BetonQuest tasks

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

- Find a BetonQuest objective that was defined but never started
- Test an inverted BetonQuest condition with both player states
- Guard a BetonQuest reward against repeat conversation clicks
- Keep a BetonQuest journal entry separate from objective progress
- Repair a cross-package BetonQuest reference after organising files
- Check a delayed BetonQuest objective across logout and restart
- Match a BetonQuest quest item without accepting lookalikes
- Hide a completed quest option without hiding unrelated NPC dialogue
- Edit BetonQuest dialogue text without renaming conversation options
- Compare BetonQuest configuration errors with player-state errors

Find a BetonQuest objective that was defined but never started

An objective definition does not prove that it is active for a player. Follow the event that starts it before changing its target or required count.

Check the installed content

BetonQuest expresses a quest through objectives, conditions, events and conversation links. Keep the package and player state together during testing. The linked reference describes the 2.2 documentation branch; select the corresponding documentation if your server runs another major version.

Work through the change

1. Record the package, objective ID and player action that should begin it. Save the package and disposable player-state checkpoint together before editing the quest flow.
2. Read the objective and event references for the installed documentation branch. Trace the conversation or preceding objective to the event responsible for starting this objective.
3. Replace valuable completion rewards with a harmless observable test event on staging. Start the quest through its ordinary entry path and inspect objective state using the supported administration tools.
4. Perform one valid objective action, then complete the test count. Compare a second account that never started the quest so normal world activity is not mistaken for quest progress.

Verify the result

The objective should become active at the intended entry point and count only eligible actions for the correct player. Completion should call the expected next event without repeating the initial start action.

If the result differs

If it is inactive, inspect the start event reference and its conditions first. If active but not progressing, then inspect action matching, world restrictions and objective conditions instead of repeatedly starting another copy.

Keep a recovery path

Restore the previous package and disposable state if the trial becomes inconsistent. Keep live player objectives untouched during diagnosis; deleting progress or recreating the entire quest can hide the missing start event.

References and related help

Check the maintainer reference: BetonQuest documentation, BetonQuest documentation.

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

Test an inverted BetonQuest condition with both player states

An inverted condition can accidentally make a quest available only to the wrong players. Write a pass-and-fail table before changing the condition on a reward or conversation branch.

Check the installed content

BetonQuest expresses a quest through objectives, conditions, events and conversation links. Keep the package and player state together during testing. The linked reference describes the 2.2 documentation branch; select the corresponding documentation if your server runs another major version.

Work through the change

1. Record the condition ID and the player state it checks. Describe in plain language when the branch should be available and when it should be hidden or rejected.
2. Check the installed-branch scripting and condition references for negation and condition-list syntax. Keep the underlying condition definition separate from the place that inverts its result.
3. On staging, prepare one disposable account that satisfies the condition and one that does not. Use a harmless observable branch rather than currency or item issuance.
4. Run the same interaction with both accounts, change the relevant test state through supported tools and repeat. Inspect combined conditions individually before testing their full conjunction.

Verify the result

The branch should follow the written truth table for both states and their transitions. Inversion should change eligibility, without unexpectedly changing the event target or creating another reward path.

If the result differs

If both accounts behave alike, inspect the underlying state and package reference. If one additional condition changes the result, test that condition separately rather than adding another negation as a guess.

Keep a recovery path

Restore the earlier condition reference and keep the reward harmless if any case fails. Preserve live player state; changing a condition should not require clearing tags, objectives or completed quests for every account.

References and related help

Check the maintainer reference: BetonQuest documentation, BetonQuest documentation.

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

Guard a BetonQuest reward against repeat conversation clicks

A reward conversation can be reopened, so its state checks need to prevent an unintended second grant. Test repeated entry before attaching valuable currency or equipment.

Check the installed content

BetonQuest expresses a quest through objectives, conditions, events and conversation links. Keep the package and player state together during testing. The linked reference describes the 2.2 documentation branch; select the corresponding documentation if your server runs another major version.

