ChatGPT

Revit Batch Export Add-in – README (Startup Guide)

Intent of the Add-in (Overview)

Revit Batch Export is an Autodesk Revit add-in designed to export user-defined **Selection Sets** (saved filters of elements) from a Revit model into separate Revit project files. The tool automates the process of splitting a large model into smaller, focused models by extracting each selection set as its own `.RVT` file [1]. In doing so, it performs optional cleanup and transformations on each export to prepare the files for independent use. This is useful for breaking a complex project into discipline-specific models, sharing isolated parts of a design with consultants, or simply extracting subsets of a model for external applications [2].

Key Features and Behavior:
- **Batch Exports by Selection Set:** Iterates through each saved selection set (Revit Selection Filter) and exports its elements to a new `.RVT` file [3]. Each output file is essentially a mini-project containing only the elements from that selection.
- **Optional Origin Recenter:** Can **recenter** each selection's geometry to the project origin (0,0) for convenience, while keeping the original elevation (Z-coordinate) intact [4]. This ensures the exported pieces align consistently when referenced into other projects.
- **Automated Cleanup:** After isolating the selection set's elements, the add-in **deletes all other objects** not in the selection and **purges** unused families, views, etc., from the exported file [5]. The result is a clean, lightweight model containing only the desired content.
- **Worksharing Safe:** If the source model is workshared (central model), the tool will **detach the export from central**, so the new files are independent and the original central model remains untouched [6]. (This can be done automatically for safety, or provided as an option to the user.)
- **Logging:** For each exported file, the add-in logs the operation details – which selection was exported, how many elements were included, processing time, and any warnings or errors [7]. This helps in auditing the batch process. By default a log file (e.g. CSV or text) can be generated in a specified folder.
- **Configurable Settings:** Key behaviors (like whether to recenter, paths for output, etc.) are configurable. These settings can be read from a JSON configuration file and will also be exposed in the UI dialog for easy user control [8]. In other words, users can either adjust a `settings.json` file or simply use the on-screen options before running the export.

System Requirements

To develop or run this add-in, ensure you have the following:
- **Autodesk Revit 2026** – The add-in targets the Revit 2026 API (and has been tested on Revit 2026.2). Compatibility with other versions may require minor adjustments (e.g., different API references or manifest changes).
- **.NET 8.0 Runtime & SDK** – Revit 2025 and later are built on .NET 8 (Core); therefore, this add-in must be compiled for **.NET 8.0** to load in Revit 2025/2026 [9]. (Revit 2024 and earlier used .NET Framework 4.8, but

going forward .NET 8 is required for add-ins.) Make sure the .NET 8 SDK is installed on your development machine.
- **Visual Studio 2022 or Newer** – A modern IDE with support for .NET 6/7/8. Visual Studio 2022 (with the **.NET Desktop Development** workload) is recommended for building the solution [10]. Ensure you have references to the Revit 2026 **API assemblies** (e.g., `RevitAPI.dll` and `RevitAPIUI.dll`) available, typically obtained from the Revit installation or the Revit SDK.
- **Revit API Assemblies** – You will need the Revit 2026 API DLLs referenced in your project. If you target a different Revit version, reference the corresponding `RevitAPI.dll` / `RevitAPIUI.dll` and update the add-in manifest accordingly (including the <RevitAddIns> XML with the proper Revit version and path).
- **Windows 10/11 64-bit** – Revit and its add-ins run on Windows. Ensure your build is set to 64-bit (Any CPU or x64) to match Revit's 64-bit process.

(Note: A sample `.addin` manifest file GSADUs.BatchExport.addin is provided/outlined in the project. You will need to place this in Revit's add-ins directory or adjust its `<Assembly>` path to point to the compiled DLL. This tells Revit to load the add-in.)

User Interface and Inputs (Stupid-Simple Workflow)

Once installed, the add-in appears as a button on the Revit ribbon (e.g. under a custom "**Batch Tools**" panel) labeled **"Revit Batch Export"** [11]. The typical user workflow is: **click the ribbon button to launch a configuration dialog, fill in the options, and run the export.** The goal is to keep this UI **simple and straightforward**, collecting all necessary inputs before the batch process begins.

