VotingPlugin tasks

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

- Match a VotingPlugin site to the service name actually received
- Investigate an offline VotingPlugin reward before granting it again
- Review the command privileges in a VotingPlugin reward
- Distinguish a VotingPlugin vote party from a personal milestone
- Reduce repeated vote reminders without disabling voting
- Preserve monthly voting history before changing leaderboard tracking

Match a VotingPlugin site to the service name actually received

A listing’s public name, vote URL and incoming service identifier can differ. VotingPlugin uses the received service value to select a configured site. Correcting that mapping can restore rewards without recreating the site or altering player totals.

Before you change settings

Choose one listing and save VoteSites.yml before editing. Record the player and timestamp of a controlled external vote. Check the installed VotingPlugin release, because the maintainer documentation distinguishes current release behaviour from development-only validation features.

Follow the request through the plugin

1. Confirm both the Votifier receiver and VotingPlugin log receipt. If only the receiver logs an event, investigate listener integration before changing the site mapping.
2. Use the received log value or the documented `/av servicesites` command to identify the service string. Do not derive it from the website address or the menu item shown to players.
3. Find the intended site entry and compare its `ServiceSite` field. Matching is documented as case-insensitive, but other differences still matter. Preserve the site’s reward settings and internal key.
4. Apply the narrowly corrected mapping using the plugin’s supported reload or a scheduled restart. Send one further external test and compare the selected site and delivered reward.

Confirm the intended result

The real listing should select the intended configured site and award its ordinary reward once. Check the player-facing vote link still points to the correct listing. A local administrative vote is useful for reward processing but cannot establish what identifier the remote website sends.

If the test fails

If an unexpected extra site appears, inspect automatic site creation. Generated entries need review, including their vote URL and rewards. Do not enable valuable default rewards for every unknown identifier merely to silence an unmatched-site warning.

Restore the saved mapping if a working listing stops matching. Keep player history and the received service evidence. If the website sends a malformed identifier, correct the integration with that website rather than inventing several competing entries.

Sources and related help

- VotingPlugin maintainer reference 1
- VotingPlugin maintainer reference 2
- Create and test Minecraft vote rewards with VotingPlugin

Investigate an offline VotingPlugin reward before granting it again

An offline player may have a valid vote waiting for delivery, especially when the reward needs an online Bukkit player. Manually granting the reward before checking its queue can create a second payout when the player reconnects.

Before you change settings

Identify one vote by player, listing and timestamp. Save the current relevant configuration and record the player’s balance or expected items. Distinguish a vote still waiting in a proxy forwarding cache from a reward already accepted and queued by VotingPlugin.

Follow the request through the plugin

1. Find the VotingPlugin receipt for that event. If it has not reached the plugin, trace the public receiver and proxy transport instead of editing the reward queue.
2. Check that the vote username resolves to the intended player identity. On Bedrock networks, compare the actual configured prefix and UUID mode; do not toggle online mode to force a name match.
3. Inspect the reward actions and requirements. Permissions, worlds, player-run commands and GUI choices may depend on the player being online and in the correct context.
4. Have the player join the intended backend once and watch the bounded delivery log. Record the resulting balance or items before considering any manual correction.

Confirm the intended result

The queued reward should be delivered according to the configured online requirements and only once. Reconnect again without submitting another vote to confirm there is no repeat payout. Keep the original event timestamp separate from the later delivery timestamp.

If the test fails

If the event is accepted but the result is missing, look for a failed economy hook, inventory limitation or unmet requirement. A normal vote count increase does not prove every reward action succeeded. Avoid repeatedly using administrative vote commands as a substitute for inspecting the failed action.

Restore a changed reward requirement if it disrupts other players, while retaining queued state. Reconcile a confirmed missing payout against recorded delivery evidence. Do not delete a player record or restore an old database over newer votes to fix one reward.

Sources and related help

- VotingPlugin maintainer reference 1
- VotingPlugin maintainer reference 2
- Create and test Minecraft vote rewards with VotingPlugin

Review the command privileges in a VotingPlugin reward

A reward command can run with more authority than the player who earned it. A typo or overly broad permission command can therefore affect the whole server. Review the exact command and substituted player before enabling a new reward for real votes.

Before you change settings

Make a copy of the specific reward definition and identify where it is referenced. VotingPlugin supports inline sections and reusable files; editing a shared file can change several sites or milestones. Use a harmless message reward for the first wiring check.

Follow the request through the plugin

1. Locate the active definition rather than generated files under `Rewards/DirectlyDefined`. The maintainer advises editing the original inline configuration, since generated queue files are not the source of truth.
2. Check whether each action executes from the console or a player context using the installed version’s reward reference. Use the target player placeholder only where supported; do not assume every output path expands PlaceholderAPI.
3. Review the substituted command for scope. Prefer a small item, currency or cosmetic action over operator access or a broad permission grant. Remove ambiguous commands that could target everyone when an argument is absent.
4. Test the harmless definition with an ordinary account, then introduce one reviewed action at a time. Check both online processing and a controlled offline delivery if that reward can be queued.

Confirm the intended result

The intended player should receive only the configured benefit and no new administrative access. Confirm a second player’s balance, permissions and inventory remain unchanged. If several sites share a reward file, test one representative path from each affected caller.

If the test fails

A file-based reward begins with its reward keys rather than an unnecessary extra wrapper. An inline section and a file reference have different structure, so preserve the documented format. A message working does not prove a command ran with the expected context.

Restore the copied source definition and disable only the new reward action if its scope is wrong. Preserve vote totals and queued evidence. If authority was accidentally granted, remove that specific grant and review logs before resuming the reward.

