Uchuva Documentation

Compiled documentation for installation, usage, commercial release, support, privacy, and legal reference.

README.md

Uchuva

Work toolkit (PyRevit-style grouping): Python MCP connector, Revit add-in, and a dedicated local HTTP bridge.

Project location

- Typical path: `Documents\code\plugins\Uchuva`
- Example absolute path: `C:\Users\<you>\Documents\code\plugins\Uchuva`

The MCP entry point is `server.py` at the **root** of this folder. Batch scripts use `%~dp0`, so they keep working if you move the folder; re-run Claude/Cursor setup if your MCP config still points to an old path.

Requirements

- Windows
- uv (batch scripts can install it if missing)

Setup

- Double-click `setup.bat` (one step: `uv sync` + register MCP in **Cursor** and **Claude Desktop**). Or run `uv sync` manually, then `setup.bat`.
- Restart **Cursor** (fully quit) and **Claude Desktop** if you use it. The MCP server id is `uchuva`, with tools `uchuva_ping` and `uchuva_revit_health` (GET `/health` on the Revit add-in bridge).

`install.bat` is a shortcut that runs `setup.bat`.

Optional environment variable: `UCHUVA_REVIT_URL` (default `http://localhost:8775`, the Uchuva Revit add-in port).

If you moved the folder

Re-run `setup.bat` so `claude_desktop_config.json` and `mcp.json` get the new paths.

Layout

Path	Description
`server.py`	MCP stdio entry
`uchuva/`	Editable Python package
`uchuva/revit_client.py`	HTTP client (`httpx`) to the Revit add-in

Path	Description
`uchuva/tools/`	Optional extension modules
`scripts/`	MCP registration helpers (`setup_cursor.ps1`, `setup_anthropic.ps1`)
`docs/`	Product and distribution notes (`CHECKLIST.md`, `DISTRIBUTION.md`, `COMMERCIAL_RELEASE.md`, `TOOLS.md`)
`revit/`	Revit add-in (.NET), standalone HTTP bridge
`tests/`	Python unit tests and bridge response fixtures

Revit add-in (Uchuva)

- Visual Studio solution: `revit/Uchuva.sln` (projects `Uchuva.Bridge`, `Uchuva.Revit`).
- **Debug** build copies outputs to `%AppData%\Autodesk\Revit\Addins\{year}\Uchuva\` and deploys `Uchuva.addin` next to other add-ins for that year.
- **Default port:** `8775` (Uchuva-only; avoids clashes with other local bridges).
- Revit ribbon tab **Uchuva**: **AI Connector** (shows the base URL) and **Schedule Exporter** (Excel/CSV).
- Close Revit before rebuilding if DLLs are locked.

Revit data tools (MCP)

With Revit open and the Uchuva add-in loaded, these tools call the localhost HTTP API:

MCP tool	Purpose
`uchuva_model_info`	Project path, element count, Revit version, worksharing
`uchuva_project_summary`	Compact AI-ready context bundle for the active model
`uchuva_model_health_summary`	Warning/link/workset/sheet health snapshot
`uchuva_list_categories`	Category names with instance counts
`uchuva_list_levels`	Levels and elevations
`uchuva_list_rooms`	Rooms with areas (requires rooms in model; capped by `limit`)
`uchuva_list_elements`	Sample elements (optional `category`, `limit`, `offset`)
`uchuva_element_summary`	One element by `element_id`
`uchuva_element_parameters`	Instance + type parameters for `element_id`
`uchuva_search_elements`	Filter by category, level, parameter name/value, name, family, type
`uchuva_select_elements`	Select elements in the active Revit UI
`uchuva_zoom_to_elements`	Select and zoom/focus elements in the active Revit UI
`uchuva_clear_selection`	Clear the active Revit UI selection
`uchuva_get_current_selection`	Return current UI selection with compact element summaries
`uchuva_select_by_search`	Search by BIM filters and select matching elements
`uchuva_zoom_to_search`	Search by BIM filters, select, and zoom to matching elements
`uchuva_select_by_category`	Select elements by category with a safety limit
`uchuva_zoom_to_warning`	Zoom to elements associated with a Revit warning
`uchuva_select_sheet_views`	Select views placed on a sheet
`uchuva_open_view`	Activate a Revit view by id or name
`uchuva_zoom_to_room`	Find and zoom to room elements by number/name/level
`uchuva_selection_summary`	Summarize current selection by category, level, family, and type

