Quests tasks

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

- Test whether a Quests item requirement consumes the offered item
- Require a completed Quests prerequisite without blocking new players
- Choose when a Quests stage action should run
- Check a daily Quests schedule against cooldown and timezone
- Find which requirement prevents an NPC from offering a quest
- Test quest rewards with a nearly full player inventory
- Distinguish a Quests command permission from quest eligibility
- Plan a Quests storage change without starting with empty progress

Test whether a Quests item requirement consumes the offered item

A quest requirement can check an item and may also remove it according to its settings. Test that distinction before asking players to bring valuable custom equipment.

Check the installed content

PikaMug Quests has its own quest editor and player progress storage. Confirm that you use this plugin rather than another project named Quests. Current maintainer documentation targets version 5 and newer; use the legacy reference when your installed release needs it.

Work through the change

1. In game on staging, open the quest requirements through `/quests editor` and record the required item and quantity. Compare the consumption option with the installed-version requirements reference.
2. Prepare a disposable ordinary account with exactly the required quantity and record its inventory. Keep the quest reward low-value so the test isolates acceptance rather than profit.
3. Accept the quest through the normal player route and inspect the inventory immediately. Repeat with one fewer item and with a similar item that should not satisfy the match.
4. Test cancellation and a second acceptance only where the quest rules allow it. Record when the item is checked and when it is removed rather than assuming cancellation refunds a consumed requirement.

Verify the result

The quest should reject insufficient or mismatched items and consume only what the configured acceptance policy requires. The player-facing description should explain the cost accurately.

If the result differs

If a custom item fails, inspect the requirement matching options and its provider metadata. If the correct item disappears at an unexpected stage, check action events and objectives for a second removal path.

Keep a recovery path

Restore the previous requirement and disposable inventory checkpoint if the trial is wrong. Keep valuable production items out of testing and do not promise an automatic cancellation refund unless that behaviour has been deliberately implemented.

References and related help

Check the maintainer reference: PikaMug Quests documentation, PikaMug Quests documentation.

Before changing live data, read how to keep a completed recovery copy. For startup errors, use the log collection guide.

Require a completed Quests prerequisite without blocking new players

A follow-up quest should become available after the intended prerequisite rather than after an unrelated quest or a display-name match. Test the chain with distinct player states.

Check the installed content

PikaMug Quests has its own quest editor and player progress storage. Confirm that you use this plugin rather than another project named Quests. Current maintainer documentation targets version 5 and newer; use the legacy reference when your installed release needs it.

Work through the change

1. Record the prerequisite quest and follow-up quest identities. Save the quest configuration and consult the installed requirements reference for the supported completed-quest requirement.
2. Set the requirement on a staging copy through the quest editor. Keep the quest IDs and prerequisite references consistent instead of renaming live quests during the change.
3. Use one disposable account that has not completed the prerequisite and another that legitimately has. Attempt the follow-up through the same NPC or command route.
4. Complete the prerequisite with the first account and attempt the follow-up again. Include any repeatability rule so a completed follow-up does not confuse the prerequisite test.

Verify the result

The new player should be denied until completion, then become eligible without an administrator editing their progress. The already qualified account should be able to start the follow-up under its normal rules.

If the result differs

If both accounts are denied, inspect the stored completion identity and other requirements. If both pass, check that the requirement was saved on the quest being offered rather than a similarly named copy.

Keep a recovery path

Restore the earlier quest definition if the chain is blocked. Preserve the player data and avoid deleting completion records to retest; disposable accounts provide a safer comparison for acceptance state.

References and related help

Check the maintainer reference: PikaMug Quests documentation, PikaMug Quests documentation.

Before changing live data, read how to keep a completed recovery copy. For startup errors, use the log collection guide.

Choose when a Quests stage action should run

A stage action can fire on entry, completion or another event. Choose its lifecycle deliberately so a cosmetic message or reward does not run at the wrong moment.

Check the installed content

PikaMug Quests has its own quest editor and player progress storage. Confirm that you use this plugin rather than another project named Quests. Current maintainer documentation targets version 5 and newer; use the legacy reference when your installed release needs it.

Work through the change

1. In game, create or inspect the action through `/quests actions` using the installed-version action editor. Start with a harmless message rather than inventory removal, weather or valuable item issuance.
2. Open the staging quest stage and identify the documented action hook required by the design. A stage also needs its actual objective; an action is not automatically an objective.
3. Test entry, partial progress and completion with an ordinary account. Record the message timestamps and compare them with the selected start or end hook.
4. Test any relevant failure, death or disconnect hook separately. Keep a command or chat-triggered action distinct from the stage completion path so repeated player input cannot issue an unintended reward.

