Diagnose Palworld mod update failures and client mismatches

After a Palworld or mod update, separate a server that cannot start from a running server that rejects a client. Those failures need different evidence. Preserve the last working world and package list before repeated starts make the original sequence harder to reconstruct.

Record exactly what changed

1. Write down the last successful connection time and the first failed attempt, including the time zone.
2. Compare the game build, operating system, mod versions and loader version before and after the update. Include dependencies, not just the main gameplay mod.
3. Save startup output from the beginning through the first error. Keep client errors separately and remove credentials before sharing logs.
4. Keep a copy of the current failed state as well as the last known working checkpoint. Do not overwrite your only recovery copy while testing.

If the service was recently moved between Windows and Linux, treat that platform change as a compatibility question. The official server-mod workflow remains Windows-only in Pocketpair's current documentation. File transfer success does not prove a Windows extension can execute on a native Linux server.

Follow the failure stage

No normal server startup: inspect the earliest loader, dependency or package error. Generic connection timeouts are expected when the server has not reached a usable state. Adding RAM or changing firewall rules will not repair a missing module.

Server starts, mod does nothing: compare the intended package with the manifest and enabled list. Pocketpair documents a generated

`Mods/ManagedMods/<PackageName>/InstallManifest.json` after deployment. Its presence is evidence of deployment activity, not proof the feature works correctly.

Only some clients fail: list each client's game build, platform and installed package versions. Ask the mod author which components belong on clients. Do not assume console, Steam and other clients share the same mod installation capabilities.

Check updates without mixing installations

For managed Workshop mods, Pocketpair explains that a changed manifest `Version` triggers redeployment on restart. Confirm which release you actually downloaded and keep the previous package archived. Do not combine unrelated manual DLL instructions with the managed layout to silence an error.

The upstream UE4SS project and each mod's own release notes are the appropriate references for loader compatibility. Experimental ports are separate projects; a fork reporting success does not establish support for the upstream runtime or your hosting template.

Reproduce safely

Use a separate disposable world and add the required dependency chain before the affected mod. Connect with one prepared client, exercise the failing action, then perform a clean stop and start. Add other packages individually until the smallest failing combination is identified. Keep the log for both the passing and failing attempts.

Do not load an established modded world without its content dependencies just to see whether startup improves. That can change the saved state and makes a clean comparison unreliable. A mod-free baseline belongs in a fresh test world.

Rollback or escalate with evidence

Restore a matching game, loader, package and save checkpoint only where those versions remain obtainable and compatible. If an update migrated the save, stop and ask for recovery guidance before downgrading. Send support the server ID, build list, first error and minimal reproduction; never include access tokens or passwords.

Related: Update Palworld with a recoverable checkpoint and game-file backups.

Revision #1
Created 2026-11-10 11:05:47 UTC by CloudExa Hosting Admin
Updated 2026-11-10 11:05:47 UTC by CloudExa Hosting Admin