Find the active NeoForge server config after a mod update

A NeoForge setting can appear unchanged after an update because you edited a client-only file, a shared value or a file overridden by the current world. Identify the mod's configuration system and active file before deleting generated configuration or resetting the world.

Record one setting and its expected effect

Write down the mod version, Minecraft and NeoForge versions, the setting name and the behaviour you expect. Save startup warnings around configuration loading. Keep private copies of the original files and the complete pre-update server backup. An update may rename a key or change its accepted type, so retain the mod author's release notes.

This guide covers mods using NeoForge's documented configuration registration. Mods can use their own filenames, formats and locations. A file ending in `.toml` does not prove that its author uses the standard system or default suffixes.

Identify the configuration type

1. Read the mod's configuration documentation for the selected release. Determine whether the setting belongs to a client, common or server configuration.
2. Inspect the actual files under `config/`. Default names often end with `-client.toml`, `-common.toml` or `-server.toml`, but registration can supply custom names.
3. Check the intended world for `serverconfig/` overrides. Derive the world name from your installation and `server.properties`, rather than assuming its directory is literally `world`.
4. Record the relevant value at each applicable location before editing. Keep client launcher files separate from the dedicated server's files.

Use the version's documented precedence

The current 26.1 NeoForge reference describes SERVER configuration loaded from the config folder, with per-world overrides such as `world/serverconfig/` on a dedicated server. It also documents synchronisation of SERVER values to connected clients. Consult the reference for your installed generation; do not treat one current layout as a guarantee for every historical loader or mod.

A local client's common setting is not necessarily the server's rule. Nor does copying a template into a defaults directory establish that an existing world's active configuration changed. Identify the file the mod actually loads, using documentation and startup evidence.

Apply one reversible correction

Stop the test copy cleanly. Edit one documented key in the identified active file, preserving TOML section names and value types. Keep the old value in your notes. Do not copy a complete old configuration over a new schema or delete all files to force regeneration.

Start normally and inspect any correction, migration or unknown-value warning. Reopen the file to see whether the mod retained or rewrote your value. A rewritten value can indicate a validation rule or outdated key; repeating the same edit will not establish a fix. If the mod supports live reload, follow its own instructions rather than assuming every setting reloads.

Check behaviour and scope

Test the intended feature with a matching non-operator client, then check a second world if the change should affect only one. Record the outcome and the exact active path. Startup success proves parsing, not that your setting controls the reported feature.

If the test fails, restore only the changed file or entry and restart. Preserve worlds and player data. Send support the mod version, key, redacted loading warning and both candidate paths. If the update migrated world content as well, recover that test from the complete checkpoint rather than downgrading files against a changed save.

Related help

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

Official references

- Current NeoForge configuration types and locations
- NeoForge version-selected configuration reference
- NeoForge server update guidance

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