FIELD NOTES

Slack Field Guide

Slack answers almost everything with HTTP 200 and puts the failure in the body as ok: false, so a bot that never posted looks like one that did. Every script here is read only.

28 fixes, each one a written note with diagrams and a small Python and Node.js script that finds the problem and repairs it. All 28 have a script in the public repo.

Before you run anything. Every script starts in a dry run: it reports what it would change and writes nothing. Read that output first, then set `DRY_RUN=false` when you are satisfied it has understood your data.

Scripts: github.com/allanninal/slack-fixes
Guides: allanninal.dev/slack
All 14 repos: github.com/allanninal

The fixes

1 **account_inactive: the installer left and took the token**

The nightly export ran for two years and stopped on a Tuesday. Nothing was deployed, no scope changed, the app is still listed in the workspace. The error is {"ok": false, "error": "account_inactive"}, and the explanation is in the HR system rather than yours: the engineer who installed the app in 2024 left the company on Monday, and SSO deprovisioning deactivated her account overnight.

`slack_installer_account_watch.py`
https://www.allanninal.dev/slack/account-inactive/
https://github.com/allanninal/slack-fixes/tree/main/account-inactive

---

2 **Socket Mode never connects: the xapp- token is underscoped**

The process starts, Bolt announces that it is connecting, and then it retries. Forever. No events arrive, nothing crashes, and the only line in the log that matters is missing_scope from apps.connections.open — a method your read-only audit script is not allowed to call, because opening a connection is a write.

`slack_socket_readiness.py`
https://www.allanninal.dev/slack/app-level-token-missing-connections-write/
https://github.com/allanninal/slack-fixes/tree/main/app-level-token-missing-connections-write

---

3 **is_archived: the target channel was archived months ago**

Nobody filed a ticket, because nothing looks broken. The channel is still there if you search for it, the ID in the config still resolves, and conversations.info still answers ok: true. What changed is one boolean inside that response, set during a reorganisation in March, and every alert since has been refused on the way in.

`slack_archived_target_sweep.py`

https://www.allanninal.dev/slack/archived-channel-target/

https://github.com/allanninal/slack-fixes/tree/main/archived-channel-target

---

4 **One event, many installs: authorizations:read fans it out**

Support says the bot only works for some customers. Your logs disagree: every event was received, handled, and acknowledged, with no errors on any of them. Both are true. Slack delivered one event that several installations should have seen, your handler processed it for the one workspace named in the payload, and nothing anywhere recorded the ones it skipped.

`slack_event_fanout_audit.py`

https://www.allanninal.dev/slack/authorizations-read-missing/

https://github.com/allanninal/slack-fixes/tree/main/authorizations-read-missing

---

5 **the bot answers its own messages in an endless loop**

A channel fills with hundreds of identical bot messages in a few seconds. Slack starts rate-limiting the app, which slows the flood without stopping it, and in the end somebody removes the bot from the channel to make it stop. The cause is one line that was never written: the handler subscribed to message.channels, which delivers every message in the channel, including the one the app posted a moment ago.

`slack_echo_loop_audit.py`

https://www.allanninal.dev/slack/bot-message-echo-loop/

https://github.com/allanninal/slack-fixes/tree/main/bot-message-echo-loop

---

6 **not_in_channel: the bot was never invited to the channel**

The app is installed. The token authenticates. The channel ID was copied out of the URL and is correct. Every call still comes back {"ok": false, "error": "not_in_channel"}, because installing an app to a workspace does not put it in a single channel — and this is, by view count, the most-asked Slack API question there is.

`slack_channel_membership.py`

https://www.allanninal.dev/slack/bot-not-in-channel/

https://github.com/allanninal/slack-fixes/tree/main/bot-not-in-channel

---

7 **the scope was granted to the user token, not the bot**

You added the scope. The consent screen listed it, an admin approved it, you reinstalled, you replaced the stored token, and the call still returns missing_scope with that exact scope in needed. Open the app configuration and there it is, plainly granted — under User Token Scopes, while every line of your code authenticates with the xoxb- bot token.

`slack_token_identity_split.py`

https://www.allanninal.dev/slack/bot-vs-user-scope-mixup/

https://github.com/allanninal/slack-fixes/tree/main/bot-vs-user-scope-mixup

---

8 **channel_not_found: a channel name where an ID belongs**

{"ok": false, "error": "channel_not_found"} for a channel that is open on the other monitor. The name in the config is spelled right, the bot is in the room, the token is fine. And the daily digest that posts to that same string has been working for a year. The difference is not the channel. It is which method you handed the string to.

`slack_channel_reference_form.py`

https://www.allanninal.dev/slack/channel-name-instead-of-id/

https://github.com/allanninal/slack-fixes/tree/main/channel-name-instead-of-id

---

9 **channel_not_found after a rename, or worse, the wrong room**

An integration that has run untouched for two years stops on a Tuesday. Nothing was deployed, no token expired, no scope changed. What happened is that a team tidied up and renamed #ops-alerts to #platform-alerts. If you are lucky, you get channel_not_found and a page. If you are not, somebody created a new #ops-alerts the following week and your alerts are still being delivered, to strangers.

