.
- **Prefer direct entitlement over key delivery.** Steam, PSN, Xbox, and Nintendo all support backend-to-backend entitlement grants for games owned/published by you. When you use these, no key string ever exists in the wild. Return `{ result: "entitlement_granted" }` and your buyer never sees a code at all.
- **Atomic key claiming.** `UPDATE keys SET status='redeemed' WHERE status='unused' LIMIT 1` in a transaction. Two concurrent redemption requests must never get the same key.
- **Replay protection.** Track used nonces (Redis with TTL, or a `nonces` table). A signature is reusable forever otherwise.
- **HTTPS everywhere.** Both your redemption page and your `data-redeem-endpoint`.
- **Don't log the key.** Strip the `key` field from request/response logging in your APM, and from any error reporters (Sentry, Bugsnag, etc.). These services often capture full responses by default.
- **Rate-limit by wallet + IP.** A signature flood is not a real redemption.

Programmatic API

```
FlipKeyRedeem.mount(document.querySelector('#my-redeem-widget'));
FlipKeyRedeem.mountAll('.fk-redeem'); // selector optional, defaults to '.fk-redeem'
FlipKeyRedeem.PLATFORMS;              // platform metadata
FlipKeyRedeem.version;                // current SDK version
```

Reference templates

Per-platform reference pages using the SDK against a stub demo backend (returns synthetic `DEMO-...` keys, useful for verifying the flow):

- [Steam template](#)
- [Xbox template](#)
- [PlayStation template](#)
- [Nintendo Switch template](#)
- [Epic Games template](#)

Versioning

Current: `0.8.0`. The SDK is pre-1.0 — APIs may change. Pin the version in your script tag once 1.0 ships.

Compatibility — what an update means for your backend

The widget is served from one URL, `https://flipkey.gg/sdk/redeem/flipkey-redeem.js`, and updates in place; every redemption page, including the Publisher Starter kit's, runs the current build the next time a buyer loads it. The POST your handler receives carries no version field, so compatibility is kept on the message itself: when the signed message changed shape (the station-bound ten-line form replaced the earlier seven-line one), the kit kept verifying both, and FlipKey.gg's custodial signer accepts either form. Changes to the handler contract are listed in the changelog below; a change that would break an existing handler will be announced here before the widget ships it. If you wrote your own handler, the one thing to re-check after any changelog entry is the message your backend rebuilds — it must match the widget's byte for byte.

Changelog

- `0.8.0` — **Custodial buyers redeem on your page.** FlipKey.gg buyers who signed up with passkey/email (no wallet extension, keys custodied by FlipKey.gg) arrive from their library with a one-time grant; the SDK detects it and produces the wallet signature via FlipKey.gg's vault-sign — same POST to your backend, nothing for you to change.
- `0.7.0` — **One-popup sign-in.** Connect + Sign-In-with-Ethereum now happen in a single Base popup (`wallet_connect` with the `signInWithEthereum` capability), replacing the separate connect → `personal_sign` step, which could hang when third-party cookies were blocked.
- `0.6.0` — Wallet layer migrated to **Base Account** (lazy-imported; passkey/email/social, nothing for the publisher to bundle). Injected EIP-1193 wallet on Base is the fallback. New

`data-chain-id` (84532 Base Sepolia default, 8453 Base mainnet). Replaces the prior Coinbase Smart Wallet / Base-chain default.

- `0.5.0` — Per-platform **identity gate** for Steam & Epic (OAuth/OpenID sign-in before key release; the platform account id is sent in the redeem POST). Widget no longer auto-redirects — even redirect-platform results render a button the buyer clicks (avoids losing the key to a mid-flow re-login). Key visibility window extended to 5 min. Redeem POST now carries `issuedAt`, an optional `contract`, and `steamId` / `epicAccountId` when gated. (0.3–0.4 were internal iterations of the gate refactor.)
- `0.2.1` — Coinbase Smart Wallet preferred when `coinbase-sdk.bundle.js` is loaded; "Use a different wallet" toggle for injected fallback.
- `0.2.0` — Safe redemption flow: keys never enter DOM; wallet-signed POST to publisher backend; `show_key` auto-redacts. Removed `data-key`.

License

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

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.



FLIPKEY

Marketplace

About

Contact

PUBLISHERS

For Publishers

Register

CREATORS

For Creators

Apply

LEGAL

Terms

Privacy

MORE

Arcade

Your Library