

PULSE / Civil 3D

Guide and commands

Autodesk Civil 3D 2023-2027

Installation, first steps and natural-language examples. Technical command names remain unchanged.

Verified source: <https://github.com/LuDattilo/pulse-for-civil3d>

Branch: main | Commit: 1da6e3cd6a5bd8b2ff4652e0aa7ca88ab0777e8f

2026-10-04 | EN

Descriptions in English. Command names, parameters and technical values are unchanged. Source catalogue dated 4 October 2026; translations updated on 8 October 2026.

<https://pulse-ai.it/guida-civil3d.html>

Installation and connection

Close Civil 3D and install the dedicated EXE package. Restart, open a drawing and switch Pulse on from its tab or with PULSECONNECT. Check that your AI client lists pulse-civil3d. Source targets cover 2023-2027; README runtime evidence refers to Civil 3D 2025.

Connect an MCP-compatible AI client: Claude, Codex, opencode or another client, including local models.

Open the project and describe the goal. Check the result and approve changes when requested.

Work on a project copy. Request a read operation or preview before changes, check the selection and approve only the intended operations.

On the first eligible registration, a personal 14-day Premium Trial. Another request does not renew an existing trial.

Example requests

- > Compare these two surfaces and show cut and fill volumes.
- > List the alignments and read the selected profile.
- > Find the survey group points and export them in the required format.

FEATURES

Surfaces and volumes

Read elevations, areas and volumes; create point-based and comparison surfaces with previews where supported.

Alignments and profiles

Inspect geometry, stations, offsets, profiles and sections; create alignments from polylines and surface profiles.

Points and surveying

Query COGO points and groups, import and export points and manage user-defined properties.

Corridors and quantities

List and rebuild corridors; query sections, material lists and available quantities.

Networks and drawings

Read pipes and structures, check networks and inspect drawing objects, layers and styles.

Dynamo and automation

Inspect Dynamo status, generate and run graphs; custom code follows the configured safety settings.

Command catalog

Descriptions in English. Command names, parameters and technical values are unchanged. Source catalogue dated 4 October 2026; translations updated on 8 October 2026.

Alignment

`get_alignment_geometry`

The geometry of an alignment, entity by entity in order: the entity type (Line, Arc, Spiral, SpiralCurveSpiral, ...) and each sub-entity's type, start and end station, length and - for arcs - radius. The entities are paginated with limit and offset; disconnectedSkipped means the alignment holds disconnected entities that are not listed. Stations and lengths are in drawing units.

Scope: | sources\civil3d\src\Pulse.Civil3D.Server\Tools\AlignmentTools.cs

`get_feature_line`

One feature line, by name or handle, as `list_feature_lines` returns them: name, handle, site, style, points count, 2D and 3D length, minimum and maximum elevation and grade, plus its points - index, x, y, z, the bulge of the segment that starts there and the grade in and out where they exist (the first point has no grade in, the last no grade out). The points are paginated with limit and offset; coordinates are in drawing units (WCS: X east, Y north).

Scope: | sources\civil3d\src\Pulse.Civil3D.Server\Tools\AlignmentTools.cs

`get_material_quantities`

The cut, fill and usable volumes of one material list along the sample lines of a group: one entry per sample line with the incremental and cumulative volumes at its station, then the total (the cumulative at the last sample line). Name the alignment and the material list as `list_material_lists` returns them; 'group' picks the sample line group when the alignment has more than one. Stations are paginated with limit and offset.

Scope: | sources\civil3d\src\Pulse.Civil3D.Server\Tools\AlignmentTools.cs

`get_point_at_station`

The x and y of each [station, offset] item on an alignment, up to 5000 per call. A station out of the alignment's range is not an error: it comes back with x and y null and outside true, and the answer counts the items inside and outside. Station and offset use the alignment's stationing, in drawing units.

Scope: | sources\civil3d\src\Pulse.Civil3D.Server\Tools\AlignmentTools.cs

`get_profile_data`

The profile data of an alignment: the PVIs (station, elevation, and the grade in and out where they exist - the first PVI has no grade in, the last no grade out) and, when stations are given, the elevation at each one. Grades are ratios (0.02 = 2%). Stations use the alignment's stationing, in drawing units. Name the profile as `list_alignments` returns it in the alignment's profiles.

