THE COUNTER-STRIKE BOT v1.5 - LIST & EXPLANATION OF COMMANDS

Contents

How to use commands

1. Press the ` key on your keyboard once, to the left of the `1` key.

2. In the Console window that appears, type a command into the text box.

   If you are adding arguments, do not type < or >.
   For example, to set **`bot_allow_grenades`** **`<0,1>`** to 0, you would enter
   `bot_allow_grenades 0` or **`bot_allow_grenades "0"`**
   but not **`bot_allow_grenades <0>`**

3. Click ***Submit*** or press `Enter` on your keyboard once.

Example valid command input: `bot_kill`

Example valid command input: `bot_allow_rogues 1`

Example valid command input: `bot_allow_rogues "1"`

Example invalid command input: `bot_allow_rogues"1"`

Example invalid command input: `bot_allow_rogues1`

Example invalid command input: `bot_allow_rogues <0>`

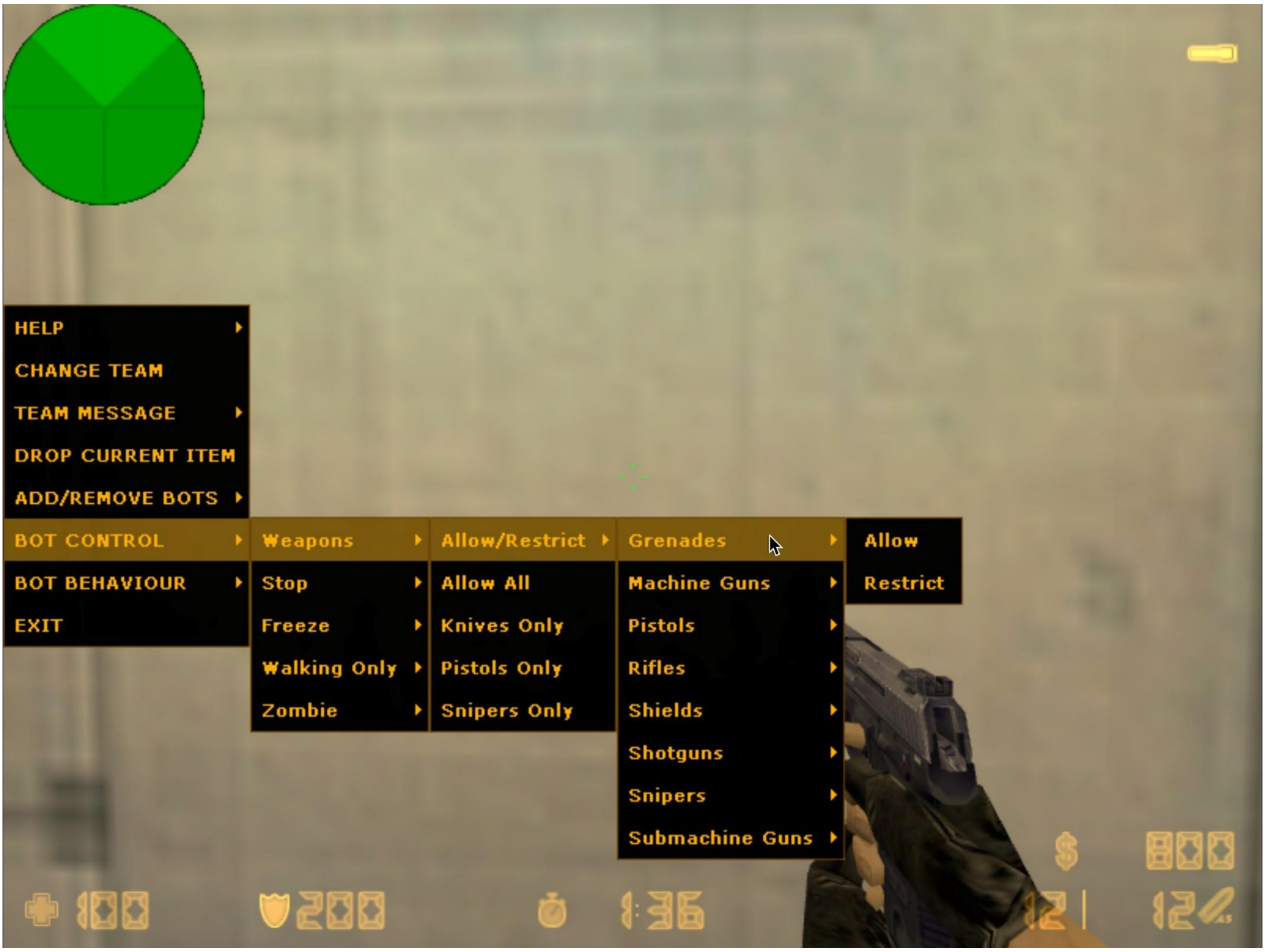
Example invalid command input: `bot_allow_rogues <0,1>`

