ViaBackwards tasks

Task guides and troubleshooting for ViaBackwards. Check the installed release before changing configuration or data.

- Check old-client access near modern world-height limits
- Test smithing and crafting menus with older Java clients
- Contain a translated-client inventory desynchronisation report
- Separate a missing ViaBackwards dependency from an unsupported client

Check old-client access near modern world-height limits

A modern world can contain playable blocks outside the height range understood by an older client. ViaBackwards documents limits for clients predating expanded world height. Translation cannot give those clients a native view of every modern coordinate, even when joining works.

Record the starting point

Record the server version and oldest permitted client. Identify the coordinates of spawn, portals, homes and minigame arenas. Use a private test area with a safe recovery location. Do not test the limitation by dropping a real player's valuable inventory into an inaccessible region.

Work through the change

1. Read the current ViaBackwards known issues for the client generation you allow. Compare its supported vertical range with the coordinates your community actually uses.
2. Join with the native client and the older client, then inspect a safe route near each relevant boundary. Check movement and interaction, not just whether a chunk appears.
3. Review automatic teleports such as homes, portals and random-teleport destinations. A plugin may select a valid modern coordinate that the older client cannot use properly.
4. Choose a documented client restriction, a compatible destination policy or a clearly explained limitation. Retest a normal login and the affected teleport using a disposable test account.

Check the result

The supported client range should match the areas players are expected to reach. Record the tested coordinates and client builds. If you retain older clients with limits, place the explanation alongside joining instructions so players do not discover it after becoming trapped.

If the result differs

If the native client also fails at the destination, investigate the world, teleport plugin or protection rule separately. If only the old client fails beyond the documented range, repeatedly reinstalling ViaBackwards will not expand the client's world model.

Return the tester to the safe location using normal server controls. Restore only the trial teleport or client policy if it causes a new issue. Keep the modern world intact; deleting regions or downgrading world files is not a safe compatibility fix.

References and related help

- ViaBackwards maintainer reference 1
- Allow compatible Minecraft Java clients with ViaVersion
- Let supported older Java clients join with ViaBackwards

Test smithing and crafting menus with older Java clients

New Minecraft releases can change workstation screens and recipes. ViaBackwards translates many interactions, but its known issues include limitations affecting older clients and modern smithing behaviour. A client that can open the menu has not necessarily completed a valid transaction.

Record the starting point

Prepare disposable copies of the ingredients and record the expected output using the server's native client. Note custom recipe and menu plugins. Keep valuable player items out of the test, and verify the installed ViaBackwards release against its current known-issues list.

Work through the change

1. Run the complete recipe with the native client first. Confirm the server creates the intended output and consumes the expected ingredients without a permission override.
2. Repeat the exact interaction with the older supported client. Check input slots, preview output, shift-click and taking the finished item, using ordinary player permissions.
3. Reconnect before checking the final inventory. A temporary visual item is different from a server-confirmed result; compare both clients without repeatedly clicking an uncertain output.
4. If the interaction is unsupported, publish the limitation or require a compatible client for that feature. Test any custom workaround on a private copy and check it cannot duplicate ingredients.

Check the result

The recipe should either complete with consistent server-side inventory or fail in a documented, understandable way. Keep the test small enough to compare item counts before and after. A workaround should preserve the intended recipe cost rather than granting an unrelated replacement reward.

If the result differs

A failure on both clients suggests a recipe, permission or plugin issue outside protocol translation. A failure only on the older client should be checked against ViaBackwards limitations before changing the global crafting system. Include the exact menu and click sequence in a report.

Remove the disposable ingredients and restore the trial menu configuration if needed. Correct a test inventory through ordinary staff tools after checking server-side counts. Do not restore the entire world because one workstation preview displayed incorrectly.

References and related help

- ViaBackwards maintainer reference 1
- Allow compatible Minecraft Java clients with ViaVersion
- Let supported older Java clients join with ViaBackwards

Contain a translated-client inventory desynchronisation report

An older client's inventory screen can disagree temporarily with server state after an unsupported click sequence. ViaBackwards lists inventory compatibility limits. Treat a report as a consistency problem to investigate, rather than immediately compensating every item shown in a screenshot.

Record the starting point

Ask for the client version, container type and exact action before the mismatch. Prepare low-value test items with counted stacks. Keep an inventory checkpoint for the test account and avoid experimenting in a shared chest containing player valuables.

Work through the change

1. Repeat the action with the native server client. Record the starting and ending stack counts and whether a container-management plugin changes the menu.
2. Use the affected older client with the same items and permissions. Try the reported action once, then close and reopen the inventory instead of repeatedly moving a suspicious stack.
3. Reconnect and compare the server-confirmed inventory. If staff inspect stored data, use established inventory tools rather than editing raw player files while the account is online.
4. Trial the compatible ViaVersion and ViaBackwards release set, or restrict the failing interaction or client version. Check both the reported action and ordinary chest transfers before reopening normal use.

Check the result

A passing result means counts remain consistent after reconnecting and the player can use the supported interaction reliably. If a visual mismatch remains a known limitation, explain the safe supported client choice. Preserve a short reproduction for the plugin maintainers.

If the result differs

If item counts actually change server-side, stop the affected test route and investigate all menu or inventory plugins involved. A screenshot alone cannot establish duplication or loss. Avoid adding compensating items until staff have checked the original and final records.

Restore the test account's checkpoint only if the controlled test altered its real inventory. Revert the isolated compatibility change if it creates a broader failure. Keep unrelated player records and worlds untouched throughout the investigation.

References and related help

- ViaBackwards maintainer reference 1
- Allow compatible Minecraft Java clients with ViaVersion
- Recover a Minecraft player inventory from a backup

Separate a missing ViaBackwards dependency from an unsupported client

ViaBackwards requires ViaVersion. A failed dependency load and an unsupported client are different problems: the first concerns the server's plugin chain, while the second concerns the protocols that chain can translate. Checking them in order avoids changing working authentication or network settings.

Record the starting point

Record all Via release numbers, server software and affected client version. Keep the first startup error, not just the final disconnect. Check whether ViaRewind is also required for the legacy client generation you intend to support.

Work through the change

1. Start with the normal boot log and confirm ViaVersion loaded before ViaBackwards. Resolve missing dependency or incompatible API messages using the maintainers' release requirements.
2. Check the supported protocol direction. ViaBackwards allows supported older clients onto newer servers; it is not a promise that every historic or snapshot build is accepted.
3. Test the native client, one supported older client and the reported failing client. On a proxy network, use the normal public route and record which destination rejects the connection.
4. Change the smallest relevant component: install a compatible dependency set, correct its placement or adjust the published support range. Restart normally and repeat the same three tests.

Check the result

The log should show a healthy dependency chain before older-client acceptance is judged. The native client should continue to work. Keep a record of the oldest tested client rather than treating the plugin name as a complete compatibility policy.

If the result differs

If ViaBackwards loads but a client fails, inspect the rejection text and any proxy-side restriction. If ViaBackwards never loads, debugging gameplay menus is premature. A Java runtime mismatch can also stop a plugin before it has handled any player traffic.

Restore the previous working Via release set if the replacement cannot load. Preserve the error and artifact names for support. Do not remove authentication, delete plugin databases or downgrade world files to work around a missing library.

References and related help

- ViaBackwards maintainer reference 1
- ViaBackwards maintainer reference 2
- ViaBackwards maintainer reference 3
- Allow compatible Minecraft Java clients with ViaVersion
- Find Minecraft logs and send useful troubleshooting details