

Setting Up an MCP Server for the PLS Software (PLS-CADD / PLS-GRID, PLS-POLE, and TOWER)

1. Purpose

This technical note summarizes the requirements and process for enabling the Model Context Protocol (MCP) servers embedded in PLS-CADD/PLS-GRID, PLS-POLE, and TOWER. This feature allows AI assistants (e.g., Copilot, Claude, Gemini/Antigravity...) to query live project data and create and modify structure models through natural-language prompts in PLS software.

2. Background

MCP is an open standard that lets an AI assistant call structured "tools" hosted by a running server, instead of relying only on text pasted into a chat window. Each PLS application hosts its own embedded MCP server on a fixed loopback port; no separate installation, Windows service, or firewall change is required.

Application	Data Source	Tool Count / Access Mode
PLS-CADD	Proxies to the PLS-GRID server over the existing authenticated connection	29 tools - Read-only
PLS-POLE	In-process against the model currently open in PLS-POLE (PLS-GRID not required)	73 tools - Read/Write
TOWER	In-process against the model currently open in TOWER (PLS-GRID not required)	49 tools - Read/Write

All three servers can run simultaneously on their own ports and can all be registered in the same AI client configuration. PLS-CADD's tool set is intentionally read-only and is bounded by the ACL permissions of the signed-in PLS-GRID user. TOWER and PLS-POLE expose a large read/write tool set that can add poles, arms, cables, insulators, and equipment, move existing elements, and run advanced automation such as steel-pole-shaft optimization. However, these features only work against the model already open in that running instance.

3. Technical Requirements

3.1 Software Version and Licensing Prerequisites

Enabling MCP support requires PLS-CADD to be at version 21.18 or later and PLS-GRID Server to be in version 6.18 or later. TOWER and PLS-POLE must be running version 21.18 or later. Engineers also need an MCP-compatible AI client or agent with network access to localhost, since every server communicates exclusively over the loopback interface.

3.2 Network / Server Configuration

Each PLS application listens on its own port. This is defined within each PLS application. All three PLS applications can run simultaneously without conflict when these ports are unique per machine. Every server binds only to 127.0.0.1, the loopback address that always resolves back to the local machine rather than a routable network address, so it can only be reached by processes running on that same PC regardless of firewall configuration. For this reason, no TLS (Transport Layer Security, the encryption behind HTTPS) is used or needed and traffic never leaves the loopback interface. A bearer token (shared secret) can be configured per server, and is recommended on shared or Remote Desktop machines, since any local process under any Windows user can reach loopback ports. Requests without a valid token receive an HTTP 401 response once a token is set.

3.3 Licensing and Access Considerations

PLS-CADD's MCP server requires an active instance of PLS-CADD running with an established PLS-GRID connection. Every query it serves is scoped to the signed-in user's existing PLS-GRID ACL permissions. This means a user can only query what they are already authorized to see in PLS-GRID.

TOWER and PLS-POLE, by contrast, require the application to be running interactively with a visible UI. The MCP server never starts in a hidden or background "vassal" process.

AI agent licenses are organization-level decisions so the organization should confirm which licenses and seats are already available before standardizing on one agent.

3.4 Recommended Rollout Controls

The current MCP tool set can perform real edits in TOWER and PLS-POLE, and because AI-generated output is inherently stochastic, initial deployment of this feature may warrant a measured, phased approach. Experienced specialists with sufficient domain knowledge of PLS-GRID, TOWER, or PLS-POLE should critically evaluate whether requested operations are technically sound.

A test sandbox workspace isolated from production projects and record environments is recommended when first testing capabilities of the MCP. This permits AI-generated changes to be exercised, scrutinized, and reverted without risk to operational work. Production models warrant protection through backups or PLS-GRID revision control, given that loading an alternate model can supplant the active one and that both editing and save operations act directly on local files. Every AI-driven modification should be subjected to validation via ``model_check`` and/or diff comparison prior to the saved output being treated as authoritative, and results generated by AI should not inform critical engineering decisions absent independent verification.