MCP tool	Purpose
`uchuva_selection_parameters`	Inspect parameter values across current selection
`uchuva_list_warnings`	Warning totals and counts per description
`uchuva_list_links`	Linked RVT paths, loaded/pinned
`uchuva_list_worksets`	Worksets (workshared docs)
`uchuva_list_sheets`	Sheet list with metadata, `limit` and `offset`
`uchuva_list_views`	Printable non-template views, `limit` and `offset`
`uchuva_list_schedules`	Schedule views (non-templates), `limit` and `offset`
`uchuva_list_families`	Family/type usage summary for loaded families
`uchuva_list_titleblocks`	Titleblock family types available for sheet creation
`uchuva_naming_audit`	Sheet/view/schedule naming audit with optional prefix check
`uchuva_schedule_column_sum`	Sum a numeric column in a schedule body
`uchuva_schedule_export_json`	Read schedule body rows as capped JSON for AI analysis
`uchuva_schedule_export_excel`	Export named schedules to one `.xlsx` (optional Cover sheet)
`uchuva_schedule_export_csv`	Export one schedule to UTF-8 `.csv`
`uchuva_update_instance_parameter`	Controlled write tool with dry-run default and explicit confirmation
`uchuva_batch_update_parameters`	Controlled batch parameter updates, capped at 50 changes
`uchuva_duplicate_view`	Controlled view duplication with dry-run and confirmation
`uchuva_rename_view`	Controlled view rename with duplicate-name validation
`uchuva_rename_sheet`	Controlled sheet number/name update
`uchuva_create_sheet_from_titleblock`	Controlled sheet creation from an existing titleblock type

HTTP routes include: `GET /model/info`, `/model/project-summary`, `/model/health-summary`, `/model/categories`, `/model/levels`, `/model/rooms`, `/model/warnings`, `/model/links`, `/model/worksets`, `/model/sheets`, `/model/views`, `/model/schedules`, `/model/families`, `/model/titleblocks`, `/model/naming-audit`, `/schedule/column-sum`, `/schedule/export-json`, `/elements`, `/elements/{id}`, `/elements/{id}/parameters`, `/search?...`, `/ui/selection`, `/ui/selection-summary`, `/ui/selection-parameters`, `POST /ui/select-elements`, `POST /ui/zoom-to-elements`, `POST /ui/clear-selection`, `POST /ui/select-by-search`, `POST /ui/zoom-to-search`, `POST /ui/select-by-category`, `POST /ui/zoom-to-warning`, `POST /ui/select-sheet-views`, `POST /ui/open-view`, `POST /ui/zoom-to-room`, `POST /schedule/export-excel`, `POST /schedule/export-csv`, `POST /elements/update-parameter`, `POST /elements/batch-update-parameters`, `POST /views/duplicate`, `POST /views/rename`, `POST /sheets/rename`, and `POST /sheets/create` (JSON body).

Large list routes use caps and pagination where useful. The Python client also keeps a short-lived metadata cache for categories, levels, sheets, views, schedules, and worksets. Configure with `UCHUVA_REVIT_TIMEOUT` and `UCHUVA_REVIT_CACHE_TTL`.

The Revit bridge writes a local session token to `%LOCALAPPDATA%\Uchuva\bridge-token.txt`. MCP requests send it as `X-Uchuva-Token`; `/health` remains open for diagnostics. Request logs go to `%LOCALAPPDATA%\Uchuva\logs\bridge.log` with UTC timestamp, method, path, status, duration in milliseconds, and response size. Write tools also append an audit line to `%LOCALAPPDATA%\Uchuva\logs\write-audit.log`. Model-changing tools default to `dry_run=true`; commit only with `dry_run=false` and `confirm=true`.

Architecture: MCP, HTTP, and Revit

Claude/Cursor talk to Python over **MCP on stdio**. Python cannot call the Revit API directly: Revit runs in another process; the API is only valid inside the Revit host (e.g. a .NET add-in).

The usual pattern:

- **Inside Revit:** an add-in exposes a **local HTTP bridge** (localhost).
- **In the MCP server (Python):** an HTTP client sends GET/POST to that localhost URL so tools map assistant calls to JSON the add-in understands.

