ArchPlug

DETAILED INSTRUCTIONS

Auto Lintel Beams

Automate placement and scheduling of lintel beams above wall openings.

For Autodesk Revit 2022–2026 · Windows

archplug.com · July 2026

Overview

Auto Lintel Beams places structural lintel beams above every door and window opening in your Revit model — automatically, in a single click. It scans the entire project for doors and windows, reads each opening's width and top elevation, and places a Structural Framing beam of your chosen type directly above it, extended by a configurable bearing length on each side.

The plugin is fully **update-aware**: run it again after moving, resizing, adding or deleting openings, and it updates every beam to match — repositioning moved beams, creating beams for new openings, and removing beams whose openings no longer exist. It never creates duplicates.

The settings window is **modeless**, meaning it floats above Revit while you keep working: you can pan, zoom, switch views, move openings, and press **RUN** again at any time without reopening the tool.

Key Benefits & Use Cases

- **Massive time savings** — model hundreds of lintels across a whole building in seconds instead of placing beams one by one.
- **Consistent engineering logic** — every lintel gets the same bearing extension and elevation rules, per your settings, with separate values for doors and windows.
- **Wall-type-aware overrides** — assign a different beam type per wall type (e.g., a wider lintel for thick exterior walls, a slim one for partition walls) in a simple table.
- **Design-change resilient** — re-run after every design iteration; beams follow their openings. Deleted openings automatically lose their beams; duplicates are cleaned up.
- **Office standards via templates** — save your settings (beam types, extensions, wall-type overrides) to a JSON template file and share it across projects and colleagues.
- **Typical users**: architects preparing structural coordination models, structural engineers doing preliminary lintel layouts, BIM managers enforcing consistent lintel modeling standards.

Requirements

- **Autodesk Revit 2022–2026** (a build is provided for each supported version).
- **Windows** operating system (Revit itself is Windows-only).
- Auto Lintel Beams is part of the **ArchPlug suite** of Revit plugins and installs alongside the shared ArchPlug components.

- Your project must contain at least one **Structural Framing family** (beam family). The plugin uses beam types already loaded in your project — you stay in full control of which lintel family and dimensions are used.

Installation

1. Close Revit.
2. Run the downloaded **.msi installer** and follow the wizard.
3. Start Revit and open a project.
4. Find the **ArchPlug** tab on the Revit ribbon — Auto Lintel Beams is launched from there.

No additional configuration is required. Your settings are stored per Windows user and survive plugin updates.

User Interface Walkthrough

The plugin has a single window: **Lintel Beam Settings**.

Window behavior

- **Modeless and always on top** — the window floats above Revit (including floated view windows on other monitors). You can keep working in the model while it is open.
- **Resizable** — drag any edge; the beam-type table grows with the window. A sensible minimum size is enforced.
- **Remembers position and size** — the window reopens exactly where you left it. If that position would be off-screen (e.g., a monitor was disconnected), it safely re-centers.
- **Closing** — use the **X** button or press **Esc**. There is no Cancel button; all current values are saved automatically on close.
- **Enter** triggers **RUN**. **Ctrl+Z / Ctrl+Y** undo and redo your edits to the input fields inside the window.
- Running does **not** close the window — adjust values and run again as often as you like.

Main settings (top section)

Control	What it does
Default beam type:	Dropdown listing every Structural Framing type loaded in the project, in the format `Family : Type`, sorted alphabetically. This type is used for all lintels unless a wall-type override applies.
Extra width each side - windows (unit):	Bearing extension added to **each side** of a window opening. A 100-wide window with 20 extra gives a 140-long beam. Default: 20.
Extra width each side - doors (unit):	Same as above, for doors. Default: 20.
Raise above opening - windows (unit):	Vertical gap between the top of a window opening and the underside of its beam. Default: 0 (beam sits directly on top of the opening).
Raise above opening - doors (unit):	Same as above, for doors. Default: 0.

The **(unit)** shown in each label follows your **project's length display unit** — mm, cm, dm, m, in, or ft. Enter values in that unit. Both `.` and `,` are accepted as the decimal separator. Negative raise values are allowed (they lower the beam relative to the opening top).