Scope: | sources\civil3d\src\Pulse.Civil3D.Server\Tools\AlignmentTools.cs

get_section_data

One section with its geometry points, by handle or by alignment and station (the two ways are mutually exclusive). The section gives its station, its source (sourceName and sourceType: TinSurface, Grading, Corridor, CorridorSurface, PipeNetwork, Material, GridSurface), minimum and maximum elevation, left and right offset and swath width where the section kind supports them, and its style. The points are paginated with limit and offset and give, per point, the index, the offset and elevation in the section plane and segmentTo - the index of the point its segment goes to, -1 when the point starts no segment. With alignment and station the tool picks the nearest sample line within tolerance (default 0.000001); when none is there, the answer is 100 with up to 50 valid stations. When the sample line carries several sections, the first is described and otherSections lists the others by handle.

Scope: | sources\civil3d\src\Pulse.Civil3D.Server\Tools\AlignmentTools.cs

get_station_offset

The station and offset on an alignment of each [x, y] point in drawing units (WCS: X east, Y north), up to 5000 per call. A point out of the alignment's range is not an error: it comes back with station and offset null and outside true, and the answer counts the points inside and outside. Station and offset use the alignment's stationing, in drawing units.

Scope: | sources\civil3d\src\Pulse.Civil3D.Server\Tools\AlignmentTools.cs

list_alignments

List the alignments of the active drawing, sorted by name: name, handle, type (Centerline, Offset, CurbReturn, Utility, Rail), site, length, starting and ending station, style and the profiles' names and handles. Stations, lengths and offsets are in drawing units (see get_drawing_info). Paginated with limit and offset.

Scope: | sources\civil3d\src\Pulse.Civil3D.Server\Tools\AlignmentTools.cs

list_feature_lines

List the feature lines of the active drawing's sites, sorted by site and name: name, handle, site, style, number of points, 2D and 3D length, minimum and maximum elevation, and minimum and maximum grade. A site (name or handle) narrows the list to that site's feature lines; feature lines that are in no site are not listed, because the Civil 3D API reaches a feature line through its site only. Lengths and elevations are in drawing units (see get_drawing_info); grades are ratios (0.02 = 2%). Paginated with limit and offset.

Scope: | sources\civil3d\src\Pulse.Civil3D.Server\Tools\AlignmentTools.cs

list_material_lists

List the material lists of the drawing's sample line groups, sorted by alignment and name: each group with its materials and their quantity type (Cut, Fill, CutFill, Earthwork, Structure). These are the names and guides get_material_quantities takes. Pass 'alignment' to narrow the answer. Paginated with limit and offset.

Scope: | sources\civil3d\src\Pulse.Civil3D.Server\Tools\AlignmentTools.cs

list_sample_lines

List the sample lines of the active drawing, sorted by alignment, group and station: the alignment and the group they belong to (name and handle), the sample line's name, handle, number, station, lockToStation, the number of sections attached to it (sectionCount) and its style. An alignment (name or handle) narrows the list to that alignment. A sample line belongs to a group and a group to an alignment, so no sample line of a drawing with no alignments is listed. Stations are in drawing units (see get_drawing_info). Paginated with limit and offset.

Scope: | sources\civil3d\src\Pulse.Civil3D.Server\Tools\AlignmentTools.cs

Base

`find_objects`

Find the objects of the model space of the active drawing, filtered by type, layer and name. type and layer accept the wildcards * and ?; type matches the .NET class name (Polyline, TinSurface) or the DXF name (LWPOLYLINE); name matches exactly, ignoring case. Answers handle, type, dxfName, layer and name, sorted by handle. Paginated with limit and offset. Pass the handles to `get_object_properties` or `select_objects`.

Scope: | sources\civil3d\src\Pulse.Civil3D.Server\Tools\BaseTools.cs

`get_coordinate_settings`

The coordinate system assigned to the active drawing (code, description, projection, datum, unit, category; assigned false when there is none, the library's '.' entry counting as none) and the local-to-grid transformation parameters: reference and rotation points, the rotation to grid north or azimuth in degrees, the grid scale factor when the user defines it, and the sea level scale factor. 'transformation' is {enabled: false} while the transformation settings are off. Compare with Civil 3D's Drawing Settings > Transformation.

Scope: | sources\civil3d\src\Pulse.Civil3D.Server\Tools\BaseTools.cs

`list_layers`

List the layers of the active drawing, sorted by name: name, handle, off, frozen, locked, whether it is the current layer, the color's display name (the one the layer manager shows, so localized) and the linetype's name. Paginated with limit and offset.

