Source Games

Counter-Strike 2, Garry's Mod and Team Fortress 2 server setup and troubleshooting.

- Configure a CS2 server and protect its Steam login token
- Configure a TF2 server and use RCON safely
- Diagnose CS2 startup and update failures without losing data
- Find the addon behind a Garry’s Mod Lua error
- Fix missing Garry’s Mod textures and client downloads
- Install or update Metamod and CounterStrikeSharp for CS2
- Install ULX and ULib and give staff limited permissions
- Load and troubleshoot CS2 Workshop maps
- Set up CS2-SimpleAdmin with limited staff permissions
- Set up Garry’s Mod Workshop collections and gamemodes

Configure a CS2 server and protect its Steam login token

Counter-Strike 2 uses Source 2. Configuration advice for the old CS:GO server can contain obsolete paths and commands. Start from the files belonging to your installed CS2 build, and change a small number of settings at a time. You do not need to reinstall the server to give it a name or make it private.

Save a baseline first

Download a backup of your configuration and record the current startup settings before editing anything. Keep token-bearing files private. Schedule a short maintenance window if a restart is needed, and tell connected players before stopping the service normally.

The usual dedicated-server configuration location is `game/csgo/cfg/server.cfg`, relative to the CS2 installation. A file browser may already open inside `game/csgo`; check the existing folders before creating another nested copy. On Linux, names are case-sensitive. Ensure your editor has not saved the filename as `server.cfg.txt`.

Choose settings for their actual purpose

- `hostname` controls the displayed server name.
- `sv_password` is a player admission password. It is different from administrative access.
- `sv_lan` distinguishes LAN operation; an internet-hosted server normally uses `0`.
- A Steam game server login token identifies the dedicated server to Steam. It is neither a player password nor a Steam Web API key.

Use quoted values when they contain spaces. Keep a brief comment beside each intentional change so another administrator can understand it later. Do not paste an entire competitive configuration into a casual or custom-mode server: mode files and startup parameters can override individual values.

Handle the Steam token privately

Use Steam's official game server account management page for your own account and the correct game application, CS2 App ID `730`. Follow its current eligibility requirements. Where your service exposes a dedicated token field, use it. Otherwise ask support how the startup template supplies `sv_setsteamaccount`; do not add a second conflicting value blindly.

Use a separate token for each running server. Never post a token in a public ticket, Git repository, screenshot or shared startup example. If it has leaked, replace it through Steam and update the server's private configuration. Redact it from diagnostic logs before sharing them. A token cannot repair an incorrect allocation, broken map or incompatible plugin.

Verify without sharing secrets

Start the service and watch its complete startup output. Confirm the intended map loads and that Steam authentication succeeds. Query a harmless setting such as `hostname` in the server console, then join from your game client using the service's assigned address and port. Do not query or screenshot password variables as a test.

If a setting changes back, inspect later executed configuration files and startup values rather than duplicating the setting repeatedly. Test persistence through a scheduled restart and a map change. If authentication fails, send support the redacted error, build information and timestamp. Keep your saved baseline so the last configuration change can be reversed without replacing game files.

Related guides

- Diagnose CS2 startup failures
- Set up CS2 Workshop maps

Further reading

- Valve: Counter-Strike 2 dedicated servers
- Steam: game server account management

Looking for another game? Browse game server setup and troubleshooting guides.

Configure a TF2 server and use RCON safely

Team Fortress 2 uses Source 1. Its configuration layout and plugin ecosystem differ from CS2. RCON provides remote server-console authority, so treat its password as an administrative credential rather than another way for players to join.

Start with the configuration file

Back up `tf/cfg/server.cfg` and your startup settings before editing. That path is relative to the TF2 installation; a hosted file manager may already open inside `tf/`. Preserve existing ban files, custom maps, plugins and configuration. A configuration change does not require deleting or reinstalling the game.

Use `hostname` for the public server name. A nonempty `sv_password` restricts player admission, while `rcon_password` controls remote console authentication. These should never share a password. Keep public-server settings separate from secrets so a harmless configuration example can be shared safely.

