Resolve “UnknownDependencyException” when a Paper plugin will not load

`UnknownDependencyException` commonly appears when Paper cannot satisfy a plugin's required dependency. The surrounding message may identify unknown or missing dependency plugins. Installing a random JAR with a similar filename is not enough: the dependency must be the correct platform build, load successfully and meet the dependent plugin's requirements.

Identify the dependent and the dependency

Read the earliest failed loading sequence in `logs/latest.log`. Record which plugin was being loaded and the exact dependency name reported. Keep the full exception for a private support ticket, with unrelated secrets removed. The outer line saying that a plugin could not load is less specific than the dependency information below it.

Check the dependent plugin's official release page. Distinguish required dependencies from optional integrations and from libraries bundled inside the JAR. A project can offer different builds for Paper, Velocity or a mod loader. A similarly named proxy plugin is not automatically the backend dependency required here.

Follow the dependency backward in the log

If the required JAR is absent, obtain the supported version from its author's official source. If it is already present, search earlier startup messages for that dependency's own failure. It may have been rejected because of an unsupported API, Java requirement, corrupt download or another missing dependency of its own.

Resolve that first failure before repeatedly reinstalling the plugin that reported it later. This is especially important for chains: plugin A may require B, while B failed because C was unavailable. Keep the chain in your notes so an attempted fix does not introduce incompatible versions elsewhere.

Check identity and location

Traditional plugins declare an internal name in their metadata; the filename alone does not establish dependency identity. Do not rename an unrelated JAR to the expected name or edit metadata inside an archive to satisfy the loader. Confirm the author and documented project instead.

Ensure the supported dependency JAR is in the active server's plugin directory, not a nested archive or a different service with the same display name. Check for duplicate releases before adding another copy. On modern Paper, Paper-plugin dependency rules can differ from traditional Bukkit declarations, so follow the documentation for the actual plugin format and release.

Repair a matched set

Back up the plugin data and relevant external database, then stop the server cleanly before adjusting JARs. Preserve the previous compatible files outside the active directory. Install the smallest author-supported combination rather than updating every dependency to an arbitrary latest build.

Do not use `/reload` to trial dependency changes. Start normally and review the log from the beginning. Confirm the dependency loads before the dependent plugin and that neither reports a new failure. Run `plugins` in the console where supported, then test the particular feature with an ordinary account. A green plugin name is a useful check, not proof that its database or permissions are correct.

Recover

If the change fails, keep the first new error and restore the known compatible set. Check migration notes before rolling back data or an older JAR. Keep worlds intact and avoid deleting the plugin's

data folder to force a fresh start. Send support the dependency chain, versions, official download links and first error for each failing component.

Related help

- Install plugins on a Paper or Purpur Minecraft server
- Find Minecraft logs and send useful troubleshooting details
- Update Paper, Purpur and plugins safely

Official references

Paper missing-dependency troubleshooting. Plugin metadata dependency names. Paper-plugin loader differences.

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