A **stub** such as `uchuva_revit_health` calls your real `GET /health` endpoint and returns the payload—useful to validate wiring before adding full model tools.

Commercial distribution

See `LICENSE` (proprietary), `THIRD-PARTY-NOTICES.txt`, and `docs/DISTRIBUTION.md` (checklist: EULA, privacy, Autodesk/App Store rules—not legal advice).

Product roadmap: `docs/CHECKLIST.md` — concrete items to reach a complete Revit connector (read/write, export, licensing, quality). Tool ownership lives in `docs/TOOLS.md` and release notes live in `CHANGELOG.md`.

Troubleshooting: MCP in Cursor

- **Re-run** `setup.bat` after moving the folder or changing Python deps so `mcp.json` stays correct (`type: "stdio"`, `.venv\Scripts\python.exe` when present).
- **Global config file** Cursor reads is `%USERPROFILE%\.cursor\mcp.json`. Older setups only wrote `%APPDATA%\Cursor\User\mcp.json`; the script now always updates the global path too.
- Fully **restart Cursor** (not only reload window) after changing MCP config.
- Open **Output** → **MCP** (or “MCP Logs”) in Cursor to see startup errors (bad path, missing Python, etc.).
- In **Settings** → **Features** → **MCP**, confirm the **uchuva** server is enabled (toggle on).

FIRST_10_MINUTES.md

First 10 Minutes With Uchuva

1. Install

Run `UchuvaSetup-<version>.exe` and keep the default components unless you only need the MCP runtime.

2. Activate

Open Revit. In the Uchuva ribbon, enter the Gumroad license key when prompted. The bridge starts only after a valid license or offline grace state.

3. Register MCP

Use the Start Menu shortcut "Register Uchuva MCP (Cursor and Claude)" if the installer did not run registration after install. Restart Cursor or Claude Desktop completely.

4. Confirm Connection

Ask your AI client to run:

```
uchuva_revit_health
```

Expected result: product `Uchuva`, status `ok`, and port `8775`.

5. Ask For Model Context

Try:

```
Give me a Uchuva project summary and list the main model health issues.
```

6. Navigate Visually

Try:

```
Find doors on Level 1 and zoom to the first 10.
```

7. Export Data

Try:

```
Export the Room Schedule to CSV.
```

8. Use Writes Carefully

Write tools default to dry-run mode. Review the proposed changes first, then commit only when the tool requires both `dry_run=false` and `confirm=true`.

TOOLS.md

Uchuva Tool Inventory

This inventory groups MCP tools and bridge routes by functional domain so new features can be added consistently.

Runtime

- `uchuva_ping`: validates the MCP process.
- `uchuva_revit_health`: calls `GET /health` on the Revit bridge.

Model Context

- `uchuva_model_info`: `GET /model/info`
- `uchuva_project_summary`: `GET /model/project-summary`
- `uchuva_model_health_summary`: `GET /model/health-summary`
- `uchuva_list_categories`: `GET /model/categories`
- `uchuva_list_levels`: `GET /model/levels`
- `uchuva_list_rooms`: `GET /model/rooms`

Elements

- `uchuva_list_elements`: `GET /elements`
- `uchuva_element_summary`: `GET /elements/{id}`
- `uchuva_element_parameters`: `GET /elements/{id}/parameters`
- `uchuva_search_elements`: `GET /search`
- `uchuva_update_instance_parameter`: `POST /elements/update-parameter`
- `uchuva_batch_update_parameters`: `POST /elements/batch-update-parameters`

Revit UI Control

- `uchuva_select_elements`: `POST /ui/select-elements`
- `uchuva_zoom_to_elements`: `POST /ui/zoom-to-elements`
- `uchuva_clear_selection`: `POST /ui/clear-selection`
- `uchuva_get_current_selection`: `GET /ui/selection`
- `uchuva_select_by_search`: `POST /ui/select-by-search`
- `uchuva_zoom_to_search`: `POST /ui/zoom-to-search`
- `uchuva_select_by_category`: `POST /ui/select-by-category`
- `uchuva_zoom_to_warning`: `POST /ui/zoom-to-warning`