Record the current startup map and allocated game port. Do not replace a working hosted launch template with a home-server example. TF2 settings, custom map scripts and plugins can execute additional configuration later, so investigate overrides if a value does not persist.

Choose the safer administration route

For routine work, use the hosting panel's authenticated console where available and give teammates their own restricted panel accounts. RCON is appropriate only when you actually need a supported remote administration tool. It is not necessary to expose it to everyone to make the server playable.

If RCON is needed, generate a long, unique password in a password manager and store it privately. Arrange source-address restrictions or a private management route with support. Source RCON is an administrative protocol, not an encrypted secret-sharing channel; do not assume the game

connection protects a password sent by an external tool.

Use only the assigned TCP endpoint approved for your service. Do not open arbitrary host firewall ports, alter your home router or ask for unrestricted public access. If the service cannot offer an appropriately restricted route, use its panel console instead.

Connect and test carefully

A TF2 client console uses `rcon_address HOST:PORT` to select a remote server and `rcon_password` to provide the secret. Replace placeholders privately; do not copy a real command containing the password into a ticket, stream overlay or shared configuration repository.

Start with a harmless command such as `rcon status`. Check that the response belongs to your intended server before running anything disruptive. Avoid sharing its output publicly because it may contain player information. A timeout and an authentication rejection are different failures: the first calls for endpoint and access checks, while the second calls for credential and target checks.

Keep access auditable

Never fix failed authentication by removing the password. Stop testing after repeated failures and ask support to check the permitted management source and server output. Do not bypass bans or defensive controls to continue guessing.

After a staff departure or suspected leak, rotate the secret and remove it from authorised tools. Restart during maintenance if needed, verify the new credential and confirm the old one no longer works. Keep configuration backups private and retain a record of who is allowed administrative access.

Related guides

- Back up game files before changes
- Invite a teammate to your game server

Further reading

- Valve: TF2 dedicated servers
- Valve: Source RCON protocol
- Valve: Source SDK 2013

Looking for another game? Browse game server setup and troubleshooting guides.

Diagnose CS2 startup and update failures without losing data

A server that exits after an update may have a download problem, an incompatible extension or a bad startup setting. Reinstalling everything hides the distinction and risks losing configuration. Start by finding the earliest meaningful error in one complete attempt.

Freeze the evidence

Stop repeated automatic restarts where your service allows it, or ask support to pause the loop. Save the complete startup output, time of the failure, last successful start and exact changes made. Download a backup of configuration, plugins, custom content and any plugin databases before testing recovery.

Record the allocated address and port without changing them. Save a redacted copy of the startup command. Remove Steam login tokens, passwords and database credentials from anything sent to another person. Do not expose secrets simply to prove that an argument exists.

Identify which stage failed

- **Update/download:** SteamCMD has not completed, reports access problems, or cannot write files. Check available storage and the first failed operation.
- **Engine launch:** the process exits before loading a map. Check the selected executable, operating system and required libraries with support.
- **Extension loading:** Metamod or CounterStrikeSharp reports binary, gamedata or API incompatibility. Compare the installed versions with current maintainer notes.
- **Map loading:** the engine runs but rejects the requested map or Workshop item. Test the previously working standard map.
- **Reachability:** the map is running, but clients cannot connect. Check the assigned allocation and server logs separately from installation.

Restore a minimal known configuration

CS2's dedicated application is App ID `730`. Its Linux executable is normally `game/bin/linuxsteamrt64/cs2`; Windows uses `game/bin/win64/cs2.exe`. A hosted startup template may wrap these commands. Do not replace that wrapper or paste a Windows batch file into a Linux service.

Keep the required `-dedicated` launch mode and your assigned `-port` value. Preserve provider-required arguments. Ask support to compare the effective command if the panel does not expose it. Changing ports to a random example will not open that port for a hosted service.

If the failure began with an extension update, test a controlled framework-free baseline only after backing up and arranging downtime. Temporarily disabling a loader or moving its files outside the active load path should be documented and reversible. Do not delete the addons directory, plugin storage, bans or database tables. A base-game success narrows the fault; it does not mean your community's configuration can be discarded.

