Product Variation Configurator — User Guide

Version: 1.0
Date: 2025-08-18

Introduction

This document explains how to use the Product Variation Configurator (SKU Tool Revamped). It covers the main workflows, explains every UI control and button, and provides a troubleshooting section with common issues and fixes.

Table of contents

```markdown
Product Variation Configurator — User Guide
==========================================

Version: 1.1
Date: 2025-08-18

Introduction

This guide documents the Product Variation Configurator (SKU Tool Revamped). It covers workflows, UI controls, menus, keyboard shortcuts, modal/dialog behavior, and troubleshooting steps based on the current codebase.

Table of contents
- Quick start
- Main workflows
- Loading and saving projects
- Recalculating prices and generating SKUs
- Uploading and downloading databases (cloud sync)
- Exporting to Excel
- Reordering SKUs (separate tool)
- UI reference (menus, buttons, dialogs)
- Troubleshooting & diagnostics
- Advanced notes and tips

Quick start

• Run SkuTool Revamped.py (or SkuTool Revamped Backup.py) with Python 3.11.

• Import a product CSV (File → Import CSV from BigCommerce...).

• Edit base SKU, option costs, weights, and associated SKUs in the Configure panel.

• Use Ctrl+S to save a temporary copy; use Ctrl+Shift+S to save + upload to cloud.
```

• Export SKUs via File → Export Current SKU Breakout to Excel... or run Batch Export.

Main workflows

Loading and saving projects
- Import CSV: File → Import CSV from BigCommerce... (method: load_csv). The app parses "Options" columns into individual option rows.
- Quick Save (Ctrl+S): Saves a temp file (timestamped) to data/ and shows an informational dialog.
- Save Database Locally: File → Save Database Locally (save_to_database) writes a timestamped JSON with current project data.

Recalculating and SKU generation
- Parse Table: Options → Manual Controls → Parse Table (parse_options) converts CSV option blobs into the master option table.
- Generate New SKUs: Options → Manual Controls → Generate New SKUs (generate_new_skus) builds/cleans New SKU values and updates the left preview.
- Recalculate Pricing: Many operations (importing cost DB, editing costs) call recalculate_all_pricing() to refresh derived prices and weights.

Cloud sync: upload / download
- Upload (push_database_to_cloud): Trigger via Ctrl+Shift+S or File → Save and Upload to Cloud. This opens a modal upload dialog and performs uploads on a background thread while UI updates are scheduled with after(0, ...).
- Download (pull_database_from_cloud): On startup the app attempts to sync the latest sku_database and cost_db into data/.
- Safe writes: _safe_write_json() performs atomic writes with temp files and moves to reduce corruption risk.

Exports
- Export Current SKU Breakout to Excel... (export_excel) exports the visible SKU table. The app ensures base price & weight are set before exporting.
- Batch Export SKUs to Excel... (batch_export_excel) takes a list of base SKUs and writes either separate files or combined exports into Excel Exports/ with a timestamped folder.

Excel SKU Reorder Tool (separate utility)
- excel_sku_reorder_gui.py provides:
- Input Excel file selection
- Choice of column to reorder by (radio buttons populated from file headers; defaults to SKU if present)
- SKU list input or file load
- Background processing and optional auto-open of the resulting Excel file

UI reference (menus, buttons, shortcuts)

This section lists the main menus and frequently used commands. Function names in parentheses indicate the code entry point.

