Resolve a NeoForge required-mod or version-range failure

NeoForge can refuse a mod set before loading the world when a required dependency is absent or outside its accepted version range. Read the named requirement and the selected release's metadata. Uploading the latest file with the same project name can create another incompatibility.

Keep the first dependency report

Save console output, `logs/latest.log` and the matching crash report if one exists. Record Minecraft, NeoForge and Java versions plus the pack release. List the requesting mod, required mod ID, accepted version range and actual version that the loader found. Keep those fields separate from display names.

NeoForge's numbering follows the Minecraft generation, but its version format has evolved. The official versioning page gives examples such as 20.2 for Minecraft 1.20.2 and a full-version form for 26.1. Choose the intended Minecraft release first; do not compare only the largest number across unrelated branches.

Inspect the published metadata without changing it

1. Download the implicated JAR and open a copy as an archive on your computer. Do not unpack or modify the production installation.
2. For releases using the current documented format, inspect `META-INF/neoforge.mods.toml`. Older NeoForge generations can differ; use their release documentation rather than renaming a metadata file.
3. Find the relevant dependency section. Current documentation describes fields including `modId`, `type`, `versionRange` and `side`. Distinguish a required dependency from an optional or incompatible declaration.

4. Compare that record with the loader's first dependency message and the author's download requirements.

NeoForge uses Maven-style version ranges in relevant metadata. For illustration, `[1.0,2.0)` includes 1.0 and excludes 2.0. This example explains the brackets; it is not a real library recommendation. Read the actual range and the project's version conventions instead of assuming an equal prefix proves compatibility.

Build a compatible set

Select the correct NeoForge download for the existing Minecraft release. A Forge or Fabric build of the same project is not an automatic substitute. If the dependency already exists, check for duplicate or older copies and for other mods that require a conflicting range. Keep removed copies outside `mods/`.

Do not change `type` to optional, widen `versionRange`, or rename a Forge JAR to bypass loading. Those metadata edits cannot supply missing classes or restore an API. A requirement for a newer NeoForge branch may mean you need a compatible older mod build or a separately planned Minecraft upgrade.

Verify before changing the live pack

On an isolated test copy, apply the matched files and confirm that the original dependency check passes. Read any new first error rather than calling the entire pack fixed after one message disappears. Join with a matching client and test the requesting mod's feature.

Return to the saved mod set if verification fails. Preserve the production world throughout. Recover a migrated test save from its complete checkpoint before using older content versions. Provide support with the mod IDs, exact files and range error, with credentials removed; no reinstall is needed to inspect a dependency mismatch.

Related help

Diagnose Minecraft mod conflicts on a test copy · Install or change a Minecraft modpack safely · Back up game files before making changes

Official references

- NeoForge dependency metadata
- NeoForge version numbering and ranges
- NeoForge server distribution and updating 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