Use validation cautiously

SteamCMD validation can restore game-managed files, including files you edited. It is not a configuration backup. Ask support before running it if your service manages updates automatically, and preserve `gameinfo.gi` and custom settings first. Avoid repeated overlapping update jobs.

After the specific fault is corrected, start once, join, check the intended map and confirm required plugins work. Record the compatible build set for the next update. If escalation is needed, provide the redacted first error, build numbers, recent changes and baseline test result so support can investigate without wiping your installation.

Related guides

- Check CS2 plugin framework compatibility
- Troubleshoot CS2 Workshop maps

Further reading

- Valve: CS2 dedicated servers
- Valve: SteamCMD
- CounterStrikeSharp: release compatibility information

Looking for another game? Browse game server setup and troubleshooting guides.

Find the addon behind a Garry’s Mod Lua error

A Lua error is a clue about a failed script, not a reason to erase the server. It may affect one menu or prevent an entire gamemode from initializing. Your aim is to reproduce the fault, identify the first failing component and test a reversible change.

Capture the first error

Save the full error block, including its file path, line number and stack trace. Record what happened immediately beforehand: startup, player join, a weapon action or a menu click. Note the active map, gamemode and recently changed addons.

Read earlier messages too. A later nil-value error may only be the consequence of a dependency that failed first. Keeping just the last line can send the investigation toward the wrong addon. Redact player details, database credentials, webhook URLs and tokens before posting logs publicly.

Find the relevant Lua realm

Garry's Mod has separate server, client and menu Lua contexts. A client-side function is not automatically available to a server script. Use the error's source and the location where it appears to decide which log you need. The dedicated console cannot show every local client failure.

Ask an affected player for their complete client error if the symptom is a broken HUD or menu. Compare with another consenting test player. If only one person is affected, investigate their addon set and downloaded content before changing the whole community's server configuration.

Read the trace without guessing a code fix

- A file path under a named addon provides a starting point, but that addon may rely on another library.
- A nil function or value can mean a missing dependency, wrong initialization order or an outdated API.
- A syntax error after a configuration edit often points to a missing quote, bracket or separator near the reported location.
- A missing include can mean an incorrect path, filename case or incomplete package.

Do not rename functions or suppress the error message simply to make the console quiet. That can conceal a broken economy, permission system or persistence feature. Compare with the author's supported release and configuration format instead.

Isolate safely

Back up configuration, `garrysmod/data/` and any addon database before testing. Use a separate test service where possible. Otherwise arrange maintenance, stop normally and save a copy of the current collection and local addon list.

Start with the most recent optional addon change and its documented dependencies. Disable a local addon reversibly by moving its folder outside the active `garrysmod/addons/` tree, retaining the original path in your notes. For Workshop content, use a temporary test collection. Do not delete persistent data or remove a required gamemode framework mid-test.

If many optional addons are involved, test groups, narrowing the failing group each time. Keep a known working dependency set. Restore the previous selection after each unsuccessful hypothesis rather than accumulating unrelated changes.

Confirm and report

Repeat the exact triggering action after the correction. Check both client and server output, then verify persistence through a controlled restart. Send the addon author a minimal reproduction, versions and redacted trace if a supported combination still fails. Keep the backup until normal gameplay and saved data have both been confirmed.

Related guides

- Review Workshop collections and dependencies
- Check ULX staff permissions

Further reading

- Facepunch: Lua error explanation
- Facepunch: Lua states and realms
- Facepunch: Lua error information

Looking for another game? Browse game server setup and troubleshooting guides.

Fix missing Garry’s Mod textures and client downloads

Pink-and-black surfaces and ERROR models usually mean a client is missing required assets. A server can run an addon successfully while players cannot see its models. Separate server mounting, client download instructions and dependencies on other games before changing anything.

Identify exactly what is missing

Record the map, affected object or texture, and relevant console error. Ask whether every player sees the problem or only one. Test with a client that does not already have your whole addon collection installed. The owner's computer often hides an incomplete content setup.

Read the map or addon's official dependency list. Some content requires another game's assets. Mounting those assets on the dedicated server does not automatically license or install them for every client. Follow the publisher's legitimate requirements; do not distribute copyrighted game packages or use an unofficial content bundle as a shortcut.