Finally, prompt scope demands deliberate constraint because broad or ambiguous instructions risk inducing unintended large-scale changes.

4. MCP Tools Available in PLS Applications

The tool lists below enumerate every tool exposed by each MCP server, drawn from the PLS-provided server reference ('PLS MCP Servers'). PLS-CADD tools are organized into the seven functional categories defined by that reference.

4.1 PLS-CADD / PLS-GRID Tools

The PLS-CADD MCP server proxies requests through PLS-CADD's existing authenticated PLS-GRID connection so every query is scoped to what the signed-in user is already permitted to see. All 29 tools are read-only. The AI agent can query and report. It cannot create, modify, or delete any data on the PLS-GRID server.

Twenty-seven tools proxy directly to PLS-GRID; the remaining two are shared framework tools also present in TOWER and PLS-POLE.

Server & Health

Tool	Description
ping	Health check -- returns server state, version, and "OK" response.
get_status	Detailed server status: uptime, memory usage, and active connection counts.
get_each_db_state	State counters for each server database (projects, spans, structures, sections, reference manager, Project Manager).

Project & Map Data

Tool	Description
get_all_spans	All span geometry: start/end coordinates, section ID, and flags.
get_all_structures	All structures: location, name, filename, height, max usage, and usage details.
get_all_sections	All cable sections: CPID, cable name, circuit, voltage, ruling span, and bounding structures.
project_filename_list	All projects with filenames, flags, and CPIDs.
project_name_to_id	Convert a project name and client ID to a CPID.
project_files	Reference files attached to a specific project.
get_all_reference_files	All reference files across all projects.
find_terrain_data	Terrain and survey files within a geographic bounding box.
get_project_json	Full project data as JSON: circuits, structures, sections, alignment, materials, assemblies, voltage, and right-of-way.

Project Manager

Tool	Description
get_pm_list_of_projects	All PM projects with CPID, name, revision number, creation/revision timestamps, comment, and user.
get_pm_project_rev_list	Full revision history for a single project.
latest_project_revision_num	Latest revision number and checkout status.
get_files_for_revision	Files and checksums for specific project revision.
get_pm_file	Download a PM file; returns base64-encoded content and size.
get_milestone_list	Named milestone bookmarks for a project.

Analytics

Tool	Description
get_analytics_summary	Analytics summary for all projects (measurement IDs, units, values).
get_survey_index_xml	Survey data index as a human-readable report or raw XML.
get_veg_worksites	Vegetation management worksite data: centroids, numbers, labels, and perimeter polygons.

Admin & Users

Tool	Description
get_user_list	All users in the PLS-GRID database (names, emails, credential flags).
get_acl_groups	All ACL groups configured on the server.
get_user_acl_groups	ACL groups a specific user belongs to.
get_user_flags	Credential flags for a specific user (admin, project management, standards editing, etc.).

Configuration

Tool	Description
get_client_config	PLS-GRID client configuration (the client.json served to web clients).
get_server_config	PLS-GRID server configuration as it was at server startup.

Framework

Tool	Description
get_pls_grid_context	Domain-terminology resource lookup tool; returns background on the PLS-CADD/PLS-GRID data model and available tools. The "run this first" tool named in the server's startup instructions.
mcp_fetch_chunk	Fetches one chunk of an oversized result by cursor, called repeatedly while has_more is true (see Result Size Limits note below).

4.2 TOWER / PLS-POLE Tools

TOWER and PLS-POLE each host their own MCP server that drives the structure model currently open in the running application. Most tools are thin wrappers around the existing CMD command API tools. Forty-eight tools are shared by both programs. TOWER has one additional tower specific tool for adding primary joints. PLS-POLE has twenty-five pole specific tools for adding poles of different material types, davit and X-arms, braces, framing, and the pole/arm/guy movement tools.

Tools marked [P] are PLS-POLE only and tools marked [T] are TOWER only; unmarked tools are available in both programs.

Reports (Read-Only)

Tool	Description
model_check	Validate the open model; returns the Model Check report (geometry, connectivity, data-consistency messages).
model_analyze	Run the structural analysis; returns the Summary Results and detailed Analysis Results reports. Pass return_json=true for structured JSON.
model_view_extents	Return the bounding cube of the model as two x/y/z corner points.