When the **Batch Export** button is clicked, a modal dialog window (e.g. WPF form) will pop up, presenting the user with a few options to configure the export. The dialog will include:

- **Selection Set Picker:** A list of all available Selection Sets (Selection Filters) in the open Revit document. The user can choose one, several, or all of these sets to export. This could be a checklist or multi-select list box of the selection set names [12]. For convenience, a "Select All" option may be provided to quickly choose every set. *(Each chosen set will result in a separate exported file.)*
- **Export Folder Selection:** A text field showing the target directory for the exported RVT files, with a **"Browse..."** button to pick a folder [13]. The user must specify where the new files should be saved. If this is left blank, the tool may prompt the user or use a default (to avoid dumping files in an unknown location). Typically, you would require the user to pick a valid folder before enabling the export.
- **Logging Path (Optional):** By default, the add-in can save a log file (e.g., CSV or TXT) of the batch results. Rather than complicating the UI with a separate path, a simple approach is to use the same folder as above for the log. A checkbox like "**Use same folder for log**" can be provided [14]. (If unchecked, a separate log file location picker could be enabled, but in a minimal setup this isn't necessary.) Often, using the export folder for logs is sufficient for simplicity.
- **Option Checkboxes:** A few simple checkboxes allow the user to toggle certain behaviors:
- **Save Before Export:** If checked, the add-in will save the active document (and if it's a workshared model, optionally sync to central) right before starting the batch export [15]. This ensures the latest changes are saved and avoids prompting the user during the process.
- **Recenter Geometry:** If checked, the geometry of each exported selection will be translated so that its bounding box center is at the project origin (0,0) in the new file [4]. If unchecked, the original coordinates are retained. (This corresponds to the "optionally recenter" feature.)

- **Detach from Central:** For workshared models, the tool by default detaches exports to avoid linking to the original central. A checkbox can explicitly indicate this behavior [16]. In a simple implementation, this might always be true for safety, but we include the option in case the user wants to maintain central links (advanced use case).

- **Purge Unused:** This can be always on (since it's usually desired), or a checkbox if the user wants to skip purging. By default, the add-in will purge unused elements in the exports to reduce file size [5].
<!-- - **Limit Export Count:** *(Advanced/testing)* Optionally, a numeric field or slider could limit the number of sets processed (for example, process only the first N sets). This corresponds to a `limitCount` setting [17]. In a basic workflow, the user can simply select fewer sets, so this is not critical for initial version. -->
<!-- - **Dry Run Mode:** *(Advanced/testing)* A checkbox "Dry Run (no file save)" could allow the user to simulate the batch process without actually writing files [18]. This is mainly for debugging or testing – it would run through the motions and log actions, but not save the new RVTs. For the initial implementation, this can be omitted to keep things simple. -->

- **Action Buttons:** At the bottom of the dialog, the user can click **"Export"** to start the batch process with the chosen settings, or **"Cancel"** to abort. There is no complex multi-step wizard – just set options and go. (Optionally, an **"Export and Close"** vs. **"Apply"** could be offered if we wanted the dialog to remain open for multiple runs, but that's unnecessary in a basic tool.)

When the user hits **Export**, the dialog closes (or perhaps a progress bar appears) and the add-in proceeds to carry out the batch export logic with the provided inputs. During processing, Revit's UI is blocked (since we use a modal dialog and run synchronously), but this is acceptable for a one-off batch operation. After all selected sets have been processed, a summary message is shown – for example, a Revit TaskDialog reporting success: *"Export complete: 5 files generated, 0 errors. Files saved to: C:\Projects\YourProject\Exports\..."* [19]. If any errors occurred for certain sets, the message or log can note that those were skipped. The original Revit model remains open and **unmodified**, aside from being saved at the start if that option was enabled [20]. Finally, the log file (if created) will contain details for each set exported. The user can then go to the output folder to find all the new RVT files.

