InteractiveChat tasks

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

- Limit repeated InteractiveChat placeholders with a measured cooldown
- Restrict one InteractiveChat custom placeholder to an intended group
- Keep InteractiveChat command links as suggestions rather than execution
- Diagnose an InteractiveChat formatter conflict one event at a time
- Reduce InteractiveChat mention noise without disabling normal chat
- Explain an expired InteractiveChat inventory link without resetting inventories

Limit repeated InteractiveChat placeholders with a measured cooldown

Item and inventory links can crowd chat when players repeat them. Use the supported limits for those features rather than slowing every normal message or removing the chat formatter that players already use.

Prepare the test

Use disposable example items and ordinary player accounts for chat tests. Keep private inventory contents, staff commands and personal data away from demonstrations that other players can click or hover.

Make the change

1. Choose two test accounts and record the current behaviour for one placeholder, two placeholders in a message and a second immediate message. Keep the examples harmless and use a quiet testing period.
2. Preserve InteractiveChat `config.yml` and inspect the installed Settings comments for `UniversalCooldown` and `MaxPlaceholders`. The upstream configuration documents the universal cooldown in seconds and uses -1 to disable the per-message limit.
3. Choose limits that allow an ordinary player to share an item without flooding chat. Apply the change using the installed release’s supported method, leaving packet-size, threading and unrelated moderation settings unchanged.
4. Repeat the three original tests with an ordinary account. Check the cooldown or limit message and confirm normal text without a placeholder still follows the intended chat policy, then allow enough time for the cooldown to expire.

Check the result

Repeated interactive links should be limited predictably, while a legitimate later share works. Verify a second account independently so one player’s restriction is not mistaken for a global cooldown affecting everyone on the server.

If the test fails

Restore the prior limits if normal use becomes frustrating or valid messages are rejected. If a separate anti-spam plugin also intervenes, compare their messages and test each documented policy rather than disabling all moderation.

Use the generated file for your installed version as the final key reference. Upstream master can change ahead of a release, so avoid pasting a whole current template over an older working configuration.

References and related help

Check the official InteractiveChat documentation against your installed release before using a command or setting from another version.

- Test InteractiveChat item previews without exposing staff inventory
- Find Minecraft logs and send useful troubleshooting details

Restrict one InteractiveChat custom placeholder to an intended group

A custom interactive keyword may be useful only for a builder or staff group. Inspect how your installed release assigns per-placeholder permissions before allowing a private action through the same rules as public item sharing.

Prepare the test

Use disposable example items and ordinary player accounts for chat tests. Keep private inventory contents, staff commands and personal data away from demonstrations that other players can click or hover.

Make the change

1. Identify the selected custom placeholder in the current configuration and record its position and intended audience. Inspect its hover and click behaviour for private data or command actions before making it available to more accounts.
2. Preserve the file and review `UseCustomPlaceholderPermissions` in the generated Settings section. The upstream comments describe numbered permission nodes such as `interactivechat.module.custom.1`; match numbering to the installed configuration rather than guessing from a display name.
3. Enable the documented permission behaviour and grant only the selected node to a disposable test group. Inspect effective inherited permissions and explicit denies for one permitted account and one ordinary unpermitted account.
4. Apply the change through the release-supported method, then send the custom keyword from each account. Verify its rendered action as well as its visible label; an attractive label does not prove the audience restriction works.

Check the result

The permitted group should use the intended feature, while the unpermitted group should receive the configured restricted behaviour. Confirm public built-in sharing still works as planned and that no staff-only custom action became available accidentally.

If the test fails

Restore the saved permission mode if the configuration changes access more broadly than intended. Remove only the trial permission assignment and inspect numbering again before changing global rank inheritance or granting a wildcard.

Recheck these numbered mappings after adding, removing or reordering custom entries. Keep a small local record linking the configured entry to its approved audience so a later edit does not silently reassign access.

References and related help

Check the official InteractiveChat documentation against your installed release before using a command or setting from another version.

- Test InteractiveChat item previews without exposing staff inventory
- Create Minecraft ranks and inheritance with LuckPerms

Keep InteractiveChat command links as suggestions rather than execution

A clickable command can insert text for review or execute an action immediately. Decide which behaviour players should receive before changing InteractiveChat’s command display, especially when the example includes user-controlled text.

Prepare the test

Use disposable example items and ordinary player accounts for chat tests. Keep private inventory contents, staff commands and personal data away from demonstrations that other players can click or hover.

Make the change

1. Choose a harmless informational command and two ordinary test players. Record the current click behaviour and identify whether another chat plugin also adds command interactions to the same message.
2. Preserve InteractiveChat `config.yml` and inspect its Commands section. The upstream configuration lists `SUGGEST_COMMAND` and `RUN_COMMAND` as different action choices; keep the suggestion action when the player should review text before submitting it.
3. Apply a small formatting change to the test command display without altering its authority or adding console execution elsewhere. Check the hover text so it accurately describes whether a click inserts a suggestion or runs something.
4. Send the example from an ordinary account and click it on the second client. Inspect the chat input before pressing Enter, then test the same display with a harmless longer argument and confirm the suggested text remains understandable.

Check the result

Clicking should insert the intended command for review when suggestion behaviour is selected. The action must still respect normal command permissions after the player chooses to submit it; a chat link should not grant administrative capability.

If the test fails

Restore the original Commands section if the action executes unexpectedly or another plugin replaces it. Disable the trial interactive command display while tracing ownership rather than testing with teleport, permission or destructive administration commands.

Describe the behaviour clearly in help text. A label saying click to use can be ambiguous, so tell players when clicking merely prepares a command that they must review and send themselves.