• File menu Import CSV from BigCommerce... (load_csv) Export Current SKU Breakout to Excel... (export_excel) Batch Export SKUs to Excel... (batch_export_excel) Bulk Update Base Prices... (bulk_update_base_prices) Save Database Locally (save_to_database) Save and Upload to Cloud (save + push_database_to_cloud) Revert to Backup Database (revert_to_backup) Restore from Local Backup to Cloud (restore_from_local_backup) Delete Base SKU from Database (delete_base_sku) Export Pricing Database... / Import Pricing Database... (export_cost_db / import_cost_db) Exit (invokes _on_quit)

• Import CSV from BigCommerce... (load_csv)

• Export Current SKU Breakout to Excel... (export_excel)

• Batch Export SKUs to Excel... (batch_export_excel)

• Bulk Update Base Prices... (bulk_update_base_prices)

• Save Database Locally (save_to_database)

• Save and Upload to Cloud (save + push_database_to_cloud)

• Revert to Backup Database (revert_to_backup)

• Restore from Local Backup to Cloud (restore_from_local_backup)

• Delete Base SKU from Database (delete_base_sku)

• Export Pricing Database... / Import Pricing Database... (export_cost_db / import_cost_db)

• Exit (invokes _on_quit)

• Options menu Config... (open_options_window) Manual Controls → Parse Table (parse_options) Manual Controls → Generate New SKUs (generate_new_skus) Manual Controls → Extract Base SKU (extract_base_sku)

• Config... (open_options_window)

• Manual Controls → Parse Table (parse_options)

• Manual Controls → Generate New SKUs (generate_new_skus)

• Manual Controls → Extract Base SKU (extract_base_sku)

• Reports menu Base SKU Summary Report (generate_base_sku_summary_report) Associated SKU Usage Report (generate_associated_sku_report) Base/Associated SKU Price Change Reports

• Base SKU Summary Report (generate_base_sku_summary_report)

• Associated SKU Usage Report (generate_associated_sku_report)

• Base/Associated SKU Price Change Reports

• Utilities menu Launch HotKeys... (HotKeys.py) Launch Excel SKU Reorder Tool... (excel_sku_reorder_gui.py) Duplicate Remover... (duplicate_remover.py)

• Launch HotKeys... (HotKeys.py)

• Launch Excel SKU Reorder Tool... (excel_sku_reorder_gui.py)

• Duplicate Remover... (duplicate_remover.py)

File menu

• Import CSV from BigCommerce... (load_csv)

• Export Current SKU Breakout to Excel... (export_excel)

• Batch Export SKUs to Excel... (batch_export_excel)

• Bulk Update Base Prices... (bulk_update_base_prices)

• Save Database Locally (save_to_database)

• Save and Upload to Cloud (save + push_database_to_cloud)

• Revert to Backup Database (revert_to_backup)

• Restore from Local Backup to Cloud (restore_from_local_backup)

• Delete Base SKU from Database (delete_base_sku)

• Export Pricing Database... / Import Pricing Database... (export_cost_db / import_cost_db)

• Exit (invokes _on_quit)

Options menu

• Config... (open_options_window)

• Manual Controls → Parse Table (parse_options)

• Manual Controls → Generate New SKUs (generate_new_skus)

• Manual Controls → Extract Base SKU (extract_base_sku)

Reports menu

• Base SKU Summary Report (generate_base_sku_summary_report)

• Associated SKU Usage Report (generate_associated_sku_report)

• Base/Associated SKU Price Change Reports

Utilities menu

• Launch HotKeys... (HotKeys.py)

• Launch Excel SKU Reorder Tool... (excel_sku_reorder_gui.py)

• Duplicate Remover... (duplicate_remover.py)

Keyboard shortcuts and bindings

• Ctrl+S: Quick save to temp JSON (shows "Saved" dialog) — handled by _ctrl_s.

• Ctrl+Shift+S / Ctrl+S+Shift: Save temp + upload to cloud — handled by _ctrl_shift_s which calls push_database_to_cloud unless in test mode.

• Ctrl+R: Restore main window visibility (restore_window_visibility).

• Mouse wheel: Scrolls the options configuration panel when mouse is over that area.

Modal dialogs & threading behavior

• Upload dialog (show_upload_status_dialog) Modal (transient() + grab_set()), disables close button during upload. Shows step rows: "Uploading Main Database" and "Uploading Pricing Database" with animated spinner and final status marks. The network upload runs in a background thread; UI updates use after(0, ...) so messageboxes and status label updates are executed on the main thread.