- `uchuva_select_sheet_views`: `POST /ui/select-sheet-views`
- `uchuva_open_view`: `POST /ui/open-view`
- `uchuva_zoom_to_room`: `POST /ui/zoom-to-room`
- `uchuva_selection_summary`: `GET /ui/selection-summary`
- `uchuva_selection_parameters`: `GET /ui/selection-parameters`

Coordination And QA

- `uchuva_list_warnings`: `GET /model/warnings`
- `uchuva_list_links`: `GET /model/links`
- `uchuva_list_worksets`: `GET /model/worksets`
- `uchuva_list_families`: `GET /model/families`
- `uchuva_list_titleblocks`: `GET /model/titleblocks`
- `uchuva_naming_audit`: `GET /model/naming-audit`

Documentation

- `uchuva_list_sheets`: `GET /model/sheets`
- `uchuva_list_views`: `GET /model/views`
- `uchuva_list_schedules`: `GET /model/schedules`
- `uchuva_schedule_column_sum`: `GET /schedule/column-sum`
- `uchuva_schedule_export_json`: `GET /schedule/export-json`
- `uchuva_schedule_export_excel`: `POST /schedule/export-excel`
- `uchuva_schedule_export_csv`: `POST /schedule/export-csv`
- `uchuva_duplicate_view`: `POST /views/duplicate`
- `uchuva_rename_view`: `POST /views/rename`
- `uchuva_rename_sheet`: `POST /sheets/rename`
- `uchuva_create_sheet_from_titleblock`: `POST /sheets/create`

Implementation Rules

- Read tools should accept `limit` and `offset` when they can return large collections.
- Write tools must default to `dry_run=true` and require explicit confirmation to commit.
- Claude should call write tools once with `dry_run=true`, explain the proposed changes, then call again with `dry_run=false` and `confirm=true` only after user approval.
- Batch write tools are capped at 50 changes per call.
- Expensive routes should return counts, caps, and `has_more` metadata when paginated.
- Bridge errors should return JSON with `error`; structured errors should also include `ok=false` and `code`.
- UI control tools change only the active Revit UI selection/focus/active view and do not modify model data.

- UI selection and zoom tools are capped at 250 elements per call and should report `missing_ids`, `invalid_ids`, and `truncated`.

DISTRIBUTION.md

Distribution checklist (commercial)

This is a **practical checklist**, not legal advice. Engage qualified counsel for sales, taxes, and marketplace contracts.

Product assets

- [ ] **English UI** for customer-facing strings (ribbon, dialogs, MCP tool descriptions, installers).
- [ ] **Versioning** aligned across Revit add-in assemblies, Python package, and release notes.
- [ ] **LICENSE** matches what you sell (proprietary vs OSS components).
- [ ] **THIRD-PARTY-NOTICES** shipped with binaries that bundle Newtonsoft.Json, etc.

Legal / policy (typical)

- [ ] **EULA** or subscription terms for end users (and resellers if applicable).
- [ ] **Privacy policy** if you collect telemetry, accounts, or personal data.
- [ ] **Support and refund** terms stated before purchase.
- [ ] Review draft templates in `docs\EULA.md`, `docs\PRIVACY.md`, and `docs\SUPPORT.md` before publishing.

Autodesk ecosystem

- [ ] Comply with **Autodesk App Store** (or Exchange) guidelines if you publish there.
- [ ] Use a stable **VendorId** / branding consistent with your legal entity.
- [ ] Do not imply Autodesk endorsement unless you have written permission.

Technical

- [ ] Document that the current installer is unsigned and may show Windows SmartScreen warnings.
- [ ] Document **localhost** HTTP behavior (port 8775 by default) in user-facing docs for security reviews.
- [ ] Commercial builds use `docs\COMMERCIAL_RELEASE.md`: compiled MCP executable, .NET obfuscation hook, checksums, and no public `.pdb`/source payload.

Windows installer (Inno Setup)

Build machine prerequisites: **.NET SDK**, **uv** on `PATH`, **Inno Setup 6** (or `ISCC.exe` on `PATH`), network access to download the **CPython embeddable** zip the first time (cached under `installer\cache\`).