Special types (override default by wall type)

Below the numeric fields, the section headed **Special types (override default by wall type):** contains a table with two dropdown columns:

Column	What it does
Special wall type	Pick a wall type from the project (all wall types are listed alphabetically).
Beam type	Pick the Structural Framing type (`Family : Type`) to use for every opening hosted in a wall of that type.

Any door or window hosted in a wall whose type name matches a row here receives that row's beam type instead of the default. Overrides apply equally to doors and windows.

Table buttons:

- **ADD ROW** — appends an empty row and selects it for editing.
- **DELETE ROW** — removes the currently selected row.

Rows where the chosen beam type no longer exists in the project are skipped at run time (the default type is used for those walls instead).

Footer buttons

- **SAVE TEMPLATE** — opens a save dialog (*Save Lintel Template*, JSON files) and exports the current settings — default beam type, all four numeric values, and the full wall-type override table — to a `.json` file (default name `lintel_template.json`).
- **IMPORT TEMPLATE** — opens a file dialog (*Import Lintel Template*) and loads a previously saved template, filling all fields and replacing the override table.
- **Hide all beams** — hides every beam created by this plugin in the **active view** (to declutter a plan or elevation). Beams that the view refuses to hide are skipped.
- **Unhide all beams** — reveals them again in the active view.
- **RUN** — validates your inputs, saves the settings, and creates/updates all lintel beams. The window stays open.

Step-by-Step Workflow

1. Make sure a suitable **beam (Structural Framing) family** is loaded in your project. If none is present, the plugin will tell you: "*No Structural Framing families found in the model. Please load a beam family first.*"
2. Launch **Auto Lintel Beams** from the **ArchPlug** ribbon tab. The **Lintel Beam Settings** window opens.
3. Select your **Default beam type**.
4. Enter the **Extra width each side** values for windows and doors (the bearing length on each side of the opening).
5. Enter the **Raise above opening** values if you want a gap between the opening top and the beam (leave 0 to seat the beam directly on the opening).
6. (Optional) Add wall-type overrides: click **ADD ROW**, pick a **Special wall type** and its **Beam type**. Repeat for each wall type that needs a different lintel.
7. (Optional) Click **SAVE TEMPLATE** to store these settings for reuse, or **IMPORT TEMPLATE** to load office-standard settings.
8. Click **RUN**. The plugin processes every door and window in the model in one Revit transaction.
9. Inspect the result in a 3D view or section. Use **Hide all beams** / **Unhide all beams** to control their visibility in the active view while you work.
10. Whenever openings change — moved, resized, added, deleted, or re-hosted — simply click **RUN** again. Beams are updated in place, recreated at new positions, created for new openings, and deleted for removed openings.
11. Close the window with **X** or **Esc** when done. All settings, plus window position and size, are remembered for next time.

One click of Revit's **Undo** reverses an entire run (all created, updated, and deleted beams), because everything happens in a single transaction named "*Create or update lintel beams*".

Feature Reference

Opening detection

- The plugin collects **all doors and windows** in the entire model (family instances in the Doors and Windows categories), on every level. There is no selection step — it always processes the whole project.
- If the model contains none, it reports: "*No doors or windows found in the model.*"

Opening width measurement

For each opening, the width is determined in this order:

1. Standard width parameters on the **instance** (Width / door width / window width, plus common names such as `Width`, `b`, `B`).
2. The same parameters on the opening's **type**.
3. Fallback: the opening's **bounding box projected onto the wall direction** — so even families without a conventional Width parameter get a correctly sized beam.

Openings measuring **10 cm or less** are considered invalid and are skipped, as are openings whose geometry cannot be resolved.

Beam placement geometry

- **Direction**: the beam runs along the **host wall's centerline direction**. If the host curve is unavailable, the opening's own orientation is used as a fallback. Curved walls are handled via the wall's end-to-end direction — the beam itself is always straight.
- **Length**: opening width + 2 × the applicable *Extra width each side* value (doors and windows each use their own value).
- **Horizontal position**: centered on the opening.
- **Elevation**: top of the opening + the applicable *Raise above opening* value.
- **Level**: each beam is placed on (and its reference level set to) the level of its opening — resolved from the opening itself, its host wall, or its level parameter.