File Operations

Tool	Description
model_open	Open a .tow or .pol file, replacing the current model.
save_structure_file_as	Save the open model to a specified path.
diff	Compare the open model against another file and return the differences report.
convert_lca_to_xml	Convert a PLS-CADD LCA load-case file to XML and return its contents.
export_graphics	Render the model to a bitmap/TIFF image with optional load case, render mode, resolution, and camera settings.
export_dxf	Export the model to a DXF file for use in a CAD system (2D or 3D).

Settings (Read-Only Session Config)

Tool	Description
set_units	Set imperial or metric units for subsequent reports.
set_error_handling	Control stop-on-error behavior and the error output file. The program never exits when driven via MCP.

Analysis Utilities

Tool	Description
plscadd_check	Run a PLS-CADD loads (LCA) check and return the results report; pass return_json=true for structured JSON.
generate_as	Generate Allowable Spans from a LIC wire-load file and return the report.
generate_id	Generate Interaction Diagrams from a LIC wire-load file and return the report.
generate_statistics	Scan a folder of model files and return the model statistics (structured JSON by default).
list_component_types	List all component files used in structures.
get_components	Gets all component files used in structures.

All tools below require the read/write build and modifies the currently open model.

Project Settings (Read/Write)

Tool	Description
new_project	Create a new empty structure model and set the active file path (does not save).
set_customize	Record the source file and a customization comment on the open model.
unset_customize	Clear the customization comment (source path is retained).
set_project_title	Set the project title text.
set_project_notes	Set the project notes text.
set_library	Set a library file for the open model by file path and numeric type code (cable, pole, brace, insulator, material libraries, etc.).
set_loads_file	Set the loads file for the open model and the analysis mode (standard / EIA-F / EIA-G).
set_postproc_options	Set the post-processor executable, output file path, and command-line options used after analysis.
set_parts_file	Set the parts (material cost) file for the open model.
set_voltage	Set the line voltage in kV (General / General Data).
add_reference_manager_file	Attach a file to the Reference Manager of the open model (e.g. a PDF drawing or image).
set_steel_pole_design_code [P]	Set the steel pole design code used for analysis (ASCE, EIA/TIA 222-F/G, EDFASCE).
set_max_pole_segment_length [P]	Set the maximum pole segment length used when subdividing poles for analysis.

Add Poles (PLS-POLE only)

Tool	Description
add_wood_pole [P]	Add a wood pole (property + wood material labels must exist in the libraries).
add_steel_pole [P]	Add a steel pole (property label must exist in the steel pole library).
add_concrete_pole [P]	Add a concrete pole (property label must exist in the concrete pole library).
add_laminated_pole [P]	Add a laminated-wood pole (property label must exist in the laminated pole library).
add_frp_pole [P]	Add a fiber-reinforced polymer (FRP) pole (property label must exist in the FRP pole library).

Add Labels & Joints

Tool	Description
add_primary_joint [T]	Add a primary joint to the open TOWER model (the program appends 'P' to the joint name automatically).
add_pole_label [P]	Add an attachment label to a pole at a given height (from tip or from ground).
add_xarm_label [P]	Add an attachment label to a crossarm at a given distance from its origin.
add_xarm_connection [P]	Add a connection label on a crossarm that connects to another joint in the model.

Add Structural Elements

Tool	Description
add_guy	Add a guy wire (property must exist in the cable library).
add_cable	Add a cable member between two joints (property must exist in the cable library).
add_can	Add a CAN (compound arm assembly) with up to 4 element and 2 insulator type/name pairs.
add_davit_arm [P]	Add a davit arm (property must exist in the davit library).
add_tubular_davit_arm [P]	Add a tubular davit arm (property must exist in the tubular davit library).
add_brace [P]	Add a brace member between two joints (property must exist in the brace library).
add_xarm [P]	Add a crossarm (property must exist in the crossarm library).
add_tubular_xarm [P]	Add a tubular crossarm (property must exist in the tubular crossarm library).
add_framing [P]	Add a framing element to a pole (property must exist in the framing library).
add_pole_linear_appurtenance [P]	Add a linear appurtenance (cable run, conduit, etc.) along a pole between two joints.