- Bump `version` in `pyproject.toml` / `uchuva/__init__.py` / Revit assembly versions as usual.
- Validate before packaging:

```
.\.venv\Scripts\python.exe -m unittest discover -s tests dotnet build revit\Uchuva.sln
-p:Configuration=Debug
```

- From the repo root run:

```
powershell -ExecutionPolicy Bypass -File scripts\Build-Installer.ps1
```

- Output: `installer\output\UchuvaSetup-<version>.exe`

Revit add-in inside the .exe: the installer only contains Revit DLLs if they were staged. Run `scripts\Build-RevitReleaseArtifacts.ps1` on a machine with the available Revit SDKs installed (builds into `installer\artifacts\revit\<year>\`), *or* copy pre-built Release outputs into `installer\vendor-revit\<year>\` (see `installer\vendor-revit\README.txt`). If a supported year such as **2026** is not installed on the build machine, the scripts stage it with the latest compatible payload that was built so the installer still offers that year to users who have it installed. If you use `Build-Installer.ps1 -SkipRevitBuild` without any artifact/vendor folders, the setup will be **MCP-only** (no Revit plugin). For release CI, pass `-RequireRevitAddinPayload` to fail the build when no add-in was staged.

The standard script stages an **embedded Python** runtime plus `uv pip install` of the `uchuva` package, then compiles the installer. The commercial path uses `scripts\Build-CommercialRelease.ps1` to compile `uchuva-mcp.exe`, stage the compiled payload, optionally run obfuscation, and create SHA256 checksums.

Revit years supported match the solution: **2024–2026** (`net48` add-in). The wizard lets users choose **Full**, **MCP runtime only**, or **Custom** (per-year Revit add-ins).

Upgrades: keep the same **AppId** in `installer\Uchuva.iss`; raise `MyAppVersion` via `pyproject.toml` when invoking `Build-Installer.ps1` (`/DMyAppVersion=`). Inno will update an existing install in the same directory when the user runs the new `UchuvaSetup-*.exe`.

Uninstall: appears in **Settings** → **Apps** as *Uchuva*; uninstall removes files installed by Setup (including Revit add-in files under ProgramData for selected years).

Without digital signing, SmartScreen may show an “unrecognized app” warning. For the current release, publish the SHA256 checksum on Gumroad and tell users the installer is unsigned.

MCP registration: post-install runs `scripts\Register-UchuvaMcp.ps1`, which merges an `uchuva` entry into `mcpServers` in `%USERPROFILE%\.cursor\mcp.json` (and any existing Cursor/Claude config paths the scripts already discover). **Other MCP servers are left intact** (plain JSON only; files with `//` comments will fail to parse). Fully quit and reopen **Cursor** (and Claude Desktop if used) so the connector list reloads.

COMMERCIAL_RELEASE.md

Commercial Release Guide

This guide defines the protected Uchuva release path for Gumroad distribution.

Protection Stack

- .NET add-in: start with a free MIT-licensed obfuscator such as Obfuscar or a maintained ConfuserEx fork. Keep Revit entry points excluded from renaming: `Uchuva.Revit.App`, external commands, ribbon classes referenced by Revit, and any type referenced from `Uchuva.addin`.
- Python MCP connector: compile `server.py` to `uchuva-mcp.exe` with Nuitka for the commercial installer. PyInstaller remains an acceptable fallback when build speed matters more than protection.
- Digital signing is not part of the current release path. Expect Windows SmartScreen warnings until a future signed release is added.
- Sources and debug files: do not include `.py`, `.pdb`, or obfuscator map files in the public installer.

Required Secret Inputs

Do not commit Gumroad API secrets or obfuscator licenses.

Supported build inputs:

- `obfuscar.console`: default .NET obfuscator command, installed by `dotnet tool install --global Obfuscar.GlobalTool`.
- `-DotNetObfuscatorCommand`: optional custom command template with `{input}`, `{output}`, and `{dir}` tokens if you replace Obfuscar later.

Commercial Build

```
powershell -File scripts\Build-CommercialRelease.ps1 -PythonCompiler Nuitka
```

Free Obfuscation Options

- Obfuscar: free, MIT-licensed, good first choice for basic name obfuscation in a commercial release. It is simpler and less aggressive, so it is less likely to break Revit loading. The commercial build script generates Obfuscar XML files under `build\obfuscar\<year>\obfuscar.xml`.
- ConfuserEx maintained forks: free, commonly MIT-licensed, stronger protection options such as control-flow and anti-tamper. Use carefully and test Revit startup, ribbon loading, bridge startup, and all write/navigation tools after obfuscation.
- Paid tools can be revisited later if the product needs stronger protection and support.