(In future, a progress indicator could be added for better UX on long runs – e.g., a simple progress bar or status text updating after each file is exported. However, to keep the first version stupid-simple, we may initially just show a final summary dialog when done.)

Development Approach and Guardrails (Keeping It Simple)

This project is being developed with a **"keep it simple"** philosophy. The aim is to implement the core functionality clearly and avoid any unnecessary complexity or over-engineering. We want the code to be straightforward, maintainable, and aligned exactly with the intended features – no more, no less. This approach not only makes the tool easier to build and debug, but also helps guide AI coding assistants (like GitHub Copilot) to stay on track with the actual requirements, without deviating into unplanned features.

Key guidelines for development:
- **Focus on Core Features:** We will implement the features outlined above and nothing beyond that scope. For instance, exporting selection sets and performing the described cleanup and transformations are the focus. Features not mentioned (e.g. unrelated model transformations, complicated UI workflows, etc.)

should **not** be introduced spontaneously.
- **Minimalistic UI & Logic:** The user interface will contain only the inputs described under **User Interface and Inputs**, presented in a single dialog. The logic will be written in a clear sequential manner (collect inputs → loop through selection sets → export each with needed operations → finish). We avoid overly abstract architectures or unnecessary layers. The existing project structure (separating the External Application, External Command, and Core logic library) will be used to organize code, but within that we keep methods focused and readable.
- **No Extraneous Additions:** Do not add features or settings unless they are required to fulfill the project intent. For example, we are **not** implementing things like modeless dockable panels (not needed for a batch one-click tool), real-time progress UI (nice-to-have but can be complex with Revit API), or any cloud integrations, etc. Every piece of code should have a clear purpose tied to the batch export functionality.
- **Guardrails for AI Assistance:** The README and code comments will clearly specify the intended behavior to prevent AI assistants from "going off the rails." By sticking to the described workflow (dialog inputs and the subsequent export steps), we ensure that suggestions from tools like Copilot remain relevant. If it's not in this document, it likely doesn't belong in the code. In short, **follow the plan** and implement the TODOs for each step without inventing new requirements.

By adhering to these principles, the development will produce a reliable add-in that meets its goals without bloat or confusion. The result should be a **simple** yet effective batch export tool. This simplicity will make it easier to test, debug, and extend in the future once the foundation is proven to work.

Summary / Next Steps

With the intent, requirements, and design outlined, the next step is to actually implement the remaining functionality of the add-in according to this guide. Developers (or AI assistants) can proceed to:

1. **Set up the Project Environment:** Ensure references to Revit 2026 DLLs and .NET 8 are configured, and the add-in manifest is in place (pointing to the compiled `GSADUs.Revit.App.dll` or equivalent).
2. **Build the UI Dialog:** Create the WPF form for the batch export options, including the selection set list and input controls discussed. Make sure it runs modally from the External Command and returns the user's choices.
3. **Implement Batch Export Logic:** In the `BatchExportCommand.Execute` method (or in core library functions it calls), implement the loop over selected sets. For each: duplicate the document or isolate elements, apply recenter transform if needed, detach if workshared, purge unused, and save as a new file in the chosen directory. Use transactions and Revit API calls as required.
4. **Logging and Final Feedback:** Write entries to the log (if enabled) for each file exported, and at the end, show a TaskDialog summary to the user.
5. **Test with Simple Cases:** Run the add-in on a sample model with a couple of selection sets to verify that each option works as expected (e.g., files are correctly created, geometry recenters if opted, nothing extraneous remains in exports, etc.). Fix any issues found.

By following this plan and the above "guardrails," the add-in should be developed in a controlled, predictable way. We'll have an accurate README to keep us oriented and a working Revit Batch Export tool that does exactly what it promises – no more, no less. [21]

---

1 2 3 4 5 6 7 8 10 11 12 13 14 15 16 17 18 19 20 21 Revit Batch Export Add-in – Codebase Analysis and Recommendations.pdf
file://file-4U5tHm16u8k1Go5LMpmuWG

9 API Changes 2025
https://www.revitapidocs.com/2025/news