Add Insulators & Equipment

Tool	Description
add_clamp	Add a clamp (tangent or dead-end) insulator attachment.
add_suspension	Add a suspension insulator.
add_strain	Add a strain (dead-end) insulator.
add_2part	Add a 2-part (V-string) insulator assembly.
add_post	Add a post insulator.
add_vang	Add a vang insulator.
add_guy_strain_insulator	Add a guy strain insulator along an existing guy.
add_equipment	Add equipment (antenna, transformer, luminaire, etc.) at an attachment joint.
attach_image	Attach a raster image to the open model or its background.

Modify / Move Elements

Tool	Description
pole_attach_height_adjust [P]	Move an existing attachment point up or down on its pole.
guy_anchor_move [P]	Move a guy anchor to new x/y/z coordinates.
pole_move_base [P]	Move a pole's base to new x/y/z coordinates and set its inclination angles.
arm_move_azimuth [P]	Rotate a davit arm or crossarm to a new azimuth angle.
framing_explode [P]	Explode a framing element into its individual component members.
framing_move [P]	Move a framing element to a different pole or vertical position.
guy_strain_insulator_move	Move an existing guy strain insulator to a new distance from the top of its guy.
move_element	Move an element (clamp, suspension, equipment, etc.) from one attachment joint to another.
auto_add_dimensions	Automatically generate dimension annotation lines for the open model.

Advanced Analysis & Automation

Tool	Description
make_saps_model	Generate a SAPS finite-element model file from the currently open structure model.
generate_as_file	Generate Allowable Spans, save the report to a temp file, and optionally save the resulting structure file.
auto_select_pole [P]	Automatically select the best-fit pole material and size based on analysis results.
calculate_2part_load_angles	Calculate and apply load angles for all 2-part insulator assemblies.
calculate_suspension_swing_angles	Calculate and apply swing angles for all suspension insulators.
optimize_steel_pole_shaft [P]	Optimize a steel pole shaft for minimum material over given search ranges; returns the optimization results.
generate_id_file	Generate Interaction Diagrams, optionally saving the resulting structure file and a list of generated files.

Framework (shared with PLS-CADD)

Tool	Description
get_pls_grid_context	Domain-terminology resource lookup tool; returns background on the structure model and available tools.
mcp_fetch_chunk	Fetches one chunk of an oversized result by cursor -- used for large reports such as model_analyze.

The TOWER nor the PLS-POLE tool set do not have a PLS-GRID server connection. All read/write operations act only on the model already open in that running instance.

5. Security Summary

- Loopback-only binding (127.0.0.1) and data is never reachable from another machine on the network.
- The bearer-token authentication is optional. This is recommended on shared/Remote Desktop machines.
- PLS-CADD tools are strictly read-only. Queries are bounded by the logged-in user's existing PLS-GRID ACL permissions.
- TOWER/PLS-POLE read/write tools can only modify the model already open in that running instance. They have no PLS-GRID server connection and cannot touch files outside paths explicitly supplied as arguments.
- Large results are chunked automatically (via mcp_fetch_chunk) so clients with a result-size limit (commonly 1 MB) still receive complete data..

6. Example: End-to-End Setup and Use with GitHub Copilot

The following walks through enabling the PLS-CADD MCP server and using it from GitHub Copilot, including a representative prompt and the tool calls it triggers.