Alternatively, you can also use the **command menu** to access the commands listed here.

Press H on your keyboard once to open the command menu. It will appear on the left of your screen.

Then, hover your mouse over the menus to expand the sub-menus. Once you have reached your desired option, left-click once to trigger it.

Ensure you keep your cursor within the menu. Otherwise, the expanded sub-menus will automatically retract (just like a normal context menu).



Bot Control

```
bot_add
bot_add_t
bot_add_ct
```

Adds a bot to the game.
"bot_add" on its own will add a bot to the team specified by the "bot_join_team" command.
"bot_add_t" and "bot_add_ct" adds a bot to the Terrorist or Counter-Terrorist teams respectively.

```
bot_allow_grenades <0,1>
bot_allow_machine_guns <0,1>
bot_allow_pistols <0,1>
bot_allow_rifles <0,1>
bot_allow_shield <0,1>
bot_allow_shotguns <0,1>
bot_allow_snipers <0,1>
bot_allow_sub_machine_guns <0,1>
```

Specifies whether bots can or cannot buy or use weapons within the specified weapon group.
0 = Bots cannot buy or use the given category of weapon.
1 = Bots can buy or use the given category of weapon (default for all except shields).

```
bot_allow_rogues <0,1>
```

Specifies whether bots may occasionally "go rogue."
Rogue bots just "run and gun" and will respond to all radio commands with "Negative."
0 = Bots will never "go rogue" (default).
1 = Bots may occasionally "go rogue.”

```
bot_all_weapons <0,1>
bot_knives_only <0,1>
bot_pistols_only <0,1>
bot_snipers_only <0,1>
```

Shortcuts that set the bot_allow_* commands to reflect the restriction state of the specified weapon group.
0 = Don't restrict bots to only using the specified weapon group (default).
1 = Restrict bots to only using the specified weapon group.

`bot_auto_vacate` `<0,1>`

Specifies whether bots will leave the game to allow human players to join if there are no available player slots in the game (i.e., the server is full).
0 - If a player attempts to join the game and no player slots are available, bots will not leave the game to allow space for human players to join.
1 - If a player attempts to join the game and no player slots are available, bots will leave the game to allow human players to join (default).

`bot_chatter` `<"off","minimal","standard","normal">`

Specifies which set of rules bots will adhere to when using the radio.
off = Bots will not use the radio.
minimal = Bots will use the radio minimally.
standard = Bots will only report standard radio callouts.
normal = Bots will use the radio normally (default).

`bot_defer_to_human` `<0,1>`

Specifies whether bots may perform objectives (e.g., defusing the bomb) if a human player is present in the game.
0 - Bots may perform objectives if a human player is present in the game (default).
1 - Bots will not perform objectives if a human player is present in the game.

`bot_difficulty` `<0-3>`

Specifies the difficulty of all newly created bots (existing bots will retain the difficulty setting they were created with).
0 = Easy.
1 = Normal (default).
2 = Hard.
3 = Expert.
Difficulty values above 3 will be reset to 3.