References and related help

Check the official InteractiveChat documentation against your installed release before using a command or setting from another version.

- Test InteractiveChat item previews without exposing staff inventory
- Find Minecraft logs and send useful troubleshooting details

Diagnose an InteractiveChat formatter conflict one event at a time

A formatter can replace or cancel a message before its interactive parts reach players. Capture the input, resulting text and plugin ownership before changing event priorities or advanced packet-handling options.

Prepare the test

Use disposable example items and ordinary player accounts for chat tests. Keep private inventory contents, staff commands and personal data away from demonstrations that other players can click or hover.

Make the change

1. Use one ordinary sender and receiver to compare plain text with a single harmless interactive keyword. Record whether formatting, hover content, click actions or the entire message disappears, and whether the console reports cancellation.
2. Identify the chat formatter and moderation plugins involved and note their versions. Preserve the current InteractiveChat configuration and compare the installed `EventPriorities` comments with the upstream reference for the specific event path you are investigating.
3. On a staging copy, change one documented priority only and repeat the identical message. Do not enable advanced Paper chat event switches merely because the server runs Paper; the upstream template explicitly warns against that assumption.
4. Check both public chat and any configured private-message parsing route. Once a supported combination works, apply that narrow change during an agreed window and confirm moderation still cancels messages that it is supposed to reject.

Check the result

The intended interactive content should survive the formatter while ordinary styling and moderation remain effective. Record the accepted priority and plugin versions so a future update can be compared with the known working combination.

If the test fails

Restore the original priority if messages duplicate, bypass a filter or lose their author context. Provide the minimal reproduction to the relevant maintainers rather than disabling every chat listener or increasing packet limits at random.

Keep staging messages free of private inventory data. The purpose is to identify event ownership, so a simple disposable item and a short fixed sentence provide better evidence than a crowded public chat capture.

> CloudExa game panel console and logs view for collecting startup and diagnostic evidence
> Collect the relevant timestamped log text in the panel, then remove private data before sharing it with support.

References and related help

Check the official InteractiveChat documentation against your installed release before using a command or setting from another version.

- Test InteractiveChat item previews without exposing staff inventory
- Find Minecraft logs and send useful troubleshooting details

Reduce InteractiveChat mention noise without disabling normal chat

Mention sounds and titles can become disruptive even when the chat message itself is reasonable. Review the mention controls independently so a noise complaint does not lead to removing ordinary player conversation.

Prepare the test

Use disposable example items and ordinary player accounts for chat tests. Keep private inventory contents, staff commands and personal data away from demonstrations that other players can click or hover.

Make the change

1. Choose two cooperating test players and agree on a short mention test. Record whether the current behaviour produces a sound, title, other visual notification or repeated alerts when the same sender mentions the same player.
2. Preserve `config.yml` and inspect the installed Chat section. Compare `AllowMention`, `MentionCooldown` and the available notification settings with the upstream configuration; use its documented units rather than assuming every duration follows the same format.
3. Adjust the one notification or cooldown responsible for the complaint. If your policy restricts broad mentions, inspect the documented here and everyone controls separately instead of assuming an individual mention cooldown handles every broadcast case.
4. Apply the change using the installed supported method and repeat the same messages. Test an immediate repeat, a later mention after the cooldown and ordinary non-mention text while both players remain connected.

Check the result

The intended alert should occur at a sensible frequency, and normal chat should remain readable. Check the receiving client’s own sound and title settings before declaring a missing notification a server failure.

If the test fails

Restore the prior section if the change suppresses wanted notifications or affects unrelated formatting. If another chat plugin also sends mention alerts, trace its configuration separately so two independent sounds are not mistaken for a failed cooldown.

Use the actual received message and notification timing as evidence. A configuration value alone does not show how a player with different client settings experiences the chosen mention policy.

References and related help

Check the official InteractiveChat documentation against your installed release before using a command or setting from another version.

- Test InteractiveChat item previews without exposing staff inventory
- Find Minecraft logs and send useful troubleshooting details

Explain an expired InteractiveChat inventory link without resetting inventories

Interactive inventory views are temporary representations rather than permanent inventory backups. An expired view message does not establish that the player’s actual items were deleted or that their world data needs restoring.

Prepare the test

Use disposable example items and ordinary player accounts for chat tests. Keep private inventory contents, staff commands and personal data away from demonstrations that other players can click or hover.

Make the change

1. Record the exact message shown when the old link is clicked and its approximate age. Ask the owner to inspect their real inventory in game, keeping private contents out of public screenshots and support messages.
2. Compare the installed message configuration and release documentation. The upstream template describes expiry for generated shared views; treat that behaviour separately from a failed fresh click or a missing permission to use sharing.
3. With consent, create a fresh share containing only a disposable example item. Have the second player open it promptly and compare the result with the old expired link, without changing the owner’s stored inventory files.
4. Test a second fresh link after reconnecting if the first fails. Note whether failure follows age, sender, viewer or server transfer; that distinction helps identify a temporary-view lifecycle issue rather than a general chat fault.

Check the result

A fresh permitted view should open while the old expired example remains unavailable as designed. The player’s real items should remain unchanged; keep that distinction explicit when explaining the result to your community.

If the test fails

If a fresh view fails immediately, gather its timestamp, versions and the relevant console error for support. Do not restore a world, clear player data or disable inventory protections to repair a temporary chat-view error.

Tell players not to use chat previews as an item backup or permanent proof of a later inventory state. For persistent troubleshooting, capture an agreed redacted screenshot and record the time it represents.

References and related help

Check the official InteractiveChat documentation against your installed release before using a command or setting from another version.

- Test InteractiveChat item previews without exposing staff inventory
- Find Minecraft logs and send useful troubleshooting details