For a dry run on a machine without obfuscation tools:

```
powershell -File scripts\Build-CommercialRelease.ps1 -SkipObfuscation -SkipInstallerCompile
```

Manual Validation

Before uploading to Gumroad:

- Install on a clean Windows user profile.
- Confirm MCP registration points to `uchuva-mcp.exe` for commercial builds.
- Open Revit and verify activation, bridge startup, `uchuva_revit_health`, model reads, selection/zoom tools, dry-run writes, and schedule export.
- Confirm `%LOCALAPPDATA%\Uchuva\bridge-token.txt` exists while the bridge is running and protected routes reject calls without `X-Uchuva-Token`.
- Confirm `%APPDATA%\Uchuva\license.cache` is created after successful license verification and offline grace works without internet.
- Verify SHA256 checksum and document that this release is unsigned.

GUMROAD_ASSETS.md

Gumroad Page Assets

Use this checklist to update the existing Gumroad page.

Positioning

Short description:

Uchuva connects Autodesk Revit to Claude/Cursor through a local MCP bridge, so AI assistants can inspect BIM data, summarize model health, select and zoom to elements, export schedules, and run controlled dry-run automation.

Key Benefits

- Local Revit bridge: BIM data stays on the user's machine unless the user sends it through their AI client.
- AI-ready model context: project summaries, warnings, links, worksets, views, sheets, rooms, families, and parameters.
- Visual navigation: select, zoom, open views, inspect current selection, and navigate warnings or rooms.
- Controlled writes: dry-run first, explicit confirmation required for parameter, view, and sheet changes.
- Commercial installer: Revit 2024-2026 payloads, Gumroad activation, unsigned release notice, and public checksums.

Requirements

- Windows 10/11.
- Autodesk Revit 2024, 2025, or 2026.
- Cursor or Claude Desktop with MCP support.
- Internet connection for first activation and periodic license checks.

Demo Video Outline

Target length: 2-4 minutes.

- Install Uchuva from the Gumroad download.
- Activate with Gumroad license key.
- Open Revit and confirm the Uchuva ribbon/bridge.
- Connect Cursor or Claude to the `uchuva` MCP server.
- Ask for `uchuva_project_summary`.
- Search/select/zoom to elements.
- Export a schedule.
- Run a dry-run parameter update and show that confirmation is required to commit.

Screenshot Checklist

- Installer welcome screen.
- Revit ribbon tab with Uchuva tools.
- License activation dialog.
- Cursor/Claude MCP tools visible.
- AI model summary response.
- Revit selection/zoom result.
- Schedule export result.

Public Changelog Format

```
## Uchuva <version>

- Added:
- Improved:
- Fixed:
- Known limitations:
- SHA256:
```

Gumroad Fields To Verify

- Product title and short subtitle.
- Supported Revit versions.
- Clear refund/support policy.
- "Not affiliated with Autodesk" notice.
- Download file name matches the release version.
- SHA256 checksum is visible near the download notes.

SUPPORT.md

Support, Refunds, and Diagnostics

Support Policy

Initial support target: reply within 2-3 business days through the support email or form listed on the Gumroad page.

Support includes:

- Installation and activation help.
- MCP registration help for Cursor and Claude Desktop.
- Bridge connectivity troubleshooting.
- Reproducible bugs in supported Revit versions.

Support does not include:

- Custom BIM automation consulting unless separately agreed.
- Fixing corrupted models or project-specific Revit standards.
- Support for modified binaries or bypassed licensing.

Refund Policy

Recommended Gumroad policy:

- Refund within 7 days if Uchuva cannot be installed or activated after support troubleshooting.
- No refund for license sharing, unsupported environments, custom workflow expectations not advertised on the product page, or requests after the stated refund window.

Bug Report Template

Ask users to include:

- Uchuva version.
- Revit version.
- Windows version.
- Cursor/Claude version if MCP is involved.
- Steps to reproduce.
- Screenshot or exact error message.
- Logs from `%LOCALAPPDATA%\Uchuva\logs`.

Users should not send Revit models or confidential project data unless a separate agreement is in place.