`bot_join_after_player` `<0,1>`

Specifies whether bots must wait to join the game until at least one human player has joined.
0 = Bots do not wait to join the game until at least one human player has joined.
1 = Bots will wait to join the game until at least one human player has joined (default).

bot_join_team <ct, t, any>

Specifies which team the bots will join.
ct = Counter-Terrorist.
t = Terrorist.
any = Any team (selected randomly, default).

bot_kick <name, "all">

Kicks the specified bot(s) present in the game.
Takes either the name of a single bot to kill or the keyword "all" (excluding quotes) - kicking all bots present in the game.
Executing the command with no arguments (on its own) will kick all bots present in the game.

bot_kill <name, "all">

Kills the specified bot(s) present in the game.
Takes either the name of a single bot to kill or the keyword "all" (excluding quotes) - killing all bots present in the game.
Executing the command with no arguments (on its own) will kill all bots present in the game.

bot_prefix <string>

Prefixes the given string to the usernames of all bots subsequently added to the game. This is useful for giving bots "clan tags."

bot_quota <minimum number of bots>

Specifies the minimum number of bots that must be present in the game (integer value).
Setting this command to a non-zero value will cause the given number of bots to be maintained in the game.
If a bot is kicked, a new bot will be added to maintain the quota. To disable the quota, set it to zero.

bot_quota_match <ratio>

Specifies the ratio of the number of bots per human player (i.e., number of bots = number of human players * bot quota match ratio).
As human players join or leave the server, the number of bots in the game will be adjusted accordingly.
This command is only available in Counter-Strike: Condition Zero, not Counter-Strike 1.6.
NOTE: that this command overrides the "bot_quota" command.

`bot_stop <0,1>`

Specifies whether bots may move or respond or not.
0 = Bots can move and respond (default).
1 = Bots stop moving and responding.

`bot_walk <0,1>`

Specifies whether bots can only walk or not.
0 = Bots are not restricted to walking only (default).
1 = Bots are restricted to walking only.

`bot_zombie <0,1>`

Specifies whether bots are in "zombie mode," causing them to idle and not fire their guns or attack.
0 - Bots are not in "zombie mode" (default).
1 - Bots are in "zombie mode."

`hostage_stop <0,1>`

Specifies whether hostages may move or respond or not.
0 = Hostages can move and respond (default).
1 = Hostages stop moving and responding.
This command is only available in Counter-Strike: Condition Zero, not Counter-Strike 1.6.

NOTE: *The following commands were not included in the original bot system.*
This ReGameDLL_CS modification added them.
Hence they are only available in Counter-Strike 1.6 and not Counter-Strike: Condition Zero.

`bot_deathmatch <0,1>`

Specifies whether bots can attempt game objectives (e.g., defusing the bomb)
0 - Bots may attempt to perform game objectives (unless bot_defer_to_human is set to 1 and a human player is present in the game, default).
1 - Bots will not attempt to perform game objectives.

`bot_freeze <0,1>`

Prevents bots from moving.
0 - Bots can move (default).
1 - Bots cannot move.

`bot_join_delay <delay>`

Setting this command to a non-zero value will cause bots to wait the given number of seconds before joining the game after a map change.

`bot_quota_mode <"normal","fill","match">`

Specifies the behaviour of the bot_quota command.
normal = Default behaviour (default).
fill = The game will automatically control bots to keep n players in the game, where n is the value specified by the bot_quota command.
match = The game will maintain a 1:n ratio of humans to bots, where n is the value specified by the bot_quota command.

Navigation Mesh Editing

Each of the following bot_nav_* commands operates on the navigation mesh, allowing hand-tuning of the automatically learned data.

It is recommended that these commands be bound to keys for ease of use while editing.

CAUTION: *There is no "undo" operation. Save your navigation mesh regularly and avoid rushing work.*

`bot_nav_edit <0,1>`

Setting this command to 1 allows hand-tuning of the current map's bot navigation mesh.
Once edit mode has been activated, the below bot_nav_* commands can be used.
0 = Disable navigation mesh editing mode (default).
1 = Enable navigation mesh editing mode.

