Allow compatible Minecraft Java clients with ViaVersion

ViaVersion translates Minecraft Java network protocols so supported client versions can join a server running a different Minecraft version. It does not upgrade your world or add every block, item and gameplay feature from a newer release. Keep the server's actual version clear when advertising compatibility to players.

Write down the versions you need

Record the server software and exact Minecraft version, proxy version if present, Java runtime and the client versions players need. Use the project's current compatibility guidance or setup tool for that combination. A broad label such as “latest” is not enough to diagnose a protocol mismatch.

ViaVersion generally serves newer clients joining older server protocols. Older clients joining newer servers require the appropriate related project, commonly ViaBackwards, within its supported range. Very old clients may require ViaRewind as well. Confirm the supported combination rather than installing every add-on.

These are Java protocol tools. They do not replace Geyser for Bedrock access, turn a Java server into a Bedrock server or solve a modpack's missing client mods. Use the relevant crossplay or modpack guide for those requirements.

Choose one translation location

Back up the existing server files and record the currently working client/server combination before installing anything. On a standalone supported Paper server, install the compatible plugin through the ordinary plugin process.

For a network, follow the project's guidance on placing translation at the proxy or backends. Do not assume adding the same translators at every hop improves compatibility. Keep your proxy current and preserve its existing forwarding and backend-access protection.

Check the release's Java requirements against your runtime. A dependency or Java startup failure is different from a player being rejected for an unsupported protocol. Fix the first startup error

before changing the world's version or authentication configuration.

Test before reopening access

1. Start the server and confirm that the translators load without errors.
2. Join with a client matching the server's own Minecraft version. This checks the ordinary route before testing translation.
3. Join with each additional client version you intend to support. Test movement, inventory, chat and important gameplay interactions.
4. If using a proxy, test the public entry address and switching between the intended backends.
5. Record any resource-pack, anti-cheat or plugin differences separately from connection success.

A successful login does not prove every visual or gameplay interaction behaves identically. Newer content can need substitutions on older clients, and anti-cheat or packet-sensitive plugins may have their own compatibility requirements. Test the actual activities your community uses.

Keep failures specific

If one client is rejected, capture its exact version, the server/proxy versions and the first relevant connection error. Compare them with the installed translator releases and current support range. Avoid repeated world upgrades as a fix for an unsupported client.

Do not disable online authentication or expose protected backend ports to bypass a translation error. If you decide to remove the translators, use a maintenance window and the saved working files, then test the original supported client again. Protocol changes should remain separate from world migration and data recovery.

Related CloudExa guides

- Set up a Velocity network with protected backend servers
- Set up Geyser and Floodgate for Java and Bedrock crossplay

Further reading

- ViaVersion installation and compatibility

- ViaVersion project
- ViaBackwards project
- ViaRewind project

Revision #2
Created 2026-11-08 21:45:13 UTC by CloudExa Hosting Admin
Updated 2026-11-08 21:45:13 UTC by CloudExa Hosting Admin