Sources and related help

- VotingPlugin maintainer reference 1
- VotingPlugin maintainer reference 2
- Create and test Minecraft vote rewards with VotingPlugin

Distinguish a VotingPlugin vote party from a personal milestone

A vote party measures shared progress, while a personal milestone measures one player’s totals or activity. Both can legitimately trigger on the same vote. Identify each reward source before assuming a doubled message means the listener received a duplicate.

Before you change settings

Save SpecialRewards.yml and record the installed release. The maintainer documents modern milestone and streak systems alongside legacy compatibility sections. Note the party target, eligibility policy and the player’s relevant personal total before submitting a controlled test.

Follow the request through the plugin

1. Locate the party section and identify global actions versus per-player rewards. A global console action should run once for the party; a per-player action can legitimately run for several eligible players.
2. Check whether eligibility is limited to voters, online players or another documented condition. Compare that policy with the player who says they missed the party reward.
3. Inspect personal milestone and legacy reward sections for an equivalent bonus. When migrating to a modern milestone, do not leave an old definition active unless you deliberately want both.
4. On a proxy network, identify which component owns party progress and reward forwarding. Test near the threshold with controlled accounts, recording the trigger and every intended recipient.

Confirm the intended result

The party should trigger at the configured shared target, and individual bonuses should follow their own conditions. Check the next vote after the party to confirm progress continues according to the configured reset or increasing-target behaviour.

If the test fails

If the same party executes on several backends, inspect forwarding ownership and shared-state configuration. If only one player misses a reward, inspect eligibility and online delivery first. Neither symptom justifies resetting everyone’s vote totals.

Restore the previous reward definition if the policy changes unexpectedly. Keep progress and delivery logs intact. Reconcile any confirmed missing benefit with the owner, rather than replaying the entire party while some recipients have already been rewarded.

Sources and related help

- VotingPlugin maintainer reference 1
- VotingPlugin maintainer reference 2
- Create and test Minecraft vote rewards with VotingPlugin

Reduce repeated vote reminders without disabling voting

Vote reminders can be triggered at login and at intervals. A short per-rule cooldown does not necessarily describe the whole reminder policy, because global cooldowns, priorities and eligibility conditions also affect which rule runs.

Before you change settings

Check the installed version and its generated configuration before using this procedure. The maintainer documents a redesigned reminder system and a legacy migration; an older release may use different settings. Save the current configuration, especially before allowing an automatic migration.

Follow the request through the plugin

1. List enabled reminder rules and their trigger types. Compare login and interval rules so two nearly identical prompts are not unintentionally scheduled for the same player.
2. For builds using the redesigned system, inspect `VoteReminderOptions`, global cooldown and `StopAfterMatch`. A higher-priority matching rule can prevent a later rule from being evaluated.
3. Check each rule’s cooldown, delay and eligibility conditions against the player’s actual vote availability. For interval rules, the documented effective cooldown is the larger of interval and cooldown.
4. Test with an ordinary player who can vote, then one who has already voted. Verify the required reminder permission and observe a full configured interval instead of repeatedly reconnecting and changing several timers together.

Confirm the intended result

The eligible player should receive a useful prompt at the intended event, while an ineligible player should not be spammed. Check that ordinary voting and rewards still work; changing reminder timing should not require disabling the receiver or vote sites.

If the test fails

If no reminder appears, inspect conditions and permission access before shortening every timer. If too many appear, compare all matching rules and other reminder plugins. A queued login delay can also make an old session look like a new interval message.

Restore the saved rule configuration if prompts become confusing. Keep migrated and original copies privately, since migration may not create a backup itself. Do not clear persistent player cooldowns merely to make a test message appear immediately.

Sources and related help

- VotingPlugin maintainer reference 1
- Create and test Minecraft vote rewards with VotingPlugin

Preserve monthly voting history before changing leaderboard tracking

A monthly leaderboard reset and permanent all-time history are different records. If you want previous months available later, decide how that history will be retained before the next reset. An empty historical leaderboard cannot necessarily be reconstructed from the current month alone.

Before you change settings

Record the installed build, database type, timezone and current month totals. Take a consistent backup of plugin data and its database before changing tracking behaviour. On a network, include the proxy and backend configuration so their understanding of shared data remains consistent.

Follow the request through the plugin

1. Check the maintainer’s Month Date Totals reference against the installed release. It describes an optional date-labelled storage system, including features marked experimental; do not assume a development option is stable in every release.
2. Distinguish storing dated monthly history from making it the primary active total tracker. The documented primary-tracker option depends on dated storage being enabled and represents a separate behaviour change.
3. Choose the smallest supported change needed for your history requirement. Keep all-time totals and reward thresholds unchanged, and align corresponding proxy settings if the documentation requires shared configuration.
4. Test on an isolated copy using the same build and data. Compare current totals and the documented previous-month display before planning a production change; do not alter the live system clock to simulate month end.

Confirm the intended result

The current leaderboard should retain its values, and supported historical views should expose the months actually stored. Record the first month from which history is available. Enabling storage today does not promise recovery of months already overwritten under the former policy.

If the test fails

If proxy and backend results disagree, inspect their shared-data configuration and plugin versions before resetting anything. A top-voter award also needs its own enabled period and reward settings; a displayed leaderboard does not prove award delivery.

If the trial changes totals unexpectedly, restore the isolated copy and review the option with the maintainer before touching production. Any production rollback must use a consistent data and database pair and account for votes accepted since the backup.

Sources and related help

- VotingPlugin maintainer reference 1
- VotingPlugin maintainer reference 2
- Create and test Minecraft vote rewards with VotingPlugin