Diagnose Unturned Workshop and plugin failures after updates

If Unturned worked before an update, separate content acquisition, world loading and player joining. A stalled Workshop download needs different evidence from a plugin exception. Start with the earliest failure and avoid deleting caches or saves as a blanket repair.

Preserve evidence and identify the change

1. **Record the affected stage.** Does the process stop before the map loads, reach its ready state but reject players, or fail only when a plugin command runs? Keep the complete startup log and a client's exact message. Note the last successful session and any game, Workshop or plugin changes.
2. **Check the actual instance.** Confirm the startup instance ID and its active data directory. An unexpected new map or empty inventory may mean a different instance started. Preserve the original directory and stop investigating on the live world if the identity is uncertain.
3. **Create a recovery copy.** Save and stop normally when possible. Copy the instance data, configuration, module/plugin files and databases before changing anything. Retain the failed state separately from your last working backup.

For Workshop download or asset errors

Inspect the active instance's `WorkshopDownloadConfig.json` for valid JSON, correct numeric IDs and unintended changes. Check each item's availability and required dependencies on its author's page. Record the failing item ID and the download error; a removed or inaccessible item is different

from an invalid map name.

The official hosting guide explains that configured Workshop content is installed or updated during startup. Compare startup completion with the client's download or asset error before changing the firewall. Check available disk space and file permissions if output indicates write failures. Do not clear the entire Workshop tree or unsubscribe players from everything.

For map problems, compare the configured map name with the selected item's instructions. For one affected client, compare its game branch and specific content error with a working client. If a targeted client download repair is required, preserve local files first and follow the item's or Steam's relevant instructions rather than guessing which saved-data directories are disposable.

For plugin or module exceptions

Record the framework, plugin and dependency versions. SDG's maintained LDM fork is the supported Rocket-compatible starting point; an archived original Rocket build is not an equivalent update source. Use the module supplied with the matching dedicated-server release, and check each plugin author's compatibility notes.

Do not repeatedly hot-reload a failing stack. LDM's upstream changelog records disabling the reload-all command because many plugins did not support it correctly. Prefer a controlled stop/start for a tested package change. Never paste database passwords or connection strings into a public error report.

Test a reversible correction

Reproduce the problem on an isolated copy. Reverse the most recent change with its required dependencies and test the affected action again. Keep dependency-complete groups together when narrowing a conflict. Use a disposable world for content-removal experiments; do not save the only live world after removing assets it needs.

Success means the expected map and packages load, a matching client joins, existing progress is present and the feature still works after saving and restarting. For rollback, use a compatible game/module/plugin set and its appropriate backed-up data together. Keep logs and the package inventory for support if a publisher update prevents a reliable rollback.

Related: Unturned joining checks and startup troubleshooting.

Revision #1
Created 2026-11-10 11:05:47 UTC by CloudExa Hosting Admin
Updated 2026-11-10 11:05:47 UTC by CloudExa Hosting Admin