Diagnostic Collection

Run:

```
powershell -File scripts\Collect-Diagnostics.ps1
```

The diagnostic package should include product version, environment summary, MCP config presence, and Uchuva logs. It should not include model files or license keys.

FAQ

Cursor or Claude does not show Uchuva

Run the Start Menu shortcut "Register Uchuva MCP (Cursor and Claude)", then fully restart the AI client.

Revit bridge does not respond

Open Revit, confirm the Uchuva ribbon appears, activate the license, and start the bridge. Check that `http://localhost:8775/health` responds.

License does not activate

Confirm the Gumroad key is copied completely, internet access is available, and the product id/permalink configuration matches the Gumroad product.

SmartScreen warning appears

The current installer is unsigned, so Windows SmartScreen may show an "unrecognized app" warning. Users should download only from the official Gumroad page and verify the published SHA256 checksum.

PRIVACY.md

Privacy Policy

Draft template for review before publication.

Summary

Uchuva is designed to run locally on the user's Windows machine. By default, Uchuva does not upload Revit models, schedules, element data, logs, or project files to Uchuva servers.

Data Processed Locally

Uchuva may read and process:

- Revit model metadata requested by the user.
- Element ids, categories, parameter names and values, warnings, schedules, sheets, views, rooms, and links.
- Local license key and license verification cache.
- Local diagnostic logs under `%LOCALAPPDATA%\Uchuva\logs`.

Third-Party Services

Uchuva uses Gumroad for license verification. License activation sends the license key and Gumroad product identifier to Gumroad.

When the user connects Uchuva to Claude, Cursor, or another AI/MCP client, model-derived text can be sent to that client according to the user's prompts and the third party's privacy policy. Uchuva does not control how those external services process data.

Logs

Logs are stored locally and are intended for troubleshooting. Users should review logs before sharing them with support and should remove project-sensitive information if needed.

No Default Telemetry

Uchuva does not include default analytics or telemetry. If telemetry is added later, it must be opt-in and must not transmit BIM model data by default.

Data Retention

Local license and diagnostic files remain on the user's machine until removed by the user or uninstaller cleanup. Gumroad retains purchase and license verification records according to Gumroad policies.

Contact

For privacy questions, contact the support email listed on the Gumroad product page.

EULA.md

Uchuva End User License Agreement

Draft template for review before publication.

License Grant

Uchuva grants the customer a non-exclusive, non-transferable license to install and use Uchuva for internal design, documentation, BIM coordination, and productivity workflows with supported Autodesk Revit versions.

License Restrictions

The customer may not:

- Reverse engineer, decompile, disassemble, modify, or bypass licensing or technical protection measures.
- Redistribute, resell, sublicense, publish, rent, or share the installer, binaries, license keys, or private build artifacts.
- Use Uchuva to violate Autodesk, Gumroad, Anthropic, Cursor, OpenAI, or other third-party terms.
- Remove copyright, license, security, or attribution notices.

Ownership

Uchuva and its source code, binaries, documentation, trademarks, and commercial materials remain the property of Uchuva or its licensors. The customer receives usage rights only.

User Data

Uchuva runs locally and does not send BIM model data to Uchuva servers by default. Data may be sent to third-party AI tools only when the user explicitly connects Uchuva through those tools and requests actions that transmit model-derived text.

Warranty Disclaimer

Uchuva is provided "as is" without warranties of merchantability, fitness for a particular purpose, non-infringement, or uninterrupted operation. BIM decisions remain the responsibility of qualified project professionals.

Limitation of Liability

To the maximum extent allowed by law, Uchuva is not liable for indirect, incidental, special, consequential, or punitive damages, loss of profits, project delays, lost data, or model corruption. Total liability is limited to the amount paid for the license in the preceding 12 months.

Termination

The license may be terminated if the customer breaches this agreement, shares license keys, bypasses protection, or redistributes the software. Upon termination, the customer must stop using Uchuva and remove installed copies.

Autodesk Notice

Uchuva is an independent product and is not endorsed by, sponsored by, or affiliated with Autodesk. Autodesk and Revit are trademarks of Autodesk, Inc.

RELEASE_CHECKLIST.md

Release Checklist

Use this checklist for every public Gumroad release.

