MCZ · MCP SERVERS

Unreal bridge manual

Unreal bridge 0.15.0 · 15 November 2026 · https://mcz.software/Mcp

Contents

1. How it works
2. Requirements
3. Install and uninstall
4. First use
5. Tools
6. Several editors and sessions
7. Settings
8. Licence
9. Troubleshooting
10. Logs

The Unreal bridge 0.15.0 lets Claude work in a running Unreal Engine 5.8 editor through 161 tools: Blueprint and material graphs, actors and levels, assets and import, project settings and input, UMG widgets, Play In Editor and builds. Several editors and several Claude sessions can share it; every call names the editor it is for. New to the MCP servers? Read Getting started first.

How it works

- An editor **plugin** (UnrealMcpBridge) runs inside each Unreal Editor and listens on this computer only, on port 9878 or the next free port up to 9899.
- Claude starts the server (`UnrealMcp.Server.exe`) for each session. It finds the running editors by checking those ports, and sends each call to the editor it names.
- Every change is an editor undo step, so you can undo what Claude did.

Requirements

- Windows, 64-bit.
- **Unreal Engine 5.8**. The plugin comes prebuilt for UE 5.8 Win64 and must match your engine build; setup warns if it does not.
- No administrator rights, no Visual Studio and no .NET install needed to use it (Visual Studio is needed only to build C++ projects).
- The plugin turns on Unreal's Python Editor Script Plugin and Enhanced Input automatically.

Install and uninstall

Close the Unreal Editor, then run `UnrealMcpSetup-0.15.0.exe` (Windows SmartScreen may warn: choose **More info > Run anyway**). It installs for your Windows account only, into `%APPDATA%\UnrealMcp` (server, plugin, Toolkit and README), and the MCZ MCP Manager if needed.

- **Unreal projects** page: your recent projects are listed, UE 5.8 ones ticked. Setup copies the plugin into each ticked project's `Plugins\UnrealMcpBridge` folder (the `.uproject` file is not changed). An older copy there is moved to `%APPDATA%\UnrealMcp\backup` first. A **browse** page adds other projects.

- With **Add to Claude Desktop** ticked it registers the server as `unreal-csharp`, with the first ticked project as the one `launch_editor` opens by default:

```
"unreal-csharp": {
  "command": "C:\\Users\\YourName\\AppData\\Roaming\\UnrealMcp\\server\\UnrealMcp.Server.exe",
  "args": ["--port", "9878", "--project", "C:\\Projects\\MyGame\\MyGame.uproject"]
}
```

The last page lists the result for each project (installed, skipped, or failed, plus any engine build warning). Restart Claude Desktop afterwards.

Silent install: `UnrealMcpSetup-0.15.0.exe /VERYSILENT /TASKS=claude [/PORT=9878] [/PROJECT="C:\...\MyGame.uproject"]` (with `/PROJECT` only that project gets the plugin; without it every recent UE 5.8 project does).

Uninstall from **Settings > Apps**: it removes the `unreal-csharp` entry and the plugin from the projects setup put it in (a plugin loaded by a running editor is left, and you are told). Licence keys stay in `%LOCALAPPDATA%\MCZ\products`.

By hand: `UnrealMcp.Server.exe --register [--name unreal-csharp] [--port N] [--project X.uproject] [--engine <engine folder>] [--timeout S]` (and `--unregister`). For Claude Code:

```
claude mcp add --scope user unreal-csharp -- "%APPDATA%\UnrealMcp\server\UnrealMcp.Server.exe" --project "C:\Projects\MyGame\MyGame.uproject"
```

First use

Open your project in the Unreal Editor (or ask Claude to `launch_editor`), then ask:

- "*What is in the current level? List the lights and their intensities.*"
- "*Create a Blueprint BP_Door with a static mesh and a box trigger that opens the door when the player overlaps, then compile it and show me the errors.*"
- "*Make a material instance of M_Base with a red base colour and apply it to the selected actors.*"
- "*Start Play In Editor, take a screenshot, then stop.*"

The 30-day trial starts when you install the server on a computer. `UnrealMcp.Server.exe --selftest`, run with the editor open, checks the connection.

Tools

161 tools in these groups:

Group	Tools
Editor	`bridge_status`, `get_editor_state`, `launch_editor`, `wait_for_editor`, `exit_editor`, `restart_editor`, `save_all`, `save_current_level`, `get_dirty_packages`, `list_editors`, `use_editor`