`bot_nav_begin_area`
`bot_nav_end_area`

Creates new nav areas.
"bot_nav_begin_area" marks one corner of the new area to be created.
"bot_nav_end_area" marks the opposite corner of the new area to be created and creates it.
To cancel the operation, issue a "bot_nav_begin_area" command again.

bot_nav_connect

Creates a ONE-WAY link from the currently marked area to the currently selected area, telling the bots they can walk FROM the marked area TO the selected area.
For most areas, it is ideal to connect the areas in both directions.
However, it may be ideal for various height areas to only allow the bots to move one way - so they can jump down from a height if they reach it, but not jump back up.

bot_nav_corner_select
bot_nav_corner_raise
bot_nav_corner_lower

"bot_nav_corner_select" selects one of four corners of a currently marked navigation mesh which allows that corner to be raised or lowered.
"bot_nav_corner_raise" raises the corner.
"bot_nav_corner_lower" lowers the corner.
Issue the "bot_nav_corner_select" command multiple times to cycle between the four corners (and then once again to deselect all of the corners).

bot_nav_crouch

Flags the currently selected area as "crouch," requiring bots to crouch (duck) to move through it.

bot_nav_delete

Deletes the currently selected nav area.

bot_nav_disconnect

Disconnects ALL connections from the currently marked area to the currently selected area.

bot_nav_jump

Flags the currently selected area as "jump." This hints to bots that they should jump to traverse this area.
NOTE: Bots will ALWAYS perform a "crouch+jump" maneuver when traversing an area flagged as "jump".

bot_nav_mark

Marks the currently selected nav area for later operations.

`bot_nav_merge`

Merges the currently selected nav area and a previously marked nav area into a new, single nav area. The merge will only occur if the two areas are the same size along the merge line.

`bot_nav_no_jump`

Flags the currently selected area as "no jump." This hints to bots that they should not jump in this area.

`bot_nav_precise`

Flags the currently selected area as "precise", requiring bots to carefully traverse this navigation mesh before continuing onto the next one. Especially useful for small catwalks that are easy to fall from.

`bot_nav_splice`

Creates a new nav area between the currently marked area and the currently selected area and bi-directionally connects the new area.
Especially useful for creating sloped nav areas.

`bot_nav_split`

Splits the currently selected nav area into two new nav areas along a new white line.

`bot_nav_strip`

Strips all approach points, encounter spots, and hiding spots (generated using the "bot_nav_analyze" command) from the currently marked navigation mesh.

`bot_nav_warp`

Warps (teleports) your view to the currently marked navigation mesh. NOTE: You must be in the "Free Look" spectator mode.

`bot_nav_zdraw <height value>`

Specifies how high above the ground to draw the visual "nav mesh" guide when in nav edit mode (integer value).
If the terrain is very irregular or highly sloped, it can be useful to increase this value to 10 or 15. The default value is 4.

`bot_quicksave <0,1>`

Specifies whether the analysis phase of bot map learning will be skipped.
This is useful when iteratively hand-tuning nav files. Note that by skipping this analysis, the bots will not look around the world properly.
0 = Do not skip the analysis phase of bot map learning (default).
1 = Skip analysis phase of bot map learning.

Navigation Mesh Place Naming

Place names are used by bots to report locations over the radio, as well as location text strings for player radio-chat.
Remember to save the navigation mesh often ("bot_nav_save") just in case you make a mistake, and then analyze the navigation mesh again ("bot_nav_analyze").
"bot_nav_edit" must be set to 1 to enter the place name editing mode.

`bot_nav_toggle_place_mode`

Toggles place naming mode. Every navigation mesh currently highlighted with the crosshair will be displayed with one of three colors:
Green = Place name currently set.
Blue = Other place name already set.
Red = Place name not set.
Enter the command again to return to navigation mesh edit mode.

`bot_nav_use_place`

Prints a list of all usable place names in the console.