Prefer the correct Workshop item

For Workshop-delivered models, materials or sounds, confirm that the required item is installed on the server where needed and explicitly requested for clients. The function `resource.AddWorkshop("ITEM_ID")` registers a client download; it does not mount the addon server-side.

A server-side declaration can live in `garrysmod/lua/autorun/server/workshop.lua`. Back up an existing file before editing it. Replace the placeholder with the actual content item's numeric ID, preserve valid Lua syntax and follow the author's instructions about dependencies. A collection ID is not a substitute for each content item expected by this function.

Avoid adding server-only Lua packages unnecessarily. Keep the client download list small and purposeful, and verify the Workshop items remain available. A missing or incompatible item needs correcting at its source rather than masking the result with another copy of the same assets.

Use FastDL for files that need it

FastDL serves registered files from a separate web endpoint configured by `sv_downloadurl`. It needs the same relative asset paths clients request. Register required content through documented functions such as `resource.AddFile`, then verify the matching HTTP resource exists.

For example, a requested `materials/example/panel.vmt` must be available beneath that relative path on the download site. Facepunch documents clients trying a `.bz2` version before the ordinary file. Preserve exact filename case, particularly when the web host runs Linux.

Never point a public download service at the entire game-server directory. Publish only intended game assets, excluding configuration, databases, backups and credentials. Ask support for an approved FastDL endpoint if your hosting plan does not include one. A file-manager folder is not automatically a public web address.

Test downloads, then gameplay

Back up configuration before changing download declarations. Restart in maintenance and reconnect with your test client. Check the client's download settings and console for failed requests, inaccessible items or path mismatches. Compare a direct asset request with the path actually reported by the client.

Do not ask everyone to erase all downloaded content as your first fix. First prove the server points at the right resource. Correct one asset family, then check models, textures and sounds in gameplay. Retain your original settings until a clean client can join and see the intended content.

Related guides

- Set up Workshop collections
- Read Garry’s Mod client and server errors

Further reading

- Facepunch: serving content
- Facepunch: client Workshop downloads
- Facepunch: registering downloadable files
- Facepunch: mounting game content

Looking for another game? Browse game server setup and troubleshooting guides.

Install or update Metamod and CounterStrikeSharp for CS2

CounterStrikeSharp is a CS2 plugin framework loaded through Metamod. It is not a drop-in replacement for every CS:GO SourceMod plugin. A SourceMod `.smx` file cannot simply become a CounterStrikeSharp plugin by changing its extension.

Choose a compatible set

Record the CS2 build, host operating system, existing framework versions and plugin list. Read the current CounterStrikeSharp installation guide and release notes alongside Metamod's CS2 requirements. Requirements change after game updates; a version that worked last month is not a promise of compatibility today.

Use official release packages matching the server's operating system and architecture. Windows binaries are unsuitable for a Linux service. For a first CounterStrikeSharp installation, its documented bundled-runtime package avoids relying on an unknown existing .NET runtime. Ask support about missing system libraries; hosted game accounts generally cannot install operating-system packages.

Back up and place the loaders

Stop the server normally after notifying players. Save copies of `game/csgo/gameinfo.gi`, your current `addons` directory and relevant configuration. Include plugin databases or external database exports where applicable. Store the backup outside the paths you are about to update.

Extract Metamod so its `addons` directory sits inside `game/csgo/`. Check that you did not create `addons/addons` or install it beneath the old CS:GO directory. Follow the current loader instructions when adding its search-path entry to `gameinfo.gi`; preserve the rest of that file.

For the currently documented CS2 layout, the search-path entry is `Game csgo/addons/metamod`, placed below `Game_LowViolence csgo_lv`. If your file or release instructions differ, stop and compare the actual supported build rather than reconstructing the whole file from an online example.

Check each layer separately

Start the server and run `meta list` in its console. An unknown command means the Metamod loading stage needs attention. A working command with no CounterStrikeSharp entry means the next loader has not yet been confirmed.

Stop again before extracting CounterStrikeSharp's release `addons` content into `game/csgo/`. Restart, check `meta list` again and inspect framework startup logs. Only install one small, trusted plugin after the framework itself loads successfully. A downloaded repository source ZIP is not necessarily a ready-to-run release.