Step 1 - Enable MCP on PLS-GRID Server

This step is performed by the PLS-GRID server administrator. It may be done by directly editing client.json or using the PLS-GRID Web Client Administrator interface to include the row shown below:

```
"McpEnabled": true,
```

1. Edit the client.json with the PLS-GRID Web Client
 - a. Log into the PLS-GRID Web Client as an admin user.
 - b. Go to Admin/Home screen and use Download Client Config button

Download Client Config ↓

- i.
 - c. Go into your downloads folder and edit the client.json file (you may want to save or move this file to another location) and save the file
 - d. Go to Admin/Home screen and use Download Client Config button. The configuration file will automatically be reloaded with this button and this feature will be enabled.
2. (***This is an alternative approach to 1. Above***) Or edit the client.json directly. This file is in the PLS-GRID server's config directory (the config\ subdirectory of the PLS-GRID server installation folder). The exact path depends on where the PLS-GRID Server application is installed.
 - a. Edit this file with a text editor
 - b. verify the format is compatible with JSON
 - c. save the client.json file
 - d. Use the Reload Config button in the Admin/Home screen to restart the service with these new configuration changes.

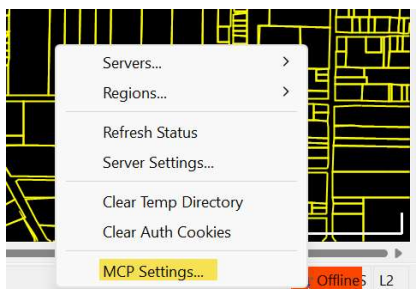
Upload Client Config ↑

Reload Config

i.

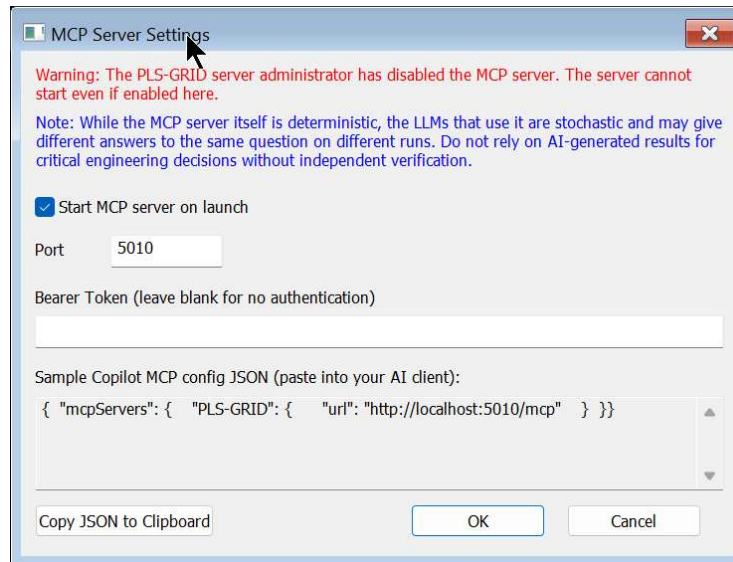
Step 2 - Enable the server in PLS-CADD

1. Open PLS-CADD and right click on the PLS-GRID Status bar in lower right and select the MCP Server Settings.



2. In the MCP Server settings dialog, check "Start MCP server on launch."
3. Define the Port to use for this application. The default is 5010. Leave this at the default unless port 5010 is already in use by another application.
4. Enter a Token. This is optional and used for additional security. This is a shared secret that the AI client must supply with every request. Choose any string — for example, MySecretToken2026. Or leave blank to disable authentication (not recommended on shared or multi-user machines).
5. Click "Copy JSON to Clipboard" to copy the ready-to-paste JSON for the AI Agent configuration file.
6. Click OK and the server starts immediately (no restart is required).

Continue the same process in PLS-POLE and TOWER. PLS-POLE defaults to port 5011 and TOWER defaults to port 5012.



Step 3 - Register the server with GitHub Copilot CLI

The AI client needs to know the server URL (and token, if configured). The exact file location and key name vary by client, but the typical JSON structure used is shown below. If TOWER and PLS-POLE MCP servers are also enabled (with their own tokens, if set), all three can be listed together:

Paste the copied configuration into the Copilot MCP config file.

```
{
  "mcpServers": {
    "PLS-GRID": {
      "tools": [
        "*"
      ],
      "type": "http",
      "url": "http://localhost:5010/mcp"
    }
  },
  "PLS-POLE": {
    "tools": [
      "*"
    ],
    "type": "http",
```

```
"url": "http://localhost:5011/mcp"
},
"TOWER": {
    "tools": [
        "*"
    ],
    "type": "http",
    "url": "http://localhost:5012/mcp"
}
}
```

Step 4 - Confirm connectivity

Check to see if the MCP status bar has updated in the lower right of the status bar in the PLS application.