Scope: | sources\civil3d\src\Pulse.Civil3D.Server\Tools\BaseTools.cs

`list_styles`

List the styles of the active drawing. With type (surface, alignment, profile, profileView, pipe, structure, point, alignmentLabelSet, profileLabelSet) it lists that kind's styles, sorted by name: name and handle. Without type it answers how many styles of each kind the drawing holds. Paginated with limit and offset.

Scope: | sources\civil3d\src\Pulse.Civil3D.Server\Tools\BaseTools.cs

`select_objects`

Select the objects named by their handles in Civil 3D (the pickfirst selection), in the given order, and by default frame them in the active view. The drawing does not change. Handles not found in the drawing come back in notFound and are not selected; an empty selection clears the current one. Read the handles from `find_objects`, `list_alignments`, `list_corridors` or the other list tools.

Scope: | sources\civil3d\src\Pulse.Civil3D.Server\Tools\BaseTools.cs

Code

`analyze_send_code_gaps`

Which operations keep needing `send_code_to_civil3d`: its calls in the audit log (`~/pulse/civil3d/logs/audit.jsonl`) grouped by ``purpose``, the most frequent first, each with the number of calls, of real runs, and the first and last date. Purposes are grouped by exact text (trimmed, case-insensitive). The code itself is never in the log. Answered by the server: works without Civil 3D. Use it to see which dedicated tool would help most.

Scope: | `sources\civil3d\src\Pulse.Civil3D.Server\Tools\CodeTools.cs`

`send_code_to_civil3d`

LAST RESORT: runs C# code in Civil 3D. Use it only when no dedicated tool covers the operation (see `tools/list`). ``code`` is the body of a method that returns object. It receives document (Document), database (Database), editor (Editor), civilDocument (CivilDocument) and transaction (Transaction), already open: open objects with it, and never commit, abort or dispose it. Never start a transaction of your own: `StartTransaction`, `StartOpenCloseTransaction` and `TopTransaction` are refused. Pulse commits when the code returns and rolls everything back when it throws. The returned value comes back as JSON (numbers, strings, lists, anonymous objects); an AutoCAD object comes back as its handle, type, layer and name. Civil 3D 2025-2027 compile C# 13 (Roslyn); 2023-2024 compile C# 5 only (CodeDom): no `$$$` strings, `?.`, `nameof`, `out var`, ``is`` patterns, tuples or local functions. No files, network, processes, registry, reflection or dynamic. Queued commands (`SendStringToExecute`, `Editor.Command`) and DWG/DXF file operations are refused. ``purpose`` is required: one line on what the code does and why no dedicated tool covers it; the audit log keeps it in clear, never the code. The preview (dryRun omitted) compiles and runs nothing. The run (dryRun:false and the preview's ``drawing``) shows the user the purpose and the whole code in Civil 3D and runs only after a yes, so no separate consent in chat is needed. Refused with `PermissionDenied` while `EnableCodeExecution` is off in `~/pulse/civil3d/settings.json`: then do not retry. One ANNULLA (UNDO) in Civil 3D undoes a run.

Scope: | `sources\civil3d\src\Pulse.Civil3D.Server\Tools\CodeTools.cs`

Corridor

`list_corridors`

List the corridors of the active drawing, sorted by name: name, handle, whether out of date, the baselines (alignment, profile, and each region's name, start and end station and assembly) and the corridor surfaces' names. Stations are in drawing units. Paginated with limit and offset.

Scope: | `sources\civil3d\src\Pulse.Civil3D.Server\Tools\CorridorTools.cs`

Drawing

`get_drawing_info`

Describe the active drawing: linear and angular units, drawing scale, coordinate system (code, description, projection, datum), how many Civil 3D objects of each kind it holds (surfaces, alignments, pipe networks, corridors, COGO points, point groups, sites, siteless feature lines, intersections) and which surfaces, corridors and point groups are out of date. Lengths and elevations in every other answer use these drawing units.

Scope: | `sources\civil3d\src\Pulse.Civil3D.Server\Tools\DrawingTools.cs`

get_object_properties

Read the public properties of objects by handle. Numbers, texts, booleans and enums (as names) come as they are, points as {x,y,z}, object references as {handle, name} (StyleId gives the style's handle and name). Collections and other complex properties are skipped and counted, properties that throw are listed with the exception type, null values are listed by name. Up to 100 handles and 200 properties per object; handles not in the drawing come back in notFound.

