

PULSE / Solibri

Guide and commands

Solibri Office 26.6

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

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

Branch: main | Commit: bc7fd4c5041832a48c62c1b0526fa08255ba02e5

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-solibri.html>

Installation and connection

Solibri Office with the supported API is required. The Java plugin and MCP server must come from the same package. Open your project and, if needed, the Pulse Bridge view. Configure your AI client using the package instructions. Commercial installer and licensing are being prepared; do not replace an approved package with development scripts.

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.

Automatic download and serial delivery are not yet connected to this form. Contact us for this product.

Example requests

- > Run checking and show the critical results of the selected rule.
- > Isolate the involved components and save a view snapshot.
- > Read the ITO table and export issues to BCF.

FEATURES

Rules and checking

Inspect rulesets, rules and parameters; change supported parameters and run project checking.

Results and triage

Filter failed components by severity and decision; update result triage.

Components and selections

Find components and IFC properties; read or update the Selection Basket.

Visual review

Manage camera, visibility and colors; isolate elements and save 3D view snapshots.

ITO and quantities

Read Information Takeoff definitions and tables, recalculating when requested.

BCF and libraries

Export issues to BCF; use the library functions exposed by the plugin.

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.

Bcf

`solibri.bcf.export`

Exports Solibri issues to a BCF (.bcfzip) file on disk and returns the path. version: V1_0|V2_0|V2_1|V3_0 (default V2_1). scope: ALL|SELECTED|MARKED (default ALL).

Scope: | sources\solibri\server\Pulse.Solibri.Server\Tools\BcfTools.cs

Checking

`solibri.checking.chosen_get`

Lists the rules and rulesets (regole/ruleset) currently checked in Solibri's Checking view: what would run if `solibri.checking.run` were called now. Unlike `solibri.rulesets.list`, which shows ruleset contents, this reports the current selection state in the interface.

Scope: | sources\solibri\server\Pulse.Solibri.Server\Tools\CheckingTools.cs

`solibri.checking.chosen_rows`

Rows currently highlighted in the checking results table, resolved to the same Result/componente/issue objects used by the other tools - to act on "what the user selected in Solibri now" rather than just what the assistant selects.

Scope: | sources\solibri\server\Pulse.Solibri.Server\Tools\CheckingTools.cs

`solibri.checking.chosen_set`

Sets which regole/ruleset are checked in the Checking view (overrides the current selection) - use this to "execute only these rules" before `solibri.checking.run`. Refs that do not resolve are ignored and reported in 'notFound'.

Scope: | sources\solibri\server\Pulse.Solibri.Server\Tools\CheckingTools.cs

`solibri.checking.decision_set`

Set the triage (ACCEPTED/REJECTED/UNDEFINED) of a checking result, given its uniqueKey (from `solibri.results.get`) - e.g. to accept an already discussed deviation instead of letting it reappear at every check.

Scope: | sources\solibri\server\Pulse.Solibri.Server\Tools\CheckingTools.cs

`solibri.checking.failed`

Components that fail a given rule, from `Checking.getFailedComponents` - direct filter for targeted triage (e.g. "only CRITICALs still UNDEFINED of this rule") instead of scrolling through all results with `solibri.results.get`.

Scope: | sources\solibri\server\Pulse.Solibri.Server\Tools\CheckingTools.cs

`solibri.checking.run`

Checks the open Solibri project and returns a summary (number of results). `checkSelected=true` limits to selected components.

Scope: | `sources\solibri\server\Pulse.Solibri.Server\Tools\CheckingTools.cs`

`solibri.results.get`

Returns the results of the latest checking: name, description, severity, uniqueKey, ruleName, componentGuids (the components involved - hookable to `solibri.components.info`, `solibri.view.color`, `solibri.issues.create` etc.), decision (ACCEPTED/REJECTED/UNDEFINED) and `slideId` (the linked issue, if it exists). Filterable by component or rule.