Versioning

- Update `pyproject.toml`.
- Update `uchuva/__init__.py`.
- Update Revit `.csproj` versions.
- Update `installer/Uchuva.iss`.
- Update `CHANGELOG.md` and public Gumroad changelog notes.

Build

- Run Python tests.
- Build Revit Release artifacts.
- Compile the Python MCP to `uchuva-mcp.exe`.
- Obfuscate .NET DLLs and remove public `.pdb` files.
- Stage installer payload with Revit 2024, 2025, and 2026 components.
- Generate SHA256 checksum.

QA

- Install on a clean machine/user profile.
- Activate with a real Gumroad license.
- Verify offline grace after a successful activation.
- Verify protected bridge routes reject missing `X-Uchuva-Token`.
- Verify Revit 2024, 2025, and 2026 where available.
- Verify MCP registration in Cursor and Claude Desktop.
- Run the manual Revit checklist in `docs/REVIT_MANUAL_CHECKLIST.md`.
- Verify no `.py`, `.pdb`, obfuscator maps, certificates, or private keys are in the commercial installer.

Publish

- Upload installer to the existing Gumroad page.
- Add SHA256 checksum.
- Add release notes and known limitations.

- Confirm download works from a customer account.
- Archive installer, checksum, and release notes internally.

Rollback

- Keep the previous installer available internally.
- If a release is broken, pause Gumroad delivery, restore the prior installer, and publish a short support note.

REVIT_MANUAL_CHECKLIST.md

Revit Manual Validation Checklist

Use a disposable model or a copy of a production model for write tests.

UI Control

- Select one known element id with `uchuva_select_elements`.
- Select multiple known element ids with `uchuva_zoom_to_elements` and verify Revit focuses them.
- Clear the selection with `uchuva_clear_selection`.
- Select elements manually in Revit and verify `uchuva_get_current_selection` returns their ids.
- Select elements with `uchuva_select_by_search` using category/level/name filters.
- Zoom to elements with `uchuva_zoom_to_search` using a parameter filter.
- Select a limited category sample with `uchuva_select_by_category`.
- Open a view with `uchuva_open_view`, then run `uchuva_zoom_to_search`.
- Zoom to a room with `uchuva_zoom_to_room`.
- Select placed views from a sheet with `uchuva_select_sheet_views`.
- Run `uchuva_selection_summary` and `uchuva_selection_parameters` on a manual selection.

Parameter Writes

- Run `uchuva_update_instance_parameter` with `dry_run=true`.
- Re-run the same call with `dry_run=false` and `confirm=true` on a test element.
- Run `uchuva_batch_update_parameters` with two valid changes and one invalid change in dry-run mode.
- Commit a small valid batch and verify per-element results.

Views And Sheets

- Run `uchuva_duplicate_view` in dry-run mode, then commit with a unique name.
- Run `uchuva_rename_view` against a duplicate target name and verify it fails without changes.
- Verify `uchuva_list_titleblocks` returns usable titleblock type ids.
- Run `uchuva_create_sheet_from_titleblock` in dry-run mode, then commit in a test model.
- Run `uchuva_rename_sheet` with a duplicate sheet number and verify it fails without changes.

Logs

- Confirm request activity in `%LOCALAPPDATA%\Uchuva\logs\bridge.log`.
- Confirm write dry-runs and commits in `%LOCALAPPDATA%\Uchuva\logs\write-audit.log`.

CHANGELOG.md

Changelog

1.2.0

- Added AI-oriented project and model health summaries.
- Added family usage and naming audit tools.
- Added schedule JSON and CSV export paths alongside existing Excel export.
- Added controlled instance parameter updates with dry-run and explicit confirmation.
- Added Revit UI control tools for selecting, zooming, clearing, and reading the current selection.
- Added search/category/context navigation tools for selecting or zooming to search results, warnings, sheet views, rooms, and views.
- Added current-selection summary and parameter inspection tools for Claude-driven review workflows.
- Added controlled batch parameter updates capped at 50 changes.
- Added controlled view duplication/rename and sheet creation/rename tools.
- Added write audit logging under `%LOCALAPPDATA%\Uchuva\logs\write-audit.log`.
- Added pagination/caps for large list routes and a short-lived Python metadata cache.
- Added bridge request logging under `%LOCALAPPDATA%\Uchuva\logs\bridge.log`.
- Added Python unit tests for the Revit bridge client.