`bot_nav_use_place <place name>`

Selects a place name (e.g., bot_nav_use_place BombsiteA).

`bot_nav_toggle_place_painting`

Toggles painting navigation meshes with the selected place name.
Enter the command again to stop painting.
NOTE: Every navigation mesh currently highlighted with the crosshair have the selected place name applied to it.
NOTE: To change the selected place name, you must first enter this command again to disable the place name painting mode.

`bot_nav_place_floodfill`

Selects the place name from the currently highlighted navigation mesh and flood fills all other meshes outward from there, setting all adjacent navigation meshes to that place name until it encounters other navigation meshes with different place names.

`bot_nav_place_pick`

Selects the place name of the currently highlighted navigation mesh as your current place name.

`bot_nav_mark_unnamed`

Marks the closest unnamed navigation mesh and displays the total number of unnamed meshes.
Useful for locating unnamed navigation meshes, when used in conjunction with the "bot_zombie 1", "bot_goto_mark" and "bot_nav_warp".

Navigation Mesh Processing

`bot_nav_analyze`

Analyzes the navigation mesh to determine Approach Points and Encounter Spots.
This may take several minutes based on the size and complexity of the map.
NOTE: Requires a bot to be in the game. The recommended procedure is to save the mesh, add a bot, and quickly enter bot_analyze.

`bot_nav_load`

Clears the current navigation mesh and loads it from the saved file.

`bot_nav_save`

Saves the current navigation mesh to a .nav file with the same name as the current map.
For example, if the current map is de_dust.bsp, the .nav file will be named de_dust.nav.

Bot Debugging

`bot_debug <0,1,2>`

Used for internal debugging of bot behavior.
0 = Do not display any bot debug messages.
1 = Print debug messages for the bot you are currently spectating (you must be in the "First Person" or "Free Chase Cam" spectator modes) to the console
2 = Display debug messages for all bots.
Debug messages will be printed to the console indicating what the bot is currently thinking and doing, as well as any error or warning messages (including an audible warning sound) pertaining to the bot's behavior or navigation.

`bot_goto_mark`

Causes one bot in the map to move to the center of the currently marked area.
Useful for testing the walkability of specific portions of the navigation mesh.

`bot_memory_usage`

Prints memory usage statistics about the active bot module and the currently loaded navigation mesh file to the console.

`bot_nav_check_consistency <filename.nav>`

Checks the consistency and integrity of a bot navigation mesh file.

`bot_show_danger <0,1>`

Specifies whether a visual guide of "danger" in each nav area is shown as a vertical line.
Blue lines represent danger for Counter-Terrorists, and red lines are danger for Terrorists.
0 = "Danger" visual guides are not shown (default).
1 = "Danger" visual guides are shown (default).

`bot_show_nav <0,1>`

Specifies whether a visual guide of a small section of the nav-mesh around each bot is shown.
0 = A visual guide of a small section of the nav-mesh around each bot is not shown (default).
1 = A visual guide of a small section of the nav-mesh around each bot is shown.

`bot_traceview <value>`

Used for internal debugging of bot navigation.

`hostage_debug <0,1>`

Used for internal debugging of the hostage's navigation. The purple line indicates the direction that the hostage is looking, the green lines indicate the hostage's field of view, the orange line indicates the next navigation mesh that the hostage is heading towards, while the yellow line indicates the hostage's complete future path.
Prints debug messages that will be printed to the console indicating what the hostage is currently thinking and doing, as well as any error or warning messages pertaining to the hostage's behavior or navigation.
This command is only available in Counter-Strike: Condition Zero, not Counter-Strike 1.6.
NOTE: You should be playing as a Counter-Terrorist, and have already pressed the use key on a hostage (to instruct them to follow you) for best results.

Miscellaneous

`bot_about`

Prints the bot version number and information about the bot's author to the console.

`bot_profile_db <filename.db>`

Specifies the filename of the bot profile database file (default is BotProfile.db)
Do not touch unless you know what you are doing.