Scope: | `sources\solibri\server\Pulse.Solibri.Server\Tools\CheckingTools.cs`

Clash

`solibri.clash.query`

Detects interferences/clash between two groups of components in the open model, calculated on the fly (does not require loaded rules - always works). Broad-phase on bounding box then precise intersection. Returns target/ostacolo pairs with dimensions, volume and `contactPoint` (precise centroid of the interpenetration mesh, useful for positioning a camera/viewpoint exactly on the interference point - null if not calculable). 'matched' indicates how many interferences exist in total, 'truncated' if the list was cut by limit.

Scope: | `sources\solibri\server\Pulse.Solibri.Server\Tools\ClashTools.cs`

Component

`solibri.components.info`

Returns the name, type and property set of one or more components given their GUID IFC.

Scope: | `sources\solibri\server\Pulse.Solibri.Server\Tools\ComponentTools.cs`

`solibri.components.query`

Searches or finds components (walls, pillars, beams, floors...) in the model and returns their GUID (necessary for selection, insulation, coloring and info). optional filters: type (e.g. WALL, DOOR, COLUMN - see `solibri.components.types`) and `nameContains`. 'matched' indicates how many exist in total, 'truncated' if the list was cut from limit.

Scope: | `sources\solibri\server\Pulse.Solibri.Server\Tools\ComponentTools.cs`

`solibri.components.types`

Lists the component types present in the open model with the count of each (e.g. WALL: 120, DOOR: 34). Use it to know what to look for before a query.

Scope: | `sources\solibri\server\Pulse.Solibri.Server\Tools\ComponentTools.cs`

Geometry

`solibri.classifications.classify`

Runs a classification system (from `solibri.classifications.list`) on a component and returns the corresponding codici/etichettes - e.g. "which Uniclass code does this wall belong to".

Scope: | sources\solibri\server\Pulse.Solibri.Server\Tools\GeometryTools.cs

`solibri.classifications.list`

Lists the classification systems available in the open model (e.g. Uniclass, OmniClass) with name, description, edition and source.

Scope: | sources\solibri\server\Pulse.Solibri.Server\Tools\GeometryTools.cs

`solibri.components.contains`

Check if a 3D point is really inside the volume of a component (test on the real mesh, not on the bounding box) - e.g. to check if a detected coordinate (laser scanner, sensor) really falls inside that element.

Scope: | sources\solibri\server\Pulse.Solibri.Server\Tools\GeometryTools.cs

`solibri.components.distance`

Real (non-bounding-box) distance between two componenti: fast scalar values (distance/horizontalDistance/verticalDistance) plus the exact segment of the two closest points on the real surfaces. For checking franco/clearance even when components DO NOT interpenetrate - complementary to `solibri.clash.query`, which only covers overlap cases.

Scope: | sources\solibri\server\Pulse.Solibri.Server\Tools\GeometryTools.cs

`solibri.components.footprint`

2D footprint of a componente: area, perimeter, number of vertices, centroid, area net of holes. Useful for superficie/copertura questions without having to export the full 3D geometry.

Scope: | sources\solibri\server\Pulse.Solibri.Server\Tools\GeometryTools.cs

`solibri.components.line`

Components crossed by the infinite line passing through two points, extended in BOTH directions (unlike `solibri.components.ray`, which goes in only one direction from an origin) - useful for the alignment axis of a grid or a pipe.

Scope: | sources\solibri\server\Pulse.Solibri.Server\Tools\GeometryTools.cs

`solibri.components.mesh`

Real triangulated geometry of a componente: area, volume, bounding box and triangoli/vertici count - independent of any declared IFC quantities. Useful for cross-checking when the source software did not export the quantities, or to verify them. Does not return the raw mesh (no vertice/triangolo listed).

Scope: | sources\solibri\server\Pulse.Solibri.Server\Tools\GeometryTools.cs

`solibri.components.ray`

Components crossed by a ray with arbitrary origin and direction (only one direction, from the origin onwards) - for checks of free verticale/orizzontale, e.g. "What do I encounter when climbing from this point up to the ceiling."

Scope: | sources\solibri\server\Pulse.Solibri.Server\Tools\GeometryTools.cs

`solibri.components.region`

Components intersecting an arbitrary 3D volume (min/max), independent of an existing component - unlike `solibri.clash.query`, which always starts from the boundaries of a target component. For a construction zone, a buffer around a point, or any volume not modeled as Space IFC.