Work through the change

1. Record the reward event and all conversation or objective paths that can call it. Save the package and identify the intended completion tag or other persistent eligibility state.
2. Check the installed-branch event and condition references for the supported state operations. Build a staging reward with a harmless observable action and a clear eligible-versus-completed condition.
3. Trigger the legitimate completion path once, then reopen the conversation and repeat the same click. Test relogging and an orderly restart so the check covers persisted state.
4. Test a second account that has not completed the quest. Confirm the anti-repeat guard does not block first-time completion or accidentally use a global state shared by every player.

Verify the result

The eligible account should receive one grant, repeat entry should be denied or take a non-reward branch, and the untouched account should remain independently eligible.

If the result differs

If repeat clicks grant again, inspect whether the completion state is set and checked on every calling path. This logical guard is not a guarantee of transactional delivery across a server crash; assess valuable external rewards separately.

Keep a recovery path

Restore the harmless event if repeat protection fails and keep the live reward unavailable. Preserve the test evidence and account states; do not repeatedly grant real rewards while debugging a completion guard.

References and related help

Check the maintainer reference: BetonQuest documentation, BetonQuest documentation.

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

Keep a BetonQuest journal entry separate from objective progress

A journal can describe progress without starting or completing the underlying objective. Check the event paths independently when the book and quest behaviour disagree.

Check the installed content

BetonQuest expresses a quest through objectives, conditions, events and conversation links. Keep the package and player state together during testing. The linked reference describes the 2.2 documentation branch; select the corresponding documentation if your server runs another major version.

Work through the change

1. Record the journal entry ID, objective ID and the point at which each should change. Save the package and a disposable test-player checkpoint before editing the flow.
2. Inspect the events that add or remove the journal entry and those that start or finish the objective. Check their references against the documentation matching your installed branch.
3. On staging, enter the quest normally and inspect both the journal and active objective state. Perform partial progress and completion while recording the resulting book text.
4. Test relogging and a normal restart, then compare a second account outside the quest. Confirm that narrative text and progression are updated for the intended player only.

Verify the result

The journal should accurately describe the active or completed state, while objective tracking should work independently of whether the player has opened the book.

If the result differs

If text changes but progress does not, inspect the objective event chain. If progress works but the journal is stale, inspect entry references and update events rather than resetting the quest.

Keep a recovery path

Restore the saved journal and event definitions together if the trial becomes misleading. Keep production player progress intact and correct the presentation path separately from any genuine missing objective state.

References and related help

Check the maintainer reference: BetonQuest documentation, BetonQuest documentation.

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

Repair a cross-package BetonQuest reference after organising files

Moving a quest package can change how other packages reach its features. Map the references before renaming folders so an organisation change does not disconnect a live quest chain.

Check the installed content

BetonQuest expresses a quest through objectives, conditions, events and conversation links. Keep the package and player state together during testing. The linked reference describes the 2.2 documentation branch; select the corresponding documentation if your server runs another major version.

Work through the change

1. Back up the complete QuestPackages tree and list the package being moved. Record the events, objectives, conditions and conversations referenced by other packages.
2. Read the package-path rules for the installed documentation branch. Distinguish a normal subfolder from a folder containing its own package.yml, which creates a separate package boundary.
3. Move a small copied package on staging and update the affected references consistently. Use documented absolute or relative package addressing rather than inserting filesystem paths into event IDs.
4. Load the staging quest set and test one entry from each dependent package. Inspect the first unresolved-reference error before continuing through the rest of the chain.

Verify the result

Every dependent call should resolve to the intended feature after organisation, with no duplicate IDs within the same package. The player should reach the next quest stage through its ordinary entry route.

If the result differs

If a local reference works but an external one fails, inspect the package path and boundary. If both fail, check whether the feature definition loaded before changing cross-package syntax again.