Update with a rollback point

Update during maintenance, preserving existing configuration and plugin data. Check release-specific migration instructions before replacing binaries; third-party plugins may require their own compatible updates. Do not overwrite locally edited settings with a sample configuration just because it came in an archive.

After a game update, recheck the loader entry because game-managed files can change. Test a client join and the plugin's actual function, then perform a controlled restart. If a regression appears, retain the error and version details and restore the saved compatible configuration and binaries only after checking database migration implications. Avoid an automatic downgrade across an irreversible plugin database change.

Related guides

- Set up scoped CS2 administration
- Diagnose CS2 startup failures

Further reading

- CounterStrikeSharp: getting started

- Metamod: official development downloads
- CounterStrikeSharp: official release notes

Looking for another game? Browse game server setup and troubleshooting guides.

Install ULX and ULib and give staff limited permissions

ULX is an administration addon for Garry's Mod, supported by the ULib library. It is separate from CounterStrikeSharp administration for CS2 and from SourceMod. Install the Garry's Mod packages and give staff individual Steam-based identities rather than sharing your hosting account.

Prepare the installation

Back up your addon list, configuration and existing administration data before changing versions. Record current staff groups and permissions, including any custom restrictions. Schedule maintenance so a complete restart does not surprise players.

Choose one installation method. The project's official Workshop items are ULX `557962280` and ULib `557962238`. Add both to the server's intended collection. Alternatively, use the authors' downloadable packages and place their extracted addon folders under `garrysmod/addons/`. Check that the files are not hidden inside an extra archive-name directory.

Avoid installing the same addon through Workshop and local files simultaneously. Duplicate copies make it unclear which version is running. If replacing an existing installation, retain the old copy outside the load path until the new version is verified. Preserve its data rather than treating an update as a fresh server.

Load and bootstrap access

Perform a full server restart; a map change is not enough for the documented installation procedure. Check the earliest startup errors if ULib or ULX fails to initialize. Confirm `ulx help` responds in the server console before trying to assign staff.

Connect your own verified Steam account. From the trusted server console, use `ulx adduser "EXACT_CONNECTED_NAME" superadmin`, replacing the placeholder with your connected player name. Confirm the target carefully when names are similar. Use the console's help for ID-based targeting if names are ambiguous; do not promote an approximate match.

The superadmin group is powerful. Use it only for the owner and explicitly trusted administrators. Once owner access works, open `ulx menu` from your game console and review the Groups controls. Keep game-panel account access separate from in-game moderation.

Create a practical moderator role

Create or review a lower-privilege group and permit only its required moderation actions. Start with a short list such as warnings, temporary communication restrictions and kicks, as supported by your installed command modules. Review inherited permissions as well as direct grants. A group that inherits a powerful parent may retain access you never selected.

Check targeting and immunity behaviour with two consenting test accounts. A moderator should not be able to alter the owner's group, execute arbitrary RCON commands or obtain unrestricted staff-management powers. Use the menu and documented commands rather than editing permission storage while the server is running.

Verify both sides of the boundary

Reconnect as the moderator, test one permitted harmless action and confirm a restricted action is denied. Test an ordinary player too. Recheck after a map change and scheduled restart so temporary permissions are not mistaken for saved configuration.

Retain an audit record of staff changes and remove access promptly when someone leaves. If permissions behave unexpectedly, collect the installed versions, relevant group inheritance and a redacted reproduction. Never upload your entire player or ban database to a public support thread.

Related guides

- Configure a Workshop collection
- Diagnose addon startup errors

Further reading

- Team Ulysses: ULX installation and usage
- Team Ulysses: ULib

- Team Ulysses: user and group command definitions

Looking for another game? Browse game server setup and troubleshooting guides.

Load and troubleshoot CS2 Workshop maps

A Workshop page existing does not prove that its map works with today's CS2 server. CS2 uses Source 2, and older CS:GO content may still appear in searches. Check the item's game, compatibility information, author notes and recent change history before adding it to a public rotation.

