Vault tasks

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

- Find the economy provider that Vault actually selected
- Check a plugin that says no economy provider is available
- Investigate different balances in a shop and a balance command
- Plan a Vault economy-provider change without losing balances
- Distinguish Vault economy failures from permissions failures
- Prepare a useful report for a Vault integration error

Find the economy provider that Vault actually selected

A shop can report that Vault loaded while still using an unexpected economy backend. Establish which provider answers requests before moving balances, installing another money plugin or changing prices.

Before you start

Vault connects consumer plugins to an economy provider; it does not replace that provider or store your player balances. Identify the component holding the money before changing integrations, and check the provider name in startup logs.

Work through the check

1. Read the complete startup sequence and note Vault, your economy plugin and the affected shop build.
2. Find the economy registration or hook message, then compare its provider name with the money plugin you intended to use.
3. Use the provider’s documented balance command for one test account, recording the exact amount and account identity.
4. Perform one inexpensive shop transaction, then compare the provider balance and the shop message; keep the timestamp for log correlation.

What to compare

Keep the provider name and the player UUID in the same private test record. A player nickname is not reliable evidence that two displays refer to the same underlying economy account.

Confirm the result

The same test account should show the expected debit or credit in the selected provider, with no second provider claiming the transaction.

If the result differs

A missing hook suggests a disabled or incompatible provider; two different balance displays suggest different backends, currencies or cached placeholders. Inspect earlier startup errors before changing the shop.

Recover safely

Leave balances untouched while diagnosing. If you removed a conflicting plugin on a test copy, restore its matching data before comparing another startup; do not merge balances by guesswork.

Documentation and related help

Check the Vault maintainer documentation and Vault maintainer reference against your installed release before applying changes.

Related CloudExa help: Connect Vault to an EssentialsX economy and Find Minecraft logs and send useful troubleshooting details.

Check a plugin that says no economy provider is available

A consumer plugin needs an enabled economy service as well as Vault. Treat the first hook failure as a dependency problem; granting players more permissions will not create a provider.

Before you start

Vault connects consumer plugins to an economy provider; it does not replace that provider or store your player balances. Identify the component holding the money before changing integrations, and check the provider name in startup logs.

Work through the check

1. Locate the first error for the economy plugin during startup, before the shop or reward plugin announces its own failure.
2. Confirm the installed money plugin documents a Vault economy hook for the server platform and release you use.
3. Check that Vault and the provider are installed in the server actually running the consumer, rather than only on a proxy or another backend.
4. Restart the isolated test server with the supported dependency set and repeat the smallest feature that previously failed.

What to compare

Read startup logs from the beginning of that boot, not just the last screenful. A dependency can fail first and cause several later consumers to print similar hook errors.

Confirm the result

Both the provider registration and consumer hook should complete, and one test payment should change the documented balance without an exception.

If the result differs

An enabled permissions plugin is not an economy plugin. If an API library loaded but no service registered, read the provider’s dependency and Java requirements rather than adding random jars.

Recover safely

Restore the previous tested dependency set if a replacement adds new startup errors. Keep player-money files and databases unchanged until the provider boots cleanly.

Documentation and related help

Check the Vault maintainer documentation and Vault maintainer reference against your installed release before applying changes.

Related CloudExa help: Connect Vault to an EssentialsX economy and Find Minecraft logs and send useful troubleshooting details.

Investigate different balances in a shop and a balance command

A balance mismatch needs a controlled comparison. Menus can display cached values, a separate currency or the wrong player identity even when the underlying payment succeeds.

Before you start

Vault connects consumer plugins to an economy provider; it does not replace that provider or store your player balances. Identify the component holding the money before changing integrations, and check the provider name in startup logs.

Work through the check

1. Write down the player UUID, backend server, exact balance command and shop or scoreboard showing the mismatch.
2. Read startup hook messages to identify the economy service selected by each consumer; distinguish money from points or item currencies.
3. Record the balance before and after a single small purchase, using the provider command between each observation.
4. Close and reopen the menu, reconnect the same account, then repeat on the same backend without changing any player data.

What to compare

Compare the amount at the same moment after the transaction settles. A sidebar placeholder may refresh less frequently than a direct provider command, so a brief stale display is not automatically missing money.

Confirm the result

A correct result ties the transaction delta to one provider account. A stale display should refresh without another debit or compensation payment.

If the result differs

If values differ only on another backend, inspect its provider and storage configuration. If only a scoreboard differs, investigate that placeholder or refresh interval before modifying economy storage.

Recover safely

Keep the transaction notes and pause the affected feature if you cannot reconcile deductions. Refund only a verified amount through the active provider, not by editing unidentified player files.

Documentation and related help

Check the Vault maintainer documentation and Vault maintainer reference against your installed release before applying changes.

Related CloudExa help: Connect Vault to an EssentialsX economy and Find Minecraft logs and send useful troubleshooting details.

