Find a client-only Fabric mod in a dedicated-server crash

A Fabric pack can work in singleplayer and fail on a dedicated server because one mod references client-only classes. A dedicated server has no rendering client. Diagnose that side mismatch before removing world content or treating every class-loading error as a missing library.

Identify the first client-side reference

Save `logs/latest.log`, console startup output and the matching file in `crash-reports/` when one exists. Record the Minecraft, Loader and mod versions. Look for the first relevant cause and its mod or mixin name. References under `net.minecraft.client` can indicate client code, but a missing class can have other causes, including a wrong Minecraft build. Treat the name as evidence to investigate.

Do not diagnose from the final exit line alone. A later failure may follow the original loading error. Keep the full private report, then redact unrelated player information, tokens and external-service credentials before sharing an excerpt.

Check the implicated mod's environment

1. Find the exact JAR named or associated with the first error. Download a copy and open it as an archive on your computer.
2. Read its root `fabric.mod.json`. The current Fabric specification uses `environment` values `client`, `server` or `*`.
3. A mod declaring `client` should not load on a dedicated server. The default `*` declares availability in both environments; it does not prove the author's implementation contains no side-related bug.

4. Inspect the documented entrypoint and mixin environment declarations as evidence. Compare them with the author's server-support statement and the release matching your Minecraft version.

Singleplayer includes both a physical client and an integrated server. A mod that loads there can still fail when the same code runs without client classes. A crash mentioning a rendering class is therefore worth comparing against dedicated-server support, even if the owner reports that the pack works on their laptop.

Test a narrow removal on a copy

Use a backed-up, isolated test server. If the author confirms that the implicated mod is client-only, move that one JAR outside the test server's `mods/` directory. Keep it installed on clients when the pack requires its client feature. Do not copy the whole desktop mods folder onto the server as a correction.

If the mod adds world blocks, dimensions or entities, stop and verify its supported removal procedure before opening a save without it. A mod with both client and server content needs a compatible release or an author fix, rather than a blind deletion. Do not edit its metadata to make the loader ignore a side declaration.

Verify the dedicated-server path

Start the test server and confirm that the original class-loading error disappears. Join with the matching client pack and check the feature involved. Preserve a short before-and-after record. If a new missing-dependency error appears, review the removed mod's dependants instead of repeating random removals.

To undo the trial, restore the original JAR set and configuration. Restore the complete test-world checkpoint if a content change already altered it. Report a genuine side-loading bug with a minimal reproduction and precise versions; an entire private server archive is unnecessary.

Related help

Diagnose Minecraft mod conflicts on a test copy · Find Minecraft logs and share useful evidence · Back up game files before making changes

Official references

- Fabric environments, entrypoints and mixins
- Fabric crash-report interpretation
- Fabric Loader behaviour

Revision #1
Created 2026-11-09 10:55:44 UTC by CloudExa Hosting Admin
Updated 2026-11-09 10:55:44 UTC by CloudExa Hosting Admin