Height-aware vertical justification

The plugin reads the beam type's section height from standard structural section parameters or common names (`h`, `H`, `d`, `D`, `Height`, `Section Height`, `Beam Height`):

- **Height known** — the beam is placed with **Z Justification = Center** and its axis raised by half the section height, so the beam's **underside sits exactly at the opening top + raise**.

This is the geometrically correct result regardless of section size.

- **Height unknown** — the beam uses **Z Justification = Bottom** at the opening top + raise, achieving the same underside alignment.

In both cases the beam's **Z Offset is set to 0**, so manual offsets never accumulate.

Wall-type overrides (Special types)

- Each opening's host wall type name is checked against the override table; the first matching row's beam type is used, otherwise the default type.
- Overrides apply to doors and windows alike.
- If a row references a beam type that has been removed from the project since the settings were saved, that row is silently skipped and the default type is used.

Smart update — no duplicates, ever

Every beam the plugin creates is tagged in its **Comments** parameter with an identifier of the form `ARCHPLUG_LB:<opening id>` linking it to its opening. On each run:

- **Unchanged opening** → the existing beam is kept and refreshed (type, reference level, justification, mark). Counted as *updated*.
- **Moved or resized opening** → the old beam is deleted and a new one is created at the correct position (structural framing cannot be reliably relocated in place). Counted as *recreated*.
- **Beam type change needed but not possible in place** → delete + recreate as well.
- **New opening** → a new beam is created.
- **Opening deleted** → its now-orphaned beam is deleted.
- **Multiple beams found for one opening** (e.g., from copy/paste) → the extras are deleted.
- Changing the default or override beam family between runs is safe — beams from earlier runs are still recognized as the plugin's own, because identification is by the Comments tag, not by family.

Important: do not edit or clear the `ARCHPLUG_LB:` text in a plugin-created beam's Comments parameter. That tag is how the plugin recognizes its own beams; without it, the beam will no longer be updated or cleaned up (and a duplicate may be created on the next run).

Marks and legacy compatibility

- The opening's **Mark** value is written into a beam parameter named **"Mark od prozora ili vrata"** if your beam family contains a text parameter with that name (used by earlier versions of the plugin). If the family has no such parameter, nothing is written — no error occurs. You can use this parameter in schedules to relate each lintel to its opening's mark.
- Beams created by older plugin versions (identified only by that mark parameter) are **automatically migrated** to the new Comments-tag system on the first run, provided the

mark maps to exactly one opening. Ambiguous marks (shared by several openings) are left untouched.

Hide / Unhide in the active view

- **Hide all beams** and **Unhide all beams** act on every plugin-created beam (found via the Comments tag), in the **currently active view only**, using Revit's standard Hide/Unhide Elements mechanism.
- Elements that cannot be hidden in the active view are skipped.
- Each operation is its own transaction ("*Lintel: Hide/Unhide all beams*"), so it can be undone with Revit Undo.

Templates

- **SAVE TEMPLATE** writes a plain JSON file containing: default beam type, both extra-width values, both raise values, and all wall-type override rows.
- **IMPORT TEMPLATE** loads such a file and applies it to the window. A default type or override entry that doesn't exist in the current project is simply not selected/skipped — nothing breaks when moving templates between projects.
- Window position/size is not part of a template.

Settings persistence

- All values (including the override table and window geometry) are saved automatically when you close the window and every time you press RUN. They are stored per Windows user profile (under `%LOCALAPPDATA%\ArchPlug\Lintel`), so they persist across sessions, projects, and plugin updates.
- If the settings file cannot be written, a one-time dialog ("*Lintel Beams - Save failed*") shows the path and the reason; the plugin keeps working normally for the current session.

Quiet, non-interruptive execution

- Revit **warnings** raised during beam creation (e.g., "beam slightly off axis" or overlap warnings) are automatically dismissed so a large run is never interrupted by warning pop-ups. Genuine errors still stop and roll back the run with an error dialog.
- There is **no completion pop-up** — when RUN finishes silently, the beams are in the model.

Project units

All numeric fields are interpreted in your **project's length display unit**, and the field labels show which unit that is. Switching the project's units changes the labels and interpretation accordingly on the next launch.