• Modal (transient() + grab_set()), disables close button during upload.

• Shows step rows: "Uploading Main Database" and "Uploading Pricing Database" with animated spinner and final status marks.

• The network upload runs in a background thread; UI updates use after(0, ...) so messageboxes and status label updates are executed on the main thread.

• Startup loading dialog (show_loading_dialog) Displays initial sync steps (web service launch, cloud sync, purge old files). Designed to be modal but not block background initialization.

• Displays initial sync steps (web service launch, cloud sync, purge old files). Designed to be modal but not block background initialization.

Upload dialog (show_upload_status_dialog)

• Modal (transient() + grab_set()), disables close button during upload.

• Shows step rows: "Uploading Main Database" and "Uploading Pricing Database" with animated spinner and final status marks.

• The network upload runs in a background thread; UI updates use after(0, ...) so messageboxes and status label updates are executed on the main thread.

Startup loading dialog (show_loading_dialog)

• Displays initial sync steps (web service launch, cloud sync, purge old files). Designed to be modal but not block background initialization.

Data safety & recovery

• get_latest_database_path() selects the newest sku_database_*.json in data/ by modification time — the app reads/writes using that canonical function to avoid accidental overwrites.

• try_repair_json_file() attempts to truncate corrupted JSON to the last closing brace/bracket and parse it; user is informed via messageboxes.

• restore_from_local_backup() lets you push a local backup file to the cloud (validates the backup before upload).

Troubleshooting & diagnostics

1) App appears "Not Responding" after opening dialogs
- Hover over program icon on

2) Upload dialog shows X or Upload fails
- Check that the helper web service is reachable (the app launches it via ensure_web_service_running() if needed).
- Inspect console logs for requests.post responses and error messages.

3) Database corrupted or empty after pull
- The app prompts to restore from a local backup when corruption is detected. If you have local backups in data/ or Old DB/, use the Restore flow.

4) Exports won't run or are missing columns
- Ensure Base Price and Base Weight fields are populated; export will warn and abort if those are blank.

5) Window disappears or appears maximized but invisible
- Use Ctrl+R to restore the main window; the app's restore_window_visibility() will deiconify, lift, and reposition the window when needed.

Next steps I can take for a 10+ page PDF

• Extract every menu label, dialog string, and hotkey with exact file/line references and add them as an appendix.

• Add annotated screenshots: cover image, main window, upload dialog, export dialog, reorder tool. (I can create placeholders or you can provide screenshots.)

• Create a formatted PDF with TOC, headers, and page numbers from this Markdown.

If you'd like, I'll regenerate the PDF now from this expanded Markdown and then continue by extracting a line-by-line UI index (menu label → file & function). Reply with "Regenerate PDF" to produce the PDF, or "Index UI" to continue scanning and produce the detailed index.
```

Use-case workflows (If you need to A, then B)

This section lists concrete, actionable step-by-step sequences for the common tasks you'll be asked about. Copy these into team replies when someone asks "How do I X in the SKU tool?"

1) If you need to import a BigCommerce CSV and begin editing
- Step 1: File → Import CSV from BigCommerce... and choose the CSV file.
- Step 2: When prompted that the base SKU already exists, choose:
- Yes to overwrite existing base SKU data with the CSV, or
- No to load the existing base SKU from the database instead, or
- Cancel to abort.
- Step 3: Confirm Base Price & Base Weight (top controls). If missing, enter them.
- Step 4: Options → Manual Controls → Parse Table to populate the option master table.
- Step 5: Options → Manual Controls → Generate New SKUs to fill the "New SKU" column in the left preview.
- Step 6: Review the Configure panel, edit any Add'l Cost or Add'l Weight values, then click Parse/Generate again if needed.

2) If you need to change pricing values for options and update all derived SKUs
- Step 1: Edit the "Add'l Cost" or "Add'l Weight" fields in the Configure panel for the relevant option value rows.
- Step 2: When finished editing, click Options → Manual Controls → Parse Table (if you changed CSV source) and then Generate New SKUs, or simply run Recalculate (Recalculate Pricing) if available.
- Step 3: Verify the left preview prices/weights updated. If not, save a temp file and reload.

3) If you need to save progress but not upload to cloud (quick save)
- Step 1: Press Ctrl+S or File → Save Database Locally.
- Step 2: The app writes a timestamped temp file to data/ and shows "Saved". Keep working.
- Step 3: When you are ready to make a permanent save, use Save Database Locally (choose folder) or follow the upload steps below.

4) If you need to save and upload to cloud (recommended before quitting)
- Step 1: Press Ctrl+Shift+S or File → Save and Upload to Cloud.
- Step 2: A modal "Uploading to Cloud" dialog will open and show two steps: Main DB and Pricing DB.
```