Scope: | sources\civil3d\src\Pulse.Civil3D.Server\Tools\DrawingTools.cs

get_selected_objects

List the objects the user selected in Civil 3D before the call (the pickfirst selection), in selection order: handle, .NET type, DXF name, layer and name. Paginated with limit and offset; the answer has totalCount, returnedCount, truncated and nextOffset. With nothing selected the list is empty and selectionStatus says why. Pass the handles to get_object_properties.

Scope: | sources\civil3d\src\Pulse.Civil3D.Server\Tools\DrawingTools.cs

Dynamo

dynamo_generate_graph

Write a complete Dynamo graph (.dyn) for this Civil 3D from a graph description: nodes (value inputs, sliders, Code Block, Watch, library functions by FunctionSignature, Python Script, package nodes), connectors, groups and notes, in the dynamo-python skill format. Library nodes must be real nodes of this Civil 3D: FunctionSignature is Namespace.Type.Member@type1,type2, and an unknown one is refused with the closest ones. Python nodes are checked (engine line, sandbox, Python 2 idioms). The graph is validated before it is saved; nothing runs. Preview first (dryRun omitted), then dryRun: false with the drawing it returns. DesignScript pitfalls: 'cond ? x : []' always gives []; use '[[, x][cond ? 1 : 0]'; DSCore.Math.Abs on an empty list fails with an ambiguous overload (Abs(double) or Abs(int)); PullOntoPlane gives a generic Curve.

Scope: | sources\civil3d\src\Pulse.Civil3D.Server\Tools\DynamoTools.cs

dynamo_get_status

The Dynamo state in the connected Civil 3D (not loaded, loaded, started without or with its window), its version, the run route that dynamo_run_graph will use, and the outcome of the last graph run. Read only.

Scope: | sources\civil3d\src\Pulse.Civil3D.Server\Tools\DynamoTools.cs

dynamo_list_graph_io

Read a Dynamo graph (.dyn) from disk and list what dynamo_run_graph can set and read: its inputs (id, name, type, current value), its outputs, the Dynamo version it was saved with, its Python nodes and engine. Answered by the Pulse server: works without Civil 3D. Call it before dynamo_run_graph and pass only the listed input names in inputValues.

Scope: | sources\civil3d\src\Pulse.Civil3D.Server\Tools\DynamoTools.cs

dynamo_run_graph

Run a Dynamo graph (.dyn) in the connected Civil 3D, with new values for its inputs, and read its outputs and node warnings. Call `dynamo_list_graph_io` first: `inputValues` takes only the listed input names. The graph can change the drawing, so this is a write: preview first (`dryRun` omitted), then `dryRun: false` with the drawing it returns; the user confirms every run. Refused while the Dynamo window is open. Route A runs Dynamo without its window and returns outputs; route B (`AECCRUNDYNAMOSCRIPT`) returns no outputs. Route B cannot read node errors: succeeded means the command ran. Both routes return `databaseEvents`, the objects appended, modified and erased in the drawing. Up to 300 s.

Scope: | `sources\civil3d\src\Pulse.Civil3D.Server\Tools\DynamoTools.cs`

Meta

get_session_info

Which Pulse pieces are running and how they are set up: the server (version, protocol, modules, settings and log files), the Civil 3D instances with Pulse switched on and the one in use and, when one is connected, its open drawings, read-only mode and Dynamo state. Answers even without Civil 3D: call it FIRST when a tool is missing or a call fails with a `BridgeFailure`.

Scope: | `sources\civil3d\src\Pulse.Civil3D.Server\Tools\MetaTools.cs`

say_hello

Test the connection to Civil 3D. Answers which Civil 3D replied (product, year, AutoCAD version), its language (LOCALE), the active drawing and the plugin and protocol versions. Needs no open drawing.

Scope: | `sources\civil3d\src\Pulse.Civil3D.Server\Tools\MetaTools.cs`

Pipe

add_structure

Add one structure to an existing pipe network. The family and size come from the parts list's structure domain, as `list_parts_lists` returns them; 'x' and 'y' are the plan position, 'z' the structure's elevation (omit it to sit on the network's reference surface), 'rotation' is in degrees and 'applyRules' defaults to true as Civil 3D's own Add Structure does. Omitting `dryRun` is a preview. With `dryRun false` the structure is added in one write transaction.