Plan a Vault economy-provider change without losing balances

Switching the plugin that owns money is a data migration. Vault offers an integration API; replacing its provider does not by itself prove that balances moved to the new backend.

Before you start

Vault connects consumer plugins to an economy provider; it does not replace that provider or store your player balances. Identify the component holding the money before changing integrations, and check the provider name in startup logs.

Work through the check

1. List every consumer, including shops, jobs, claims and auction features, and confirm the new provider supports their required money model.
2. Export or back up the current provider using its supported method, preserving UUIDs, decimal precision and any world-specific accounts.
3. Rehearse the provider’s documented import on a disconnected test copy with a separate database and no live integrations.
4. Compare selected low, high and fractional balances, then test one consumer debit and credit before planning the live maintenance window.

What to compare

Agree how a duplicate account or unsupported old currency will be handled before migration. Keep an account-count and total-balance comparison as well as a few sample players, because spot checks can miss omissions.

Confirm the result

The test should preserve account identities and balances and demonstrate each consumer using the intended service after a clean restart.

If the result differs

An empty new account is not evidence of a successful import. Stop if the migration method cannot represent your existing currencies, negative balances or per-world behavior.

Recover safely

Keep the original provider and its consistent data snapshot available. Roll back the whole money-storage checkpoint before reopening transactions if reconciliation fails; avoid running both writable backends in parallel.

Documentation and related help

Check the Vault maintainer documentation and Vault maintainer reference against your installed release before applying changes.

Related CloudExa help: Connect Vault to an EssentialsX economy and Find Minecraft logs and send useful troubleshooting details.

Distinguish Vault economy failures from permissions failures

A denied shop action and a missing money hook need different fixes. Capture the exact message so you do not respond to an economy error by granting broad administrative access.

Before you start

Vault connects consumer plugins to an economy provider; it does not replace that provider or store your player balances. Identify the component holding the money before changing integrations, and check the provider name in startup logs.

Work through the check

1. Reproduce with one non-operator account and record whether the failure happens when opening the feature or during payment.
2. Check the consumer’s own permission documentation and compare the required node with that account’s effective permissions.
3. Read the startup economy hook and the provider balance for the same account; keep these checks separate from permission inheritance.
4. Allow only the documented task permission on a test role, then retry a low-value transaction and remove the role afterward.

What to compare

Use one known successful payment as a control. If the provider processes that payment but a particular user cannot open the shop command, investigate that access path separately from service registration.

Confirm the result

The account should open only the intended feature and complete the payment through the provider, while unrelated administrative commands remain denied.

If the result differs

If access succeeds but charging fails, inspect provider registration, balance limits and the transaction error. If no access check passes, examine world or server permission contexts instead.

Recover safely

Remove temporary grants after the test. Leave Vault and money data unchanged when the evidence points to authorization, and restore the original role if the new node grants more access than intended.

Documentation and related help

Check the Vault maintainer documentation and Vault maintainer reference against your installed release before applying changes.

Related CloudExa help: Connect Vault to an EssentialsX economy and Find Minecraft logs and send useful troubleshooting details.

Prepare a useful report for a Vault integration error

A maintainer can investigate an integration error faster when you identify both sides of the bridge. A screenshot saying that a plugin is red omits the dependency that failed first.

Before you start

Vault connects consumer plugins to an economy provider; it does not replace that provider or store your player balances. Identify the component holding the money before changing integrations, and check the provider name in startup logs.

Work through the check

1. Copy the server software, Java version, Vault build, provider build and affected consumer build from the running service.
2. Capture the first startup exception and its surrounding hook messages, then note the exact action and timestamp that trigger the later error.
3. Reproduce with a low-value test account and record the before-and-after balance, including whether an item or reward arrived.
4. Compare the reproduction on a backed-up test copy with optional consumers removed one at a time, keeping required dependencies intact.

What to compare

Include the first relevant exception and its surrounding startup lines in the report. A final message saying a consumer disabled itself usually explains less than the earlier dependency error that caused it.

Confirm the result

Your report should name the active provider and show a repeatable action, a complete exception chain and the observed transaction outcome.

If the result differs

Redact database passwords, tokens and player addresses before sharing logs. Do not upload an entire server directory or use a public diagnostic upload without reviewing its contents.

Recover safely

Keep production transactions paused if outcomes are inconsistent. Restore any temporarily removed test plugins and their matching configuration before using that test copy for another comparison.

> CloudExa game console with the log controls visible.
> The CloudExa console provides the log view used in this check. This is a general panel screenshot, not plugin-specific output.

Documentation and related help

Check the Vault maintainer documentation and Vault maintainer reference against your installed release before applying changes.

Related CloudExa help: Connect Vault to an EssentialsX economy and Find Minecraft logs and send useful troubleshooting details.