Record a working starting point

Back up your configuration, record the current map and save your startup settings privately. First confirm a standard installed map, such as `de_dust2`, can load and accept a player. A Workshop test is much easier to interpret when networking and the base game already work.

Arrange a maintenance window: loading another map disconnects or interrupts players. Use the hosted server console, rather than running commands on your own client's listen server by mistake. You do not need root access to perform these configuration checks.

Test one item before building a rotation

1. Open the map's actual Workshop item page. Copy the numeric value after `id=` in its URL. A collection ID and a map item ID are different objects.
2. Review whether the item is public and whether its author lists dependencies or special mode settings.
3. In the dedicated-server console, use `host_workshop_map MAP_ITEM_ID`, replacing the placeholder with that map's numeric ID.
4. Watch for download progress, completion and the subsequent map load. Large content can take time; repeated restarts interrupt useful evidence.
5. Join with a normal client, check that the expected map is active and complete a basic round or spawn test.

For startup, the equivalent command form is `+host_workshop_map MAP_ITEM_ID`. Only change the startup template if your service permits it, and preserve the existing required arguments. A console command and a shell launch argument are not interchangeable: the plus sign belongs to the launch form.

Keep rotation support separate

Successfully loading one item does not automatically create a Workshop rotation. Rotation behaviour depends on the current game mode and any installed map-management plugin. Use the selected manager's documented CS2 format, not an old CS:GO mapgroup recipe or Garry's Mod's collection parameter. Save its configuration before adding entries.

Keep a small record of item ID, readable name, author and test date. Test new maps individually before making them the boot map. Avoid changing the map, mode, plugin framework and network allocation in the same maintenance session; that makes failures difficult to attribute.

Find the failing stage

A download error suggests item access, Steam availability, space or connectivity. A completed download followed by a map-load error suggests content or build compatibility. A running map that clients cannot join needs separate client and server logs. Preserve the first relevant error and its timestamp instead of relying only on the final exit message.

Return to the recorded standard map if the new item fails. Do not erase Workshop caches, validate files or reinstall as the first response. Send support the item URL, redacted startup output and confirmation that the baseline map works. Never include Steam login tokens in the report.

Related guides

- Configure CS2 and protect its token
- Diagnose CS2 startup failures

Further reading

- Steam: Counter-Strike 2 Workshop
- Valve: CS2 dedicated servers

Looking for another game? Browse game server setup and troubleshooting guides.

Set up CS2-SimpleAdmin with limited staff permissions

CS2-SimpleAdmin runs on CounterStrikeSharp. It provides player administration, but the plugin loading successfully is only the first check: the correct person must receive the intended permissions, and an ordinary player must remain unable to use them.

Prepare a reversible installation

Back up the existing plugin folders, configuration and administration database before changing anything. Record your CS2 and CounterStrikeSharp versions. Read the plugin release's requirements, including its current menu dependencies; do not assume an older installation has every dependency a newer release expects.

The current author documentation lists CounterStrikeSharp, AnyBaseLibCS2, PlayerSettings and MenuManagerCS2. Install compatible releases from their maintainers, using the exact package layout. Start and check each layer during maintenance. Avoid downloading an unrelated plugin with a similar name or copying compiled files from an unknown server.

Use the generated configuration

The plugin lives under `game/csgo/addons/counterstrikesharp/plugins/CS2-SimpleAdmin/`. The author's release may also supply shared API files; retain their intended location. After a successful first launch, its configuration is generated under `addons/counterstrikesharp/configs/plugins/CS2-SimpleAdmin/`, relative to `game/csgo/`.

Stop before editing generated JSON. Preserve its existing schema and validate commas, quotation marks and nesting. Use the exact options documented for the installed release rather than pasting a complete configuration from a different version.

Choose database storage deliberately. The project documents MySQL for shared deployments and also offers SQLite, which its main project description qualifies as experimental. Keep production requirements in mind. A MySQL account should be limited to this application's database. Use the database endpoint supplied for your service; `localhost` inside a container usually means that container, not the database machine.

Give staff only the actions they need

CounterStrikeSharp's framework reads administrator identities and flags from `addons/counterstrikesharp/configs/admins.json`. Identify staff by their verified Steam identity rather than display name. Its string permissions differ from old SourceMod letter flags. For example, `@css/kick` and `@css/changemap` describe separate capabilities.