Scope: | `sources\civil3d\src\Pulse.Civil3D.Server\Tools\PipeTools.cs`

check_pipe_network

The problems of one network, one entry per pipe or structure with the check's name. Always: `missing_structure_start/end` (a pipe with no structure at an end), `no_pipes` (a structure with no pipes), `zero_length`, `adverse_slope` (the pipe rises along its direction). With the limits given: `slope_below_minimum` and `slope_above_maximum` (percent), `cover_below_minimum` (drawing units, from the reference surface; `cover_unavailable` when there is none).

Scope: | `sources\civil3d\src\Pulse.Civil3D.Server\Tools\PipeTools.cs`

`get_pipe_network`

The pipes and structures of one network. Every pipe: handle, family, size, straight or curved, its two ends, invert elevations (the centreline minus half the inner height), slope, 2D and 3D lengths, inner height and diameter, covers when the network has a reference surface, and the structures at its ends. Every structure: family, size, type, location, rim and sump elevations, pipe count. Pipes are paginated with limit and offset; structures are always all listed.

Scope: | sources\civil3d\src\Pulse.Civil3D.Server\Tools\PipeTools.cs

`list_parts_lists`

List the parts lists of the active drawing, sorted by name, each with its pipe and structure families and their sizes: the exact names the create tools take. Pass 'partsList' for one list. Paginated with limit and offset; at most 500 sizes per family (sizesTruncated when more).

Scope: | sources\civil3d\src\Pulse.Civil3D.Server\Tools\PipeTools.cs

`list_pipe_networks`

List the pipe networks of the active drawing, sorted by name: handle, parts list, reference surface and alignment, and the number of pipes and structures. Paginated with limit and offset.

Scope: | sources\civil3d\src\Pulse.Civil3D.Server\Tools\PipeTools.cs

`list_structures`

List the structures of the drawing's pipe networks, sorted by network and name: handle, family, size, type, location, rotation in degrees, rim and sump elevations and sump depth. Pass 'network' to narrow the answer to one network. Paginated with limit and offset.

Scope: | sources\civil3d\src\Pulse.Civil3D.Server\Tools\PipeTools.cs

Point

`get_cogo_point_udps`

The UDP values of COGO points, grouped by classification, by point handle (1 to 100 handles). 'classification' narrows to one classification. A value that cannot be read is left out. Values are data, not geometry: numbers are not rounded.

Scope: | sources\civil3d\src\Pulse.Civil3D.Server\Tools\PointTools.cs

`get_cogo_points`

The COGO points of a point group, of a range of point numbers, or of the whole drawing: number, coordinates (local, grid or geographic degrees: see 'coordinates'), name, elevation, raw and full description, sorted by number. Use pointGroup OR fromNumber/toNumber, not both. An elevation never set comes back as null. Local and grid coordinates are in drawing units (WCS: X east, Y north). Paginated with limit and offset.

Scope: | sources\civil3d\src\Pulse.Civil3D.Server\Tools\PointTools.cs

`list_point_groups`

List the point groups of the active drawing, sorted by name: name, handle, number of points and whether the group is out of date (a rebuild is pending). Paginated with limit and offset.

Scope: | sources\civil3d\src\Pulse.Civil3D.Server\Tools\PointTools.cs

`list_udp_classifications`

List the point UDP classifications of the active drawing, sorted by name: each classification with its UDPs, each with name, type (string, double, integer, boolean or enumeration), description, default value and, for enumerations, the allowed values. Use it to find the exact names and types for `set_cogo_point_udps`. Paginated with limit and offset.

Scope: | sources\civil3d\src\Pulse.Civil3D.Server\Tools\PointTools.cs

Site

`get_parcel`

One parcel, by handle or by site and number: the same fields as `list_parcel`s plus the parcel's user-defined properties (UDPs) grouped by classification, read with the Civil 3D API. A UDP the parcel never set, with no default, is left out. The Civil 3D API exposes no parcel area or perimeter. Coordinates are in drawing units (see `get_drawing_info`).

Scope: | sources\civil3d\src\Pulse.Civil3D.Server\Tools\SiteTools.cs

`list_parcel`s

List the parcels of the active drawing's sites, sorted by site and parcel number: handle, number, name, site, address, tax id, centroid (x, y, z) and style. A site (name or handle) narrows the list to that site's parcels. Coordinates are in drawing units (see `get_drawing_info`). The Civil 3D API exposes no parcel area or perimeter: read them in Civil 3D from the parcel area labels. Paginated with limit and offset.