- Step 3: Wait for the dialog to finish — it will show ✓ on success or X on failure; do not close the main window while upload is in progress.
- Step 4: After upload completes, the dialog will close and you'll be shown an informational dialog of success/failure.

5) If you need to export the current SKU breakout to Excel
- Step 1: Ensure Base Price and Base Weight are filled (the app will refuse the export otherwise).
- Step 2: File → Export Current SKU Breakout to Excel... and select a .xlsx filename.
- Step 3: When prompted "Open in Excel?" choose Yes to immediately open the exported file, or No to leave it on disk.

6) If you need to batch-export multiple base SKUs to Excel files
- Step 1: File → Batch Export SKUs to Excel... to open the batch export dialog.
- Step 2: Paste or type one base SKU per line into the text box and choose variant handling (Both, Only (SSS), Only (MXT)).
- Step 3: Click Export to Excel and wait for progress to finish; exported files are saved under Excel Exports/BatchExport_/.
- Step 4: When prompted, choose whether to open the output folder.

7) If you need to reorder rows in an Excel sheet to match a given list of SKUs
- Step 1: Utilities → Launch Excel SKU Reorder Tool... (or run excel_sku_reorder_gui.py).

BigCommerce (BC) workflows — detailed

The following are BC-focused step-by-step workflows that map common e-commerce tasks (create variations, update pricing, add/remove options, reporting, merging, exports, renames, and cloud-save best practices) to concrete actions in the SKU Tool.

1) Creating variations in BigCommerce → Import → Configure → Export (and verify order)
- In BigCommerce: Create the product and add variant rows (one row per variant). Include a clear "Options" column that contains Name=...|Value=... blobs or otherwise follows your store's export format.
- Export the product/variants CSV from BC.
- In SKU Tool: File → Import CSV from BigCommerce... and select the CSV.
- If prompted that the Base SKU exists, choose overwrite or load existing depending on whether you want to replace DB contents.
- Check/adjust the Base SKU name and make sure suffix (SSS/MXT) is correct.
- Set Base Price & Base Weight in the top controls.
- Options → Manual Controls → Parse Table to populate the master option table.
- Review the Configure panel; edit Add'l Cost/Add'l Weight and Associated SKUs as needed.
- Options → Manual Controls → Generate New SKUs to populate the "New SKU" column. Confirm the prefix and padding logic is correct (prefix entry behavior).
- IMPORTANT: If you plan to import the generated New SKUs back into BigCommerce, verify the order and formatting:
- Step A: Use File → Export Current SKU Breakout to Excel... (or Batch Export) to create an Excel with the New SKU column.
- Step B: Open the exported Excel and confirm the New SKU column is the one BC expects (header, order). Reorder columns if necessary using the Excel Reorder Tool.
- Step C: Import the file into BigCommerce according to your store's import template.