Tips & Best Practices

- **Load a dedicated lintel family first.** Any Structural Framing family works, but a simple rectangular concrete/steel lintel family with a proper section height parameter gives the best results (the plugin can then seat the beam underside exactly on the opening).
- **Use a beam type whose section height is defined** in a standard parameter — you get exact underside-on-opening placement via center justification.
- **Set realistic bearing lengths.** The default 20 (in cm-based projects) on each side matches common practice for masonry lintels; adjust per your local standards, separately for doors and windows.
- **Use wall-type overrides for wall thickness.** Create one beam type per wall thickness (e.g., "Lintel 25", "Lintel 38") and map each wall type to its lintel in the override table — one run handles the whole building correctly.
- **Save a template per office standard** and import it at the start of every project — including the full wall-type mapping.
- **Re-run early, re-run often.** After any door/window change, one click of RUN re-synchronizes all lintels. Keep the window open while you iterate — it is modeless.
- **Don't manually move plugin-created beams.** The next run will snap them back to their opening (by delete/recreate). If a specific opening genuinely needs a manual lintel, model it yourself with a beam that has no `ARCHPLUG_LB:` tag in Comments — the plugin will not touch it, though it will still place its own beam for that opening.
- **Schedule your lintels** with a Structural Framing schedule; filter by the Comments parameter beginning with `ARCHPLUG_LB` to isolate plugin-created lintels, and use the "Mark od prozora ili vrata" parameter (if present in your family) to reference each opening's mark.
- **One Revit Undo reverses a whole run** — experiment freely with settings.

Troubleshooting & FAQ

"No Structural Framing families found in the model. Please load a beam family first." The project has no beam families. Load any Structural Framing family (Insert → Load Family, or from your template), then relaunch the tool.

"No doors or windows found in the model." The model contains no door or window family instances. Note that openings modeled as wall openings, generic models, or in linked files are not detected — only elements in the Doors and Windows categories of the current model.

"Default beam type not found: ..." The saved/selected beam type no longer exists in the project (it was deleted or renamed). Re-open the dropdown and pick a valid type, then RUN again.

"Extra width - windows must be a number." (or the equivalent for the other fields) The highlighted field contains non-numeric text. Enter a plain number; both `.` and `,` decimals are accepted.

Some openings did not get a beam. An opening is skipped when its level cannot be determined, its width cannot be measured, its width is 10 cm or less, or its direction/top elevation cannot be resolved (unusual custom families). Check that the family is wall-hosted and has sensible geometry or a Width parameter.

Beams appear at the wrong height (e.g., centered on the opening top instead of sitting on it). The beam type's section height could not be read, so bottom justification was used — which still aligns the underside — or your family's geometry is not symmetric about its axis. Prefer families whose section height is stored in a standard structural section parameter or a parameter named `h`, `Height`, or similar.

I moved a door and the beam didn't follow. Beams update only when you press **RUN** — the plugin does not monitor the model live. Run it again after changes.

I get duplicate beams above one opening. This can only happen if the plugin's tag was removed from a beam's Comments, or the beam was copied manually. Run the tool once — it deletes surplus tagged beams itself; untagged manual copies must be deleted by hand.

"Hide all beams" seems to do nothing. Hide/Unhide affects only the **active view**, and only beams the view allows to be hidden (visible category, not already hidden, view supports hiding). Also, beams are only recognized if their Comments tag is intact.

The window opened off-screen or oddly sized after changing monitors. The plugin validates the saved position and re-centers if it would be invisible, and corrects sizing when moving between monitors with different DPI scaling. If it ever looks wrong, close and reopen the window.

"Lintel Beams - Save failed" appears once. Windows blocked writing the settings file (shown in the message, under your user's local application data). Your inputs still work for the session; check folder permissions or antivirus if it persists.

Does it work on openings in linked models? No. Only doors and windows in the active model are processed.

Does it place lintels over plain wall openings (Opening by Face / Wall Opening)? No. Detection is based on the Doors and Windows categories.

Can I undo a run? Yes — one Revit Undo step reverses the entire run (creation, updates, and deletions).