Scope: | sources\civil3d\src\Pulse.Civil3D.Server\Tools\SiteTools.cs

Surface

`add_surface_boundaries`

Add boundaries to a TIN surface from the lines and polylines named by their handles (1 to 500) and rebuild it. 'type' is one of outer (defines the surface extent), dataClip (trims the data to it), show, hide; 'midOrdinate' is in drawing units (default 1.0); 'nonDestructiveBreakline' defaults to false. Omitting `dryRun` is a preview: it names the surface and the objects. With `dryRun` false the boundaries are added in one write transaction.

Scope: | sources\civil3d\src\Pulse.Civil3D.Server\Tools\SurfaceTools.cs

`add_surface_breaklines`

Add standard breaklines to a TIN surface from the lines and polylines named by their handles (1 to 500) and rebuild it. 'midOrdinate' is the mid-ordinate distance in drawing units (default 1.0); the optional 'maximumDistance', 'weedingDistance' and 'weedingAngle' default to 0, which adds every vertex. Omitting `dryRun` is a preview: it names the surface and the objects. With `dryRun` false the breaklines are added in one write transaction.

Scope: | sources\civil3d\src\Pulse.Civil3D.Server\Tools\SurfaceTools.cs

`add_surface_point_file`

Add a point file to a TIN surface's definition and rebuild it. The format comes from 'format' (a name Civil 3D knows) or from the file's extension. Omitting `dryRun` is a preview: it names the surface, the format and the file's points. With `dryRun` false the file is added in one write transaction.

Scope: | sources\civil3d\src\Pulse.Civil3D.Server\Tools\SurfaceTools.cs

`create_volume_surface`

Create a TIN volume surface comparing two surfaces, for cut and fill volumes (scavi e sterri). Omitting `dryRun` is a preview: nothing changes and the answer names the style it would use. With `dryRun` false the surface is created with the chosen style (the drawing's first surface style when style is omitted) and rebuilt; the answer names the style and, when readable, the unadjusted cut, fill and net volumes. The name must be unused, and the two surfaces must be different. Volumes are in cubic drawing units.

Scope: | sources\civil3d\src\Pulse.Civil3D.Server\Tools\SurfaceTools.cs

`get_surface_elevations`

Read the elevation of a surface at points given as [x, y] in drawing units (WCS: X east, Y north), up to 5000 per call. A point outside the surface is not an error: it comes back with z null and outside true, and the answer counts the points inside and outside. Name the surface by name or handle, as `list_surfaces` returns them.

Scope: | sources\civil3d\src\Pulse.Civil3D.Server\Tools\SurfaceTools.cs

`get_surface_volumes`

Read the volumes of a volume surface: unadjusted and adjusted cut, fill and net volumes, the cut and fill factors, and the base and comparison surfaces it compares. Volumes are in cubic drawing units. Name the surface by name or handle, as `list_surfaces` returns them (isVolume true); a surface that is not a volume surface is refused with the list of what it is.

Scope: | sources\civil3d\src\Pulse.Civil3D.Server\Tools\SurfaceTools.cs

`list_surfaces`

List the surfaces of the active drawing, sorted by name: name, handle, type (TinSurface, GridSurface, TinVolumeSurface, GridVolumeSurface), style, minimum, maximum and mean elevation, number of points, and for TIN surfaces the number of triangles and the 2D and 3D areas (grid surfaces: the areas). `outOfDate` means the surface needs a rebuild; `empty` means it has no triangles or cells. Paginated with limit and offset. Elevations and areas are in drawing units (see `get_drawing_info`).

Scope: | sources\civil3d\src\Pulse.Civil3D.Server\Tools\SurfaceTools.cs

`paste_surface`

Paste one surface into another TIN surface and rebuild it, as Civil 3D's Paste Surface does. Name the surface to paste into and the source surface to paste from; they must differ and the target must be a TIN surface. Omitting `dryRun` is a preview: it names both surfaces. With `dryRun` false the paste happens in one write transaction.