Verify the result

The action should run only at the intended lifecycle point, while the objective controls progression. Ordinary partial activity should not be mistaken for completion or repeat a one-time reward.

If the result differs

If the action never runs, inspect its saved reference and selected hook. If it runs repeatedly, check chat or command hooks and other stage actions before adding conditions to an unrelated objective.

Keep a recovery path

Restore the harmless action or previous hook if timing is unclear. Do not test destructive actions against real inventories, and keep the quest and action definitions backed up together so their references remain recoverable.

References and related help

Check the maintainer reference: PikaMug Quests documentation, PikaMug Quests documentation.

Before changing live data, read how to keep a completed recovery copy. For startup errors, use the log collection guide.

Check a daily Quests schedule against cooldown and timezone

A daily quest window and a player completion cooldown are separate clocks. Check both before changing a schedule that appears unavailable to players.

Check the installed content

PikaMug Quests has its own quest editor and player progress storage. Confirm that you use this plugin rather than another project named Quests. Current maintainer documentation targets version 5 and newer; use the legacy reference when your installed release needs it.

Work through the change

1. Record the intended opening hours, timezone and repeat policy. Save the planner configuration and use the installed planner editor rather than guessing the date-field order in YAML.
2. Compare the start and end window, repeat interval and player cooldown with the maintainer reference. Its planner intervals are expressed in seconds, and the repeat cycle is based on the start time.
3. Create a short disposable staging schedule with an observable window. Test before opening, during availability and after closing, using accounts with and without a recent completion.
4. Check the documented cooldown override policy at the next repeat boundary. Record server timezone and timestamps so a daily opening is not confused with a fixed delay since the player last finished.

Verify the result

The quest should open within its intended window and enforce the chosen completion policy. A qualified player at the next cycle should behave according to the explicit override setting.

If the result differs

If everyone is denied, inspect the window and timezone first. If only recent completers fail, inspect their cooldown rather than moving the global start date or clearing all player records.

Keep a recovery path

Restore the previous planner values if the trial behaves unexpectedly. Keep production scheduling unchanged until the short-window test is understood, and preserve completion data instead of deleting it to bypass a timer.

References and related help

Check the maintainer reference: PikaMug Quests documentation.

Before changing live data, read how to keep a completed recovery copy. For startup errors, use the log collection guide.

Find which requirement prevents an NPC from offering a quest

A visible quest NPC does not prove that an account meets the quest requirements. Compare acceptance rules before recreating the NPC or removing every condition.

Check the installed content

PikaMug Quests has its own quest editor and player progress storage. Confirm that you use this plugin rather than another project named Quests. Current maintainer documentation targets version 5 and newer; use the legacy reference when your installed release needs it.

Work through the change

1. Record the NPC and the exact quest being offered, including its internal identity. Save the quest definition and check the requirements reference matching the installed release.
2. Compare the affected player with an eligible disposable test account. List money, items, permissions, completed quests and other requirements actually configured on this quest.
3. On staging, test one failing requirement at a time while keeping the NPC and quest route unchanged. Use harmless quantities and do not grant operator status to make the offer appear.
4. Once acceptance works, repeat with a deliberately ineligible account and check the rejection message. Confirm the player receives useful guidance without revealing staff-only commands or private data.

Verify the result

The NPC should offer the intended quest to eligible players and reject the specific failing case predictably. The test should identify the blocking requirement without changing other quests.

If the result differs

If acceptance works by command but not through the NPC, inspect the NPC association or dependency integration. If neither route works, inspect quest availability and requirements before replacing the NPC entity.

Keep a recovery path

Restore the prior requirement changes and remove temporary test permissions. Preserve quest progress and the NPC configuration; deleting an NPC or resetting all player records is unnecessary for an unexplained eligibility check.

References and related help

Check the maintainer reference: PikaMug Quests documentation, PikaMug Quests documentation.

Before changing live data, read how to keep a completed recovery copy. For startup errors, use the log collection guide.

Test quest rewards with a nearly full player inventory

A reward that works with empty pockets may behave differently when storage is almost full. Test delivery before advertising a valuable item reward to players.

Check the installed content

PikaMug Quests has its own quest editor and player progress storage. Confirm that you use this plugin rather than another project named Quests. Current maintainer documentation targets version 5 and newer; use the legacy reference when your installed release needs it.