Or in a GitHub Copilot Desktop session, ask a simple health-check question, e.g.:

- "Is the PLS-GRID server up? What version is it running?"

Copilot calls *ping* for a fast connectivity check, then *get_status* for uptime, memory usage, and active connection counts, and reports the results back in plain language.

Step 5 – Use AI Agents and MCP

PLS-CADD Example

Prompt:

"Show me every project that changed in the last 24 hours."

Resulting tool-call sequence:

- Find files changed in last 24 hours with GRID server/project grouping and 3 other tool calls
- Thought for 5s

Searched `**/*.txt`

View .

Thought for 7s

Find files changed in last 24 hours with GRID server/project grouping `$cutoff = (Get-Date).AddHours(-24)`

Thought for 4s

Summarize changed projects grouped by GRID server `$cutoff = (Get-Date).AddHours(-24)`

Thought for 4s

Here's everything that changed on disk in the last 24 hours, grouped by GRID server:

GRID Server	Project	Files Changed	Last Change
1441 GRID Server	McKee Road Line	15	10/2/2026 10:53 AM
localhost GRID Server	BTS TLine OHL for Data Center	64	10/1/2026 6:18 PM
localhost GRID Server	McKee Road Line 3d	14	10/1/2026 6:03 PM
(workspace root)	(root files) — PLS_MCP_Server_Technical_Note.docx	1	10/2/2026 10:49 AM

Note: this reflects file-system modification times across the whole workspace folder, not session-tracked changes (only our docx edit showed up in session history). Want me to drill into which specific files changed in any of these projects?

TOWER Example

With TOWER's MCP server enabled and the BLE TOWER model open, use GitHub Copilot and a prompt such as:

"Use the BLE example tower model, run a full structural analysis, and give me a summary of the governing member utilization by load case, the foundation reactions at each support joint, and the insulator/hardware usage and just report the values. Time is money, so let me know how long the whole analysis took."

Result:

BLE Example Tower — Structural Health Report

(full model_analyze run, port 5012, 51-tool MCP server, ~67 seconds end-to-end)

Governing member utilization by load case (max 20 cases):

Load Case	Max Usage	Governing Member
NESC Heavy	76.6%	11XY
Extreme Wind	82.0%	40BE-Leg4P
Heavy Ice	88.5%	11XY
Broken Subconductor (4a/b/c)	89.97–92.8%	40BE-Leg7P
Broken Ground Wire (5a)	93.2%	11X ← overall max
Differential Ice	88.7%	89Y
Construction cases	32–60%	various

No member exceeds 100% — tower passes all 20 cases with margin as low as ~7%.

Foundation reactions (governing case per joint):

Joint	x_force	y_force	z_force (vert.)	Resultant
FP	-23.22	-18.31	-90.83	95.53
LX_FP	8.47	0.14	13.33	15.79
LXY_FP	-11.11	-17.84	61.15	64.66
LY_FP	10.22	-11.36	-44.62	47.16
(kips/kip-ft, as reported — no limit comparison per your request)				

Insulator hardware:

- Suspension (G1/G2): 14.6% tension usage — large margin.
- Clamps (C1–C11): 0.02–0.3% usage — negligible.
- 2-Parts insulators (C1–C3): 40.8–65.0% in the snapshot case, up to **94.9%** peak across all cases — tightest margin in the model.

Flags: 8 crossing-diagonal pairs and 449 redundant-member checks evaluated — no usage over 10% in redundants; crossing diagonals all well under capacity.

Process time: ~67 seconds (reopen model → run full analysis → chunked JSON retrieval/reassembly (2.5MB, 4 chunks) → extract 6 data tables → synthesize) — versus the 20+ minutes a manual UI review across 20 load cases would typically take.