Scope: | sources\solibri\server\Pulse.Solibri.Server\Tools\GeometryTools.cs

`solibri.components.related`

IFC relationship graph of a component - e.g. VOIDS/FILLS for opening<->host wall, SERVES for terminal MEP<->system that powers it, DEFINES_BY_TYPE for instance<->type, CONTAINS for composition hierarchies. One relationship at a time.

Scope: | sources\solibri\server\Pulse.Solibri.Server\Tools\GeometryTools.cs

Issues

`solibri.issues.comments_add`

Adds a textual comment to an existing issue - useful for recording a check or a decision made (e.g. "confirmed during inspection of the 27/08").

Scope: | sources\solibri\server\Pulse.Solibri.Server\Tools\IssuesTools.cs

`solibri.issues.comments_list`

Lists textual comments on an issue (from `solibri.issues.list`).

Scope: | sources\solibri\server\Pulse.Solibri.Server\Tools\IssuesTools.cs

`solibri.issues.create`

Creates a new issue (Slide) in the open project, optionally linked with e/o components to a checking result. It is added to a Presentation (by default "Pulse Issues", reused if it already exists) - that's what makes it visible in the Solibri Issues panel, not just an object in memory.

Scope: | sources\solibri\server\Pulse.Solibri.Server\Tools\IssuesTools.cs

`solibri.issues.delete`

Delete an issue (Slide) by removing it from each Presentation to which it belongs, given its `uniqueId` (from `solibri.issues.list`).

Scope: | sources\solibri\server\Pulse.Solibri.Server\Tools\IssuesTools.cs

`solibri.issues.list`

Lists Solibri native issues (Slides) in the open project - the same as the Issues panel. scope: all (all), selected (selected in the interface), marked (manually marked by the user, typically for a targeted export).

Scope: | sources\solibri\server\Pulse.Solibri.Server\Tools\IssuesTools.cs

`solibri.issues.update`

Updates an existing issue given its `uniqueId` (from `solibri.issues.list`). Change only the past fields.

Scope: | sources\solibri\server\Pulse.Solibri.Server\Tools\IssuesTools.cs

`solibri.issues.viewpoint_add`

Capture the CURRENT 3D view of Solibri (camera state + image snapshot) and attach it to an issue as visual proof - use it immediately after framing the item in question with `solibri.view.look_at` or `solibri.view.camera_set`.

Scope: | `sources\solibri\server\Pulse.Solibri.Server\Tools\IssuesTools.cs`

Ito

`solibri.ito.definitions`

Lists the Information Takeoff definitions (calculations/quantities) present in the Solibri project.

Scope: | `sources\solibri\server\Pulse.Solibri.Server\Tools\ItoTools.cs`

`solibri.ito.table`

Returns the current ITO table (columns, rows, row total). With `calculate=true` recalculate all takeoffs first (potentially long operation).

Scope: | `sources\solibri\server\Pulse.Solibri.Server\Tools\ItoTools.cs`

Library

`solibri.library.list`

Lists the rulesets AVAILABLE in the Solibri library (.cset file on disk), i.e. those that can be loaded from the Ruleset Manager - not necessarily already open in the project. Optional filter for nome/cartella. NB: to see the RULES inside a ruleset and their configuration you must first load it into Solibri (action GUI), then use `solibri.rules.search` / `solibri.rules.describe`.

Scope: | `sources\solibri\server\Pulse.Solibri.Server\Tools\LibraryTools.cs`

Rule

`solibri.rules.get`

Returns the parameters of a rule given its ref (from the 'ref' of `solibri.rulesets.list` field): id, type, value.

Scope: | `sources\solibri\server\Pulse.Solibri.Server\Tools\RuleTools.cs`

Rule Catalog

`solibri.rules.describe`

Describes a caricata rule: what it controls (official description) and how it is CONFIGURED NOW (parametri values: tolerances, targets, exceptions). It also works on the built-in rules of Solibri. NB: values can be read but not modified via API - the modification must be made in GUI of Solibri.