Group	Tools
Levels and actors	`find_actors`, `get_actor`, `spawn_actor`, `destroy_actors`, `set_actor_transform`, `rename_actor`, `set_actor_folder`, `attach_actor`, `add_component`, `remove_component`, `get_current_level`, `load_level`, `new_level`, `trace_line`
Assets and import	`list_assets`, `list_folders`, `get_asset_info`, `asset_thumbnail`, `create_asset`, `create_font`, `duplicate_asset`, `rename_asset`, `delete_assets`, `save_asset`, `import_asset`, `export_asset`, `create_folder`, `reload_asset`
Blueprints	`create_blueprint`, `get_blueprint`, `bp_add_variable`, `bp_remove_variable`, `bp_rename_variable`, `bp_set_default`, `bp_add_function`, `bp_remove_graph`, `bp_add_event`, `bp_add_custom_event`, `bp_add_interface`, `bp_add_component`, `bp_remove_component`, `bp_get_graph`, `bp_add_node`, `bp_connect`, `bp_disconnect`, `bp_set_pin_default`, `bp_remove_node`, `bp_set_node`, `bp_find_nodes`, `bp_compile`
Materials	`mat_get_graph`, `mat_add_expression`, `mat_set_expression`, `mat_delete_expression`, `mat_connect`, `mat_disconnect`, `mat_compile`, `mat_layout`, `mi_get_parameters`, `mi_set_parameters`, `mat_check_usage`
UMG widgets	`create_widget_blueprint`, `get_widget_tree`, `add_widget`, `remove_widget`, `set_widget_property`
Enhanced Input	`create_input_action`, `create_input_mapping_context`, `add_key_mapping`, `remove_key_mapping`, `get_input_mappings`, `list_input_keys`
Settings, INI, console variables, plugins	`list_settings_classes`, `get_settings`, `set_settings`, `get_ini`, `set_ini`, `list_plugins`, `set_plugin_enabled`, `get_cvar`, `set_cvar`
Play In Editor	`start_pie`, `stop_pie`, `pause_pie`, `is_pie_running`, `toggle_eject`
Build	`live_coding_compile`, `hot_reload`, `compile_all_blueprints`, `build_project`
Viewport and editor UI	`get_selection`, `select_actors`, `sync_content_browser`, `focus_actors`, `get_viewport_camera`, `set_viewport_camera`, `set_viewport_realtime`, `viewport_screenshot`, `open_asset_editor`, `close_asset_editors`, `get_open_asset_editors`, `show_notification`, `collect_garbage`
Inspecting the scene	`pick_at_screen`, `find_instances`, `mesh_info`, `mesh_usage`, `set_lods`, `set_view`
Python, console and reflection	`exec_python`, `exec_python_file`, `exec_console`, `call_function`, `call_command`, `list_commands`, `batch`, `get_object`, `get_property`, `set_property`, `set_properties`, `describe_class`, `find_classes`, `find_functions`, `start_job`, `get_job`, `wait_job`
Snapshots and undo	`get_snapshot`, `list_snapshots`, `take_snapshot`, `goto_snapshot`, `snapshot_compare`, `undo`, `redo`, `get_undo_history`
Fab (items you already own)	`fab_status`, `fab_library`, `fab_add_to_project`, `fab_choose`, `fab_page`, `fab_downloads`, `fab_show_mature_content`
Procedural Vegetation	`pve_list`, `pve_set_export`, `pve_export`
Toolkit	`toolkit_list`
Logs	`get_log`, `clear_log`, `set_log_verbosity`
Licence	`license_status`, `license_install`

Notes:

- Object references accept asset paths (`/Game/Foo/BP_Bar`), actor labels and class names.
- `call_function` and `set_property` reach any Blueprint-callable function or property; `exec_python` and `exec_console` cover the rest.
- Fab tools work only with items already in your Fab library; nothing can be bought through them.
- `toolkit_list` lists tested scripts (for example LOD and foliage tools) and how to run them.
- After C++ changes the recommended way is `exit_editor`, a build (`build_project` or Visual Studio), then `launch_editor`. Use `live_coding_compile` only when you want Live Coding.

Several editors and sessions

- Every tool (except `list_editors` and the licence tools) takes an **editor** argument: the project name (or a unique part of it), the port, or the `.uproject` path.
- With **one** editor running, calls without `editor` go to it. With **two or more** running, a call without `editor` is refused with a message listing them, so a session never works in the wrong project by accident (`list_editors`, `launch_editor`, `build_project`, `bridge_status` and `wait_for_editor` are exceptions).
- Before the first call on a connection the server checks that the editor on that port still has the expected project, and refuses the call if not.
- Calls to different editors run in parallel.
- `use_editor` sets this session's default editor when only one is running.

Settings

Command-line options (in `args` of the Claude entry) or environment variables; the option wins:

Option	Variable	Default	Meaning
`--port`	`UNREAL_MCP_PORT`	9878	The port used when no editor is running yet (and by `launch_editor`).
`--host`	`UNREAL_MCP_HOST`	127.0.0.1	Address of the editor plugin.
`--timeout`	`UNREAL_MCP_TIMEOUT`	120	Seconds a call may take (up to 2 hours).
`--project`	`UNREAL_MCP_PROJECT`	(none)	The project `launch_editor` opens by default.
`--engine`	`UNREAL_MCP_ENGINE`	from the project's engine association	The engine folder.

The plugin's own settings are in the editor under **Project Settings > Plugins > Unreal MCP Bridge** (port, start automatically, and switches for Python, console commands and exiting the editor). Console commands: `UnrealMcp.Start`, `UnrealMcp.Stop`, `UnrealMcp.Status`.

Note: the port setting does not tie a session to one editor. The server always checks ports 9878 to 9899, so name the editor in each call when several are open.

Licence

30 days free, then a key. `license_status` and `license_install` always work. From a prompt: `UnrealMcp.Server.exe --license` shows the state (exit code 0 when the tools work, 2 when not); `UnrealMcp.Server.exe --license MCZ1-...` installs a key. See Getting started and the licence terms.

Troubleshooting

- **No editor found**: open the project in the Unreal Editor and check the plugin is enabled (Edit > Plugins, "Unreal MCP Bridge") and running (`UnrealMcp.Status` in the console).
- **"N Unreal editors are running ... Pass editor=..."**: say which project in your request ("*in MyGame, ...*").
- **Engine build warning in setup, or the editor asks to rebuild the plugin**: the plugin was built for a different UE 5.8 build. Rebuilding needs Visual Studio with C++; otherwise install the matching engine build.
- **A call timed out**: the message says the action may or may not have finished; check the editor before repeating it. Raise `--timeout` for long imports.
- **A modal dialog in the editor blocks calls**: close the dialog in the editor.
- **The server is not listed in Claude**: see Troubleshooting in Getting started.

Logs

- In the editor: the Output Log, category **LogUnrealMcp**; Claude can read it with `get_log`.
- Server messages go to standard error, in Claude's MCP log (`%APPDATA%\Claude\logs\mcp-server-unreal-csharp.log`).
- Setup: the installer log in `%TEMP%`.