`slack_channel_name_drift.py`

https://www.allanninal.dev/slack/channel-renamed-hardcoded/

https://github.com/allanninal/slack-fixes/tree/main/channel-renamed-hardcoded

---

10 **Classic Slack app: the scope list says bot, client, read**

You go to add one scope. The OAuth & Permissions page does not offer the scope picker you have seen in every tutorial, the code references chat:write:bot which the documentation says does not exist, and the header on every API response reads bot,client,identify,post,read. Nothing is broken yet. What is broken is the assumption that this app can be fixed by adding something to it.

`slack_classic_scope_audit.py`

https://www.allanninal.dev/slack/classic-app-coarse-scopes/

https://github.com/allanninal/slack-fixes/tree/main/classic-app-coarse-scopes

---

11 **The Slack app configuration token died twelve hours ago**

The manifest step in CI passed yesterday and fails today. Someone regenerates the token by hand, the build goes green, and the same thing happens tomorrow morning. Meanwhile the audit that runs beside it reports every manifest-derived check as clean, because a check that could not run returned nothing to complain about.

`slack_config_token_state.py`

https://www.allanninal.dev/slack/config-token-expired/

https://github.com/allanninal/slack-fixes/tree/main/config-token-expired

---

12 **the same message posted three times, and the ts says why**

The alert channel has the same incident in it four times. Every one of those messages is a real, successful chat.postMessage call that returned ok: true and a distinct ts, so nothing failed and nothing will appear in your error tracker. The duplicates are not a display bug — they are four separate decisions your system made to send, and the record of all four is sitting in conversations.history waiting to be read.

`slack_duplicate_messages.py`

https://www.allanninal.dev/slack/duplicate-messages-no-dedupe/

https://github.com/allanninal/slack-fixes/tree/main/duplicate-messages-no-dedupe

---

13 **installs keyed on team_id alone collide on Enterprise Grid**

Two customers file the same ticket in the same week: messages from their Slack app are arriving in a channel that belongs to somebody else. Both are on the same Enterprise Grid organisation. Your installation store has one row per team_id, it has always had one row per team_id, and on a single-workspace customer that was correct. On Grid it is a cross-tenant data leak with a green dashboard.

`slack_install_key_audit.py`

https://www.allanninal.dev/slack/enterprise-id-not-stored/

https://github.com/allanninal/slack-fixes/tree/main/enterprise-id-not-stored

---

14 **Slack disabled event delivery and will not turn it back on**

There was a two-hour outage on Tuesday. The service came back, the health checks went green, the on-call went to bed. On Thursday somebody asks why the bot has not answered anyone since Tuesday. Slack turned event delivery off during the outage, emailed the app owner about it, and does not turn it back on when you recover — a human has to click a button that nobody knows exists.

`slack_event_silence_audit.py`

https://www.allanninal.dev/slack/event-subscriptions-auto-disabled/

https://github.com/allanninal/slack-fixes/tree/main/event-subscriptions-auto-disabled

---

15 **files.upload is retired: one probe returns method_deprecated**

Every internal tool that posts a screenshot into Slack stopped posting screenshots, all of them on the same day, none of them deployed that week. The logs say 200. The bodies say {"ok": false, "error": "method_deprecated"}. Nothing broke: files.upload reached the end of a sunset that was announced eighteen months earlier and moved once.

`slack_files_upload_probe.py`

https://www.allanninal.dev/slack/files-upload-retired/

https://github.com/allanninal/slack-fixes/tree/main/files-upload-retired

---

16 **slack answers HTTP 200 and puts the failure in the body**

The deploy is green. The log line reads POST https://slack.com/api/chat.postMessage 200. Nothing has appeared in the channel for three weeks. When somebody finally logs the response body it reads {"ok": false, "error": "not_in_channel"} — and it has read that, unchanged, every single time.

`slack_ok_false_audit.py`

https://www.allanninal.dev/slack/http-200-ok-false/

https://github.com/allanninal/slack-fixes/tree/main/http-200-ok-false

---

17 **invalid_auth: the xapp- token is in the Web API slot**

{"ok": false, "error": "invalid_auth"} on a call that plainly should work, with a token copied out of the app configuration ten minutes ago. The token is not expired, not revoked, and not missing a scope. It starts with xapp-, and the Web API has never accepted one of those.

`slack_token_class_check.py`

https://www.allanninal.dev/slack/invalid-auth-wrong-token-type/

https://github.com/allanninal/slack-fixes/tree/main/invalid-auth-wrong-token-type

---

18 **missing_scope tells you the scope needed and the ones you have**

{"ok": false, "error": "missing_scope", "needed": "channels:history", "provided": "chat:write,commands,users:read"}. The developer swears the scope is in the app configuration, and it is — but the app was never reinstalled, so the token in production still carries the grant it was issued with.