Work through the change

1. Record the reward item types, quantities and command-based grants. Save the reward definition and check the installed rewards reference for the supported delivery settings.
2. Prepare a disposable staging account with a nearly full inventory and a completed recovery copy of its test state. Leave room for only part of the intended reward.
3. Complete the quest normally and inspect the inventory, nearby ground and relevant messages. Compare the same completion with an empty inventory using a separate eligible test state.
4. Test any reward integration that uses a command separately, since its delivery behaviour belongs to that plugin. Decide how the quest description should tell players to prepare inventory space.

Verify the result

The delivery outcome should be understood for both inventory states, with no assumed guarantee that every external reward command handles overflow the same way. Quantities should match the configured reward exactly.

If the result differs

If a reward appears missing, inspect drops and the command-provider logs before granting it again. If the quest completed but an integration failed, record that stage rather than repeatedly replaying completion.

Keep a recovery path

Restore the reward definition and disposable inventory checkpoint if needed. Keep a record of actual issued items so retries or compensation cannot duplicate a reward that was delivered somewhere unexpected.

References and related help

Check the maintainer reference: PikaMug Quests documentation.

Before changing live data, read how to keep a completed recovery copy. For startup errors, use the log collection guide.

Distinguish a Quests command permission from quest eligibility

A player who cannot open the quest interface may lack a command permission even when they meet every quest requirement. Check access before editing the quest itself.

Check the installed content

PikaMug Quests has its own quest editor and player progress storage. Confirm that you use this plugin rather than another project named Quests. Current maintainer documentation targets version 5 and newer; use the legacy reference when your installed release needs it.

Work through the change

1. Capture the exact command and rejection message. Confirm that the server uses PikaMug Quests and consult its installed-version command and permission reference.
2. Compare the affected ordinary account with a working non-operator account. Inspect inherited permission groups and any command aliases or translations that route the request differently.
3. Grant the narrow documented interface permission to a temporary staging group. Test opening the interface, viewing the quest and attempting acceptance as separate steps.
4. Use an account that deliberately fails a quest requirement to confirm interface access does not bypass eligibility. Also check that administrative editing commands remain denied.

Verify the result

The account should use the intended player interface while the quest still enforces its own conditions. Staff-only editing or progress-management commands should remain unavailable.

If the result differs

If the interface opens but the quest is unavailable, inspect its schedule and requirements. If the command still fails, check alias ownership and inherited denials before granting broad plugin permissions.

Keep a recovery path

Restore the previous permission assignment if the test grants more access than intended. Remove the temporary group and keep quest definitions and player progress unchanged while resolving a command-access problem.

References and related help

Check the maintainer reference: PikaMug Quests documentation, PikaMug Quests documentation.

Before changing live data, read how to keep a completed recovery copy. For startup errors, use the log collection guide.

Plan a Quests storage change without starting with empty progress

Changing storage settings alone is not a safe progress migration. Prepare a supported conversion and comparison before pointing a live quest server at a different database.

Check the installed content

PikaMug Quests has its own quest editor and player progress storage. Confirm that you use this plugin rather than another project named Quests. Current maintainer documentation targets version 5 and newer; use the legacy reference when your installed release needs it.

Work through the change

1. Record the current storage backend and installed Quests version. Take a completed backup of quest definitions and all current player progress while the staging copy is stopped.
2. Read the storage documentation and its supported conversion tooling. Keep the target database separate and empty for the trial rather than mixing copied records with an existing live dataset.
3. Run the documented conversion procedure on staging with the matching tool versions. Keep credentials private and avoid editing tables or assuming that switching the backend setting moves the data.
4. Start the staging server on the converted backend and compare several known accounts: active objectives, completed quests and cooldown state. Check that new progress persists through a normal restart.

Verify the result

The converted backend should preserve representative progress and store subsequent changes correctly. Record the source checkpoint and comparison results before planning a production maintenance window.

If the result differs

If accounts appear new, stop the trial and inspect the selected backend, database and conversion output. Do not let players continue on two diverging stores while deciding which contains the authoritative progress.

Keep a recovery path

Return staging to the original configuration and backed-up source data if validation fails. Keep both trial outputs for diagnosis, and prepare an explicit production cutover and rollback plan before any live storage change.

References and related help

Check the maintainer reference: PikaMug Quests documentation.

Before changing live data, read how to keep a completed recovery copy. For startup errors, use the log collection guide.