Scope: | `sources\solibri\server\Pulse.Solibri.Server\Tools\RuleCatalogTools.cs`

`solibri.rules.search`

Search among ALL the rules loaded in the project for those that correspond to a request in natural language (search in the name AND official description of the rule). Use this when the user asks 'what check can I do for X': returns candidate rules with path to ruleset and description.

Scope: | sources\solibri\server\Pulse.Solibri.Server\Tools\RuleCatalogTools.cs

Rule Parameter

`solibri.rules.set_parameter`

Sets the value of a rule parameter. With confirm=false it only returns a preview (actual vs proposed value) and does NOT write anything. With confirm=true applies setValue and returns oldValue/newValue (post-verification). Types supportati: Boolean/Int/Double/String/Enumeration.

Scope: | sources\solibri\server\Pulse.Solibri.Server\Tools\RuleParameterTools.cs

Ruleset

`solibri.rulesets.export_json`

Returns the native JSON serialization of a ruleset (Ruleset.toJson) plus its description. It is the only public surface that could contain the configured VALUES of the built-in rule parameters.

Scope: | sources\solibri\server\Pulse.Solibri.Server\Tools\RulesetTools.cs

`solibri.rulesets.get`

Returns the subtree of a ruleset given its uniqueId.

Scope: | sources\solibri\server\Pulse.Solibri.Server\Tools\RulesetTools.cs

`solibri.rulesets.list`

Lists the rulesets of the open Solibri project, with the rules tree (uniqueId, name, isRule, children).

Scope: | sources\solibri\server\Pulse.Solibri.Server\Tools\RulesetTools.cs

Selection

`solibri.selection.get`

Returns the GUID IFC of the currently selected components - the current selection (Selection Basket) of Solibri. Use it to know what is selected now.

Scope: | sources\solibri\server\Pulse.Solibri.Server\Tools\SelectionTools.cs

`solibri.selection.set`

Sets the Selection Basket to components with the given GUID IFC (overrides the current selection).

Scope: | sources\solibri\server\Pulse.Solibri.Server\Tools\SelectionTools.cs

Settings

`solibri.bcf.servers`

Lists configured BCF-API servers (e.g. BIMcollab, Trimble Connect) for live issue synchronization - distinct from `solibri.bcf.export` which produces a static file. lettura only: Authentication requires an interactive browser login.

Scope: | `sources\solibri\server\Pulse.Solibri.Server\Tools\SettingsTools.cs`

`solibri.layouts.list`

Lists tab layouts saved in the Solibri interface (e.g. "3D + Info") with name and number of panels.

Scope: | `sources\solibri\server\Pulse.Solibri.Server\Tools\SettingsTools.cs`

`solibri.layouts.switch`

Change the active tab layout in the Solibri interface (from `solibri.layouts.list`).

Scope: | `sources\solibri\server\Pulse.Solibri.Server\Tools\SettingsTools.cs`

`solibri.settings.format`

Format a number exactly as the user would see it in the Solibri interface (units and locale configured) - use this to present `lunghezze/aree/volumi/percentuali/angoli` returned by other tools instead of showing the raw number.

Scope: | `sources\solibri\server\Pulse.Solibri.Server\Tools\SettingsTools.cs`

Spatial

`solibri.components.door`

Typical details of a door (DOOR component): overall width and height, opening type (e.g. `SINGLE_SWING_LEFT`, `SLIDING_TO_RIGHT`) - data not available via generic `solibri.components.info`.

Scope: | `sources\solibri\server\Pulse.Solibri.Server\Tools\SpatialTools.cs`

`solibri.components.quantities`

Standard IFC quantities of a component (area, volume, length, width, height, thickness, perimeter, diameter etc. - 23 possible types), recalculated by Solibri itself WITHOUT having to first set up a table ITO - immediate response to "quant" It is the thickness of this wall". Returns only the types that the component actually has.

Scope: | `sources\solibri\server\Pulse.Solibri.Server\Tools\SpatialTools.cs`

`solibri.components.site`

Typed details of the SITE component: elevation and geographic coordinates (gradi/primi/secondi) of the model's IFC reference point - useful for verifying that federated models from different disciplines share the same geographic reference.