Keep a recovery path

Restore the original directory tree and package references together if the move breaks the chain. Avoid renaming live persistent quest identities during a file tidy-up without a separate state-migration plan.

References and related help

Check the maintainer reference: BetonQuest documentation.

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

Check a delayed BetonQuest objective across logout and restart

A waiting objective may use a different timing policy than players expect. Test offline and restart behaviour before advertising a real-time deadline or a guaranteed online-only delay.

Check the installed content

BetonQuest expresses a quest through objectives, conditions, events and conversation links. Keep the package and player state together during testing. The linked reference describes the 2.2 documentation branch; select the corresponding documentation if your server runs another major version.

Work through the change

1. Record the installed BetonQuest version, objective type and intended timing policy. Read the matching objective reference, including any options that affect delay or persistence.
2. Create a short disposable staging wait objective with a harmless completion event. Record its activation time and compare it with the server clock.
3. Run one test while remaining online and another while logging out during the wait. Rejoin and inspect whether the objective completed or retained remaining work according to its documented policy.
4. Run a separate test through an orderly staging restart. Keep the server unavailable interval in the timing record and compare the resulting state before increasing the production delay.

Verify the result

The measured behaviour should match the documented objective policy for online, offline and restarted cases. The quest description should explain that policy rather than promise a different kind of clock.

If the result differs

If a wait never finishes, confirm it was active for the account and inspect its completion event. If timing differs only during shutdown, review persistence semantics before shortening every deadline.

Keep a recovery path

Restore the prior delay definition and disposable player checkpoint if the trial fails. Do not clear live waiting objectives to make them disappear; assess any needed correction against the affected player states separately.

References and related help

Check the maintainer reference: BetonQuest documentation.

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

Match a BetonQuest quest item without accepting lookalikes

A renamed ordinary item can resemble a quest item without sharing its required identity. Test matching rules before a delivery objective accepts valuable substitutes or rejects legitimate rewards.

Check the installed content

BetonQuest expresses a quest through objectives, conditions, events and conversation links. Keep the package and player state together during testing. The linked reference describes the 2.2 documentation branch; select the corresponding documentation if your server runs another major version.

Work through the change

1. Record the quest item definition and the condition or objective that checks it. Save the package and identify which properties the installed branch uses for matching.
2. Prepare expendable staging examples: the legitimate item, a plain item of the same material and a renamed lookalike. Keep their quantities and creation paths recorded.
3. Test the relevant item condition or delivery objective with each example using an ordinary account. Inspect the required quantity and any removal event separately from the match itself.
4. If custom-item integrations are involved, verify their supported matching method in the installed release. Repeat after a restart so the outcome is not dependent on a temporary test item.

Verify the result

The legitimate item should pass, lookalikes should fail where the quest requires a distinct identity, and the intended amount should be counted or removed without touching unrelated inventory.

If the result differs

If every same-material item passes, inspect how strict the definition is rather than increasing quantity. If the legitimate item fails, compare the producing reward and matching definition before issuing replacement items.

Keep a recovery path

Restore the earlier item definition and disposable inventory checkpoint if matching is uncertain. Keep valuable player inventory outside the trial and do not delete similar items globally to enforce a quest rule.

References and related help

Check the maintainer reference: BetonQuest documentation, BetonQuest documentation.

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

Hide a completed quest option without hiding unrelated NPC dialogue

A completion condition should affect the intended conversation option rather than the whole NPC. Test first-time and returning players before changing the conversation entry point.

Check the installed content

BetonQuest expresses a quest through objectives, conditions, events and conversation links. Keep the package and player state together during testing. The linked reference describes the 2.2 documentation branch; select the corresponding documentation if your server runs another major version.