`slack_scope_audit.py`

https://www.allanninal.dev/slack/missing-scope-on-read/

https://github.com/allanninal/slack-fixes/tree/main/missing-scope-on-read

---

19 **conversations.history clamped to 15 objects and 1 per minute**

A backfill that used to take an hour now takes weeks, and nothing in your code changed. conversations.history still returns ok: true, still returns a valid page, still gives you a cursor — it just returns 15 messages when you asked for 200, and refuses the second call inside the same minute. This is Slack's May 2025 rate-limit change for apps that are not approved for the Marketplace, and it is working exactly as designed.

`slack_history_clamp_probe.py`

https://www.allanninal.dev/slack/non-marketplace-history-clamp/

https://github.com/allanninal/slack-fixes/tree/main/non-marketplace-history-clamp

---

20 **not_allowed_token_type: right secret, wrong token class**

{"ok": false, "error": "not_allowed_token_type"}. The token works. It authenticates, it has scopes, it calls a dozen other methods happily. This one method, and only this one, will not take it — and the error names the problem without naming the solution, because it says what is wrong with the class you brought and nothing about which class it wanted.

`slack_method_token_class.py`

https://www.allanninal.dev/slack/not-allowed-token-type/

https://github.com/allanninal/slack-fixes/tree/main/not-allowed-token-type

---

21 **the token holds admin scopes the app has never called**

The security review asks a reasonable question: why does the nightly digest bot hold admin.users:write, files:write, chat:write.public and users:read.email? Nobody knows. Nothing is broken, no error has ever been logged, and every scope on that list was added by somebody who needed it for ten minutes in 2023.

`slack_scope_surplus.py`

https://www.allanninal.dev/slack/over-broad-scopes/

https://github.com/allanninal/slack-fixes/tree/main/over-broad-scopes

---

22 **next_cursor is ignored so only the first page is ever seen**

The channel inventory has exactly 100 entries. So does the user directory. Nobody questions either number until somebody reports that a channel which plainly exists is missing from the report — and by then the sync has been dropping four fifths of the workspace every night for a year, with ok: true on every single response.

`slack_pagination_audit.py`

https://www.allanninal.dev/slack/pagination-not-followed/

https://github.com/allanninal/slack-fixes/tree/main/pagination-not-followed

---

23 **channel_not_found: private, and the token cannot see it**

You are looking at the channel. It is on the screen, the ID is copied from its details panel, other people are talking in it. conversations.info says channel_not_found, and conversations.list returns four hundred channels without it. Nothing is wrong with the ID. The token has not been told that private channels exist.

`slack_private_visibility_probe.py`

https://www.allanninal.dev/slack/private-channel-invisible/

https://github.com/allanninal/slack-fixes/tree/main/private-channel-invisible

---

24 **files made public with a link that works without a Slack login**

Nothing errored, and nothing is going to. Somewhere in the app's history a developer needed an image URL that Block Kit could actually fetch, called files.sharedPublicURL, and it worked. Every file that call has been made against since — customer exports, database dumps, screenshots with a token still on screen — is readable by anyone holding the link, with no Slack account, no workspace membership and no expiry. The flag is on each file, and files.list will hand you all of them.

`slack_public_files_audit.py`

https://www.allanninal.dev/slack/public-file-links-exposed/

https://github.com/allanninal/slack-fixes/tree/main/public-file-links-exposed

---

25 **Slack refresh tokens are single use: a replay kills the pair**

Rotation was working. Then one afternoon every call returns token_expired, the refresh call answers invalid_refresh_token, and the only recovery is to send a customer through the install flow again. Nothing was deployed. What changed is that the service went from one replica to two, and both of them woke up on the same cron minute.

`slack_refresh_ledger_audit.py`

https://www.allanninal.dev/slack/refresh-token-reused/

https://github.com/allanninal/slack-fixes/tree/main/refresh-token-reused

---

26 **token_expired every 12 hours because rotation is on**

The app is perfect for half a day after every deploy and then every call returns {"ok": false, "error": "token_expired"}. A nightly redeploy hid it for four months, because each deploy ran the install flow again and each install handed back a fresh token. Then the redeploy was removed as an optimisation, and the app started dying every lunchtime.

`slack_rotation_clock.py`

https://www.allanninal.dev/slack/token-expired-rotation/

https://github.com/allanninal/slack-fixes/tree/main/token-expired-rotation

---

27 **token_revoked: the app is gone and retrying will not help**

One tenant of your multi-workspace app has received nothing for six weeks. No alert fired. The installation row is still there, still marked active, still being handed to the scheduler every fifteen minutes, and every call it makes returns {"ok": false, "error": "token_revoked"}. Somebody removed the app from Manage apps in January and your store has been retrying ever since.

`slack_dead_install_sweep.py`

https://www.allanninal.dev/slack/token-revoked/

https://github.com/allanninal/slack-fixes/tree/main/token-revoked

---

28 **every Slack profile has a null email and nothing errored**

The nightly user sync has been green for four months. It reads every member out of Slack, writes them to the warehouse, and joins them against the HR system on email. The join has matched nothing since the day it shipped, because every row it wrote has email = null — and users.list returned ok: true, with complete-looking profiles, every single night.

`slack_email_scope_audit.py`

https://www.allanninal.dev/slack/users-read-email-missing/

https://github.com/allanninal/slack-fixes/tree/main/users-read-email-missing

---