Scope: | sources\civil3d\src\Pulse.Civil3D.Server\Tools\SurfaceTools.cs

Write

`create_alignment_from_polyline`

Create an alignment from a Polyline, Polyline2d or Polyline3d. Omitting `dryRun` is a preview: nothing changes and the answer names the layer, style and label set it would use. With `dryRun` false the alignment is created on the layer the drawing settings give alignments (the current layer when that layer is missing), with the chosen style and label set (the drawing's first when omitted), optionally inside a site, optionally adding curves between tangents, and optionally erasing the polyline. The name must be unused.

Scope: | sources\civil3d\src\Pulse.Civil3D.Server\Tools\WriteTools.cs

`create_cogo_points`

Create COGO points at the given locations, each with an optional description. Omitting `dryRun` is a preview: nothing changes and the answer says how many points would be created. With `dryRun` false the numbers follow the drawing's point settings, and the answer lists the created numbers and handles. One call creates up to 5000 points.

Scope: | sources\civil3d\src\Pulse.Civil3D.Server\Tools\WriteTools.cs

`create_surface_from_points`

Create a TIN surface from a point group and rebuild it. Omitting `dryRun` is a preview: nothing changes and the answer names the style it would use. With `dryRun` false the surface is created with the chosen style (the drawing's first surface style when style is omitted) and built from the point group's points. The name must be unused; the answer names the style and the point group used.

Scope: | sources\civil3d\src\Pulse.Civil3D.Server\Tools\WriteTools.cs

`create_surface_profile`

Create a surface (EG) profile for an alignment. Omitting `dryRun` is a preview: nothing changes and the answer names the layer, style and label set it would use. With `dryRun` false the profile is created on the layer the drawing settings give profiles (the current layer when that layer is missing), with the chosen style and label set (the drawing's first when omitted); the name defaults to '<alignment> - <surface>' and must be unused under the alignment. The answer names type, stations and the layer, style and label set used.

Scope: | sources\civil3d\src\Pulse.Civil3D.Server\Tools\WriteTools.cs

`delete_objects`

Erase the objects named by their handles (1 to 500), as `find_objects` returns them. Omitting `dryRun` is a preview: it lists the type, name and layer of every object and refuses objects Civil reports as in use by another object, so a referenced object never erases half of what was asked. With `dryRun` false every object is erased in one write transaction: all or nothing.

Scope: | sources\civil3d\src\Pulse.Civil3D.Server\Tools\WriteTools.cs

`export_points`

Write COGO points to a point file on this machine: the whole drawing, or one point group with 'pointGroup'. The format comes from 'format' (a name Civil 3D knows) or from the file's extension. Omitting `dryRun` is a preview: it names the path, the format and how many points would go. With `dryRun` false the file is written after the confirmation and the answer says how many points went.

Scope: | sources\civil3d\src\Pulse.Civil3D.Server\Tools\WriteTools.cs

`import_points`

Import COGO points from a point file on this machine. The format comes from 'format' (a name Civil 3D knows) or from the file's extension; 'pointGroup' puts the imported points in that group. Omitting `dryRun` is a preview: it names the format and counts the file's points. With `dryRun` false the points are imported in one write transaction and the answer says how many came in.

Scope: | sources\civil3d\src\Pulse.Civil3D.Server\Tools\WriteTools.cs

`rebuild_corridor`

Rebuild a corridor after the elements it depends on changed (alignments, profiles, surfaces). Omitting `dryRun` is a preview: it says whether the corridor is out of date and how many regions it has. The rebuild can be slow (its limit is 300 s); the answer says how long it took and whether the corridor is still out of date. Name the corridor by name or handle, as `list_corridors` returns them.

Scope: | sources\civil3d\src\Pulse.Civil3D.Server\Tools\WriteTools.cs

`set_cogo_point_udps`

Set UDP values on COGO points. 'updates' is an array of {handle, udp, value, classification?} objects, 1 to 500: the value must match the UDP's type (a string, a number, a boolean, or one of an enumeration's values), and 'classification' disambiguates a UDP name shared by more than one classification. Omitting `dryRun` is a preview: it lists what would be set with the current values. With `dryRun` false every value is set in one write transaction.

Scope: | sources\civil3d\src\Pulse.Civil3D.Server\Tools\WriteTools.cs

Verified source

<https://github.com/LuDattilo/pulse-for-civil3d>

main / 1da6e3cd6a5bd8b2ff4652e0aa7ca88ab0777e8f

<https://pulse-ai.it/guida-civil3d.html>