Work through the change

1. Record the option ID and completion state that should hide it. Save the conversation and relevant condition definitions together before editing the staging copy.
2. Place the documented condition on the narrow option or branch intended by the design. Keep the general greeting and unrelated dialogue paths available unless the story explicitly restricts them.
3. Test an account without completion and an account with the legitimate completed state. Follow both the quest option and one unrelated dialogue branch from the same NPC.
4. Complete the quest with the first account and reopen the conversation. Verify the option visibility and the fallback dialogue, including the case where no quest-specific branch is eligible.

Verify the result

New players should reach the quest option, completed players should receive appropriate returning dialogue, and unrelated NPC functions should remain available to both accounts.

If the result differs

If the entire conversation disappears, inspect conditions on its starting path. If a hidden option is still reachable elsewhere, trace every pointer to that branch instead of adding a second completion tag blindly.

Keep a recovery path

Restore the previous condition placement and conversation pointers if navigation breaks. Preserve live player completion state; a dialogue-visibility adjustment should not reset quests or remove NPC entities.

References and related help

Check the maintainer reference: BetonQuest documentation, BetonQuest documentation.

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

Edit BetonQuest dialogue text without renaming conversation options

Wording can be improved without changing the internal IDs that connect a conversation. Keep text editing separate from flow changes so existing references remain valid.

Check the installed content

BetonQuest expresses a quest through objectives, conditions, events and conversation links. Keep the package and player state together during testing. The linked reference describes the 2.2 documentation branch; select the corresponding documentation if your server runs another major version.

Work through the change

1. Back up the conversation and record the option IDs and pointers before editing. Note any translations or external events that reference the same conversation features.
2. Change only the player-facing text on staging. Keep option keys, event references and condition IDs unchanged, and follow the installed-branch quoting rules for punctuation and multiline YAML.
3. Load the quest configuration and inspect the first parsing error if one appears. Walk every edited branch with an ordinary test player, including back links and the normal exit.
4. Compare the edited language with any other available locale and check variable rendering. Confirm that translated text has not accidentally become a new option ID or broken pointer.

Verify the result

The conversation should retain its original routes and events while displaying the intended wording. Variables should resolve correctly and every edited option should still reach its expected next node.

If the result differs

If a branch disappears, compare its key and pointer before changing conditions. If all dialogue fails to load, inspect quoting or indentation around the first edited text rather than recreating the whole package.

Keep a recovery path

Restore the saved conversation file if a text-only edit alters navigation. Keep a diff of the wording so it can be reapplied carefully without renaming persistent or externally referenced quest features.

References and related help

Check the maintainer reference: BetonQuest documentation.

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

Compare BetonQuest configuration errors with player-state errors

A quest that fails for one player may have valid configuration but different stored state. Compare accounts before changing a working quest path or clearing progress.

Check the installed content

BetonQuest expresses a quest through objectives, conditions, events and conversation links. Keep the package and player state together during testing. The linked reference describes the 2.2 documentation branch; select the corresponding documentation if your server runs another major version.

Work through the change

1. Record the quest package, failing interaction and first relevant log message. Preserve the package files and player storage checkpoint before making a staging reproduction.
2. Load the same package on staging and inspect unresolved definitions or syntax errors. Fix a genuine load failure before interpreting differences between player accounts.
3. Use one disposable account in the expected starting state and another reproducing the affected state. Compare active objectives, relevant tags and eligibility conditions through supported inspection tools.
4. Run the same interaction on both accounts and trace the first differing decision. Keep a written record of state changes rather than clearing everything until the quest happens to work.

Verify the result

The comparison should identify whether the failure is shared configuration or a specific state transition. A proposed fix should pass both the originally working route and the reproduced failing route.

If the result differs

If both accounts fail identically, return to the shared definitions and event chain. If only one fails, inspect its precise condition or objective state before changing global rewards or dialogue.

Keep a recovery path

Restore the previous staging package and checkpoints if the trial introduces another fault. Any live state repair should be a separate narrow operation with a completed backup, not an undocumented full reset of the player quest history.

References and related help

Check the maintainer reference: BetonQuest documentation, BetonQuest documentation, BetonQuest documentation.

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