2) Updating pricing in the software (local changes that flow to exports)
- If you need to change option costs for SKUs before exporting:
- Step 1: Open the product in SKU Tool (File → Import CSV or Load from DB).
- Step 2: In Configure, find the option Name and Value and edit the Add'l Cost field.
- Step 3: After edits, click Options → Manual Controls → Generate New SKUs (or run Recalculate) to propagate new prices into the preview.
- Step 4: Export to Excel and use that file to update pricing in BC if required (follow your marketplace flow). Alternatively, keep changes in the central cost_db.json by importing a cost file (Import Pricing Database) so changes are persistent.

3) Add/remove options in BigCommerce → reconfigure in software (delete old, regenerate SKUs)
- Scenario: You add a new option (e.g., Color) or remove an option value (e.g., "Blue") in BC and want to reflect that in the tool.
- Step 1: Export the updated CSV from BC.
- Step 2: In SKU Tool, File → Import CSV from BigCommerce... and point to the new CSV.
- Step 3: If the new CSV includes added options, Parse Table will populate them as new rows in the master table.
- Step 4: If you removed option values in BC, find those values in the Configure panel and delete the corresponding rows from the master table (select and delete via the UI). Do not leave orphaned option values.
- Step 5: Save locally (Ctrl+S) then Save + Upload (Ctrl+Shift+S) to push the new DB to the cloud if you use the cloud-sync workflow.
- Step 6: Run Generate New SKUs (Options → Manual Controls → Generate New SKUs) to regenerate New SKUs based on current option set.
- Step 7: Export SKUs (Excel) and re-import to BigCommerce if you need BC to reflect the regenerated New SKUs.

4) Reports — how to generate each kind and use-cases
- Base SKU Summary Report
- Use-case: Audit a base SKU, see number of variants, last export time, base price distribution.
- How: Reports → Base SKU Summary Report... → choose filters (date range/base sku) → Generate → optional Export to Excel.
- Associated SKU Usage Report
- Use-case: Find where a particular associated SKU (part number) is used across all base SKUs.
- How: Reports → Associated SKU Usage Report... → enter associated SKU or filter → Generate. Use this before renaming or deleting parts.
- Base SKU Price Change Report / Associated Price Change Report
- Use-case: Compare two database snapshots and list price differences (useful for audits before upload).
- How: Reports → choose Price Change Report → select two database files (older and newer) → Generate → Export to Excel for management review.
- Exporting report results: All reports can be exported to Excel or saved as files for sharing.

5) Merge Associated SKUs (combine part numbers used in options)
- Use-case: Multiple part numbers refer to the same physical item or you want to consolidate usage.
- Step 1: Open Utilities or appropriate dialog (Associated SKU management).
- Step 2: Choose Merge Associated SKUs or open the Edit Associated SKUs window for a base SKU.
- Step 3: Provide the mapping (old part -> new part) or select multiple parts to combine.
- Step 4: Confirm merge; the app will update master table entries and may trigger price recalculation.
- Step 5: Save locally and upload to cloud (Ctrl+Shift+S) after verifying results.

6) Dump to Excel (full dataset or selected SKUs)
- Use-case: Provide full SKU breakouts to marketplaces or analysts.
- Export current view: File → Export Current SKU Breakout to Excel... (exports the visible breakout table).
- Batch dump: File → Batch Export SKUs to Excel... → paste SKU list → Export. Files saved to Excel Exports/BatchExport_/.
- Tip: Use the Batch Export dialog's suffix options to limit to (SSS) or (MXT) variants where needed.

7) Change name of Associated SKUs (rename part numbers globally)
- Use-case: Part number changed at supplier; update everywhere.
- Step 1: Find the associated SKU in the pricing DB explorer or Associated SKU usage report.
- Step 2: Use the Rename Associated SKU feature (File → Rename Associated SKU... or Edit Associated SKUs), enter old and new part numbers.
- Step 3: Confirm rename — the app will search usage and, if found, update all relevant master and in-tree entries.
- Step 4: Recalculate pricing and generate SKUs to ensure no downstream inconsistencies.
- Step 5: Save and upload (Ctrl+Shift+S).

8) Save to cloud often — keep your data safe
- Practice: Press Ