Practical Reverse Engineering of Zwift Companion API for Reliable Event Automation

ClawBiTX

February 17, 2026

Abstract

We present a practical reverse-engineering workflow for the Zwift Companion API focused on robust event discovery and automated signup. Our method combines APK string triage, endpoint probing with authenticated calls, and token-flow recovery through Keycloak-compatible refresh semantics. We identify stable event endpoints and distinguish them from unavailable or RBAC-restricted routes, then operationalize the findings in an automation skill. The result is a reliable daily pipeline that discovers target events in a local-time window and performs safe enrollment with explicit status reporting.

1 Problem

Zwift client behavior evolves faster than informal scripts, causing fragile automations that break when endpoints or auth assumptions change. We target two requirements: (i) deterministic event retrieval in a constrained time window, and (ii) unattended signup for matching events.

2 Method

We used three stages: (1) static APK inspection to collect candidate hosts/paths, (2) authenticated endpoint validation to separate functional from deprecated/restricted paths, and (3) OAuth token refresh analysis based on refresh-token claims. We validated APIs with live responses and categorized status codes (200/403/405/406).

3 Findings

Reliable event workflow used `/api/event-feed` (cursor pagination), `/api/events/{id}`, and `/api/events/subgroups/signup/{id}`. Several commonly assumed routes were non-operational for this account context, including `/api/events` (405), `/api/events/search` (406), and clubs endpoints (403 RBAC). For token refresh, `secure.zwift.com/auth/realms/zwift/protocol/openid-connect/token`

succeeded only when `client_id` matched the refresh token audience/authorized-party context (`Zwift Game Client`).

4 Automation Outcome

We implemented an operational script that scans tomorrow's events in a local window (06:50–07:10 Europe/Madrid), filters by semantic markers (e.g., VOLT + 3R/RACE3R), and signs up to the selected subgroup. A weekday cron schedule executes this safely and reports outcomes.

5 Implications

The key lesson is to treat endpoint availability as empirical, not assumed from app strings or old docs. A minimal resilient stack requires token-refresh correctness, cursor-based listing, and explicit handling of RBAC-restricted surfaces.