Scope: | `sources\solibri\server\Pulse.Solibri.Server\Tools\SpatialTools.cs`

`solibri.components.window`

Typed details of a window (WINDOW component): partitioning type (pane scheme) - data not available via generic `solibri.components.info`.

Scope: | `sources\solibri\server\Pulse.Solibri.Server\Tools\SpatialTools.cs`

`solibri.spaces.summary`

Summary of a room (SPACE component): GUID of connected porte/finestre/aperture, total area by boundary element type (walls, windows, doors, columns, beams, floors, ceilings), and `spaceBoundaries` - the per-boundary detail (area, type PHYSICAL/VIRTUAL, connected real component) to identify unmodeled VIRTUAL boundaries that distort calculations. Data not available via generic `solibri.components.info`.

Scope: | `sources\solibri\server\Pulse.Solibri.Server\Tools\SpatialTools.cs`

`solibri.storeys.components`

Lists the components present on a given piano/livello (GUID from `solibri.storeys.list`) - e.g. "what's on the floor 3". Optional filter by type (see `solibri.components.types`).

Scope: | `sources\solibri\server\Pulse.Solibri.Server\Tools\SpatialTools.cs`

`solibri.storeys.list`

Lists all piani/livelli (BUILDING_STOREY) in the open model, with name, elevation, and elevation, and how many components each contains.

Scope: | `sources\solibri\server\Pulse.Solibri.Server\Tools\SpatialTools.cs`

System

`solibri.project.scores`

History of the project's copertura/qualita score (percentage of components passed, percentage of rules verified), calculated and maintained by Solibri itself - for a trend over time instead of just an instantaneous value.

Scope: | `sources\solibri\server\Pulse.Solibri.Server\Tools\SystemTools.cs`

`solibri.system.info`

Status of Solibri: whether it is busy (busy, i.e. it is still checking - useful to check before the next command), the connected user, the path and saving status of the project (NO_FILE/UNSAVED/SAVED/MODIFIED - check it before suggesting risky actions), the header IFC of the model (schema, author, source software) and the federated models that compose it (name, UUID, source software).

Scope: | `sources\solibri\server\Pulse.Solibri.Server\Tools\SystemTools.cs`

View

`solibri.view.camera.get`

Returns the 3D camera state of Solibri: location, direction and upVector (model coordinates).

Scope: | `sources\solibri\server\Pulse.Solibri.Server\Tools\ViewTools.cs`

`solibri.view.camera.set`

Set the Solibri 3D camera. You can pass locations e/o upVector (if omitted they remain unchanged); the direction is NOT settable via API. Returns the resulting state (post-verification).

Scope: | sources\solibri\server\Pulse.Solibri.Server\Tools\ViewTools.cs

`solibri.view.color`

Colors the components with the given GUID in the 3D view. Color ARGB 0-255 (default matte red).

Scope: | sources\solibri\server\Pulse.Solibri.Server\Tools\ViewTools.cs

`solibri.view.hide`

Hides components with the given GUID in the 3D view.

Scope: | sources\solibri\server\Pulse.Solibri.Server\Tools\ViewTools.cs

`solibri.view.isolate`

Isolates only the components with GUID dati in the 3D view: show everything, then hide everything else.

Scope: | sources\solibri\server\Pulse.Solibri.Server\Tools\ViewTools.cs

`solibri.view.look_at`

Frame the components with the given GUID in the 3D view (zoom-to-fit). To be used after a isolamento: otherwise the camera remains where it was and can look into space.

Scope: | sources\solibri\server\Pulse.Solibri.Server\Tools\ViewTools.cs

`solibri.view.show`

Shows the components with the given GUID in the 3D view (without touching the others).

Scope: | sources\solibri\server\Pulse.Solibri.Server\Tools\ViewTools.cs

`solibri.view.show_all`

Makes all components visible in the 3D view (undoes isolations and hiding).

Scope: | sources\solibri\server\Pulse.Solibri.Server\Tools\ViewTools.cs

`solibri.view.snapshot`

Captures the current 3D view to a PNG file on disk and returns the path.

Scope: | sources\solibri\server\Pulse.Solibri.Server\Tools\ViewTools.cs

Verified source

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

main / bc7fd4c5041832a48c62c1b0526fa08255ba02e5

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