Check SimpleAdmin's own permission and database administration behaviour for your release before choosing the source of truth. Avoid creating conflicting grants in several systems. Do not give every moderator `@css/root`: that framework flag grants all flags and bypasses immunity. Limit access to RCON, configuration changes, permanent bans and other staff management to the owner or a specifically trusted role.

Test allowed and denied actions

Restart in maintenance and inspect framework, dependency and database messages. The documented `css_admin` command is a useful menu check for an authorised player. A missing menu can indicate a dependency or disabled command, not necessarily a missing permission.

Test a harmless allowed action with a consenting player, then verify an unprivileged account is denied administrative access. Recheck after reconnecting and changing maps. Keep the action audit trail. Share only redacted logs with support: database passwords, Discord webhooks, player IPs and Steam identifiers do not belong in public screenshots.

Related guides

- Install Metamod and CounterStrikeSharp
- Configure CS2 and protect its token

Further reading

- CS2-SimpleAdmin: official project
- CS2-SimpleAdmin: installation
- CS2-SimpleAdmin: configuration
- CounterStrikeSharp: defining admins

Looking for another game? Browse game server setup and troubleshooting guides.

Set up Garry’s Mod Workshop collections and gamemodes

Garry's Mod uses Source 1. Its Workshop collection startup option is different from CS2's map command, and a collection does not automatically select a gamemode. Keep collection loading, gamemode selection and client content downloads as three separate checks.

Prepare a small collection

Back up your startup settings, configuration and addon data first. Record the last working map and gamemode. Make changes during a maintenance window, because a full restart is usually the clearest way to confirm what mounted at startup.

Create a collection containing the intended addons and their required dependencies. Make it public or unlisted so the dedicated server can access it. Record the collection's numeric ID from its Workshop URL. Check that you copied a collection ID rather than an individual addon ID, and review each item's required content and supported gamemode.

Start with only the essentials. A large collection makes first startup slower and gives you many possible causes when an addon fails. Keep a dated list of the items you have tested, especially when several staff members can edit the collection.

Mount content and select the mode

1. Set the service's collection value if its startup template provides one. Otherwise ask support to add `+host_workshop_collection COLLECTION_ID` to the managed startup configuration.
2. Select the gamemode using its internal folder name, such as `+gamemode sandbox`. A Workshop item number is not a gamemode name.

3. Use the installed map name without `.bsp`, for example `+map gm_construct`, while preserving your assigned network settings.
4. Restart normally and read the Workshop mounting output, gamemode initialization and map load messages.

A manually installed gamemode belongs under `garrysmod/gamemodes/`, with its expected internal directory structure. Avoid accidentally creating two nested copies of the same gamemode folder. Workshop-managed content should remain managed through the collection rather than being duplicated as extracted local addons.

Check what players receive

Mounting an addon on the server and telling clients to download its assets are different tasks. Facepunch's `resource.AddWorkshop` marks an item for client download; it does not install that item on the server. A server-side Lua file can declare the required client content item IDs.

Do not indiscriminately mark every collection entry for client download. A server-only Lua addon may not need to be downloaded as Workshop content. Check the author's instructions for models, materials, sounds and gamemode resources. The active Workshop map has additional automatic handling, but other assets still need checking.

Verify the complete experience

Join with a test client that has not already subscribed to your full collection. Confirm the correct mode is active, the map loads, required assets appear and basic gameplay works. A staff member with every addon installed can miss a broken download configuration.

If it fails, keep the first error and item ID. Return to the saved baseline, then introduce dependencies and addons in small groups. Preserve data folders and external databases throughout testing. Do not wipe Workshop caches, saves or all addons to fix a single incompatible item; narrow down the failing content and keep a reversible record of each change.

Related guides

- Fix missing Garry’s Mod content
- Isolate Garry’s Mod Lua errors

Further reading

- Facepunch: Workshop for dedicated servers
- Facepunch: dedicated server setup
- Facepunch: resource.AddWorkshop

Looking for another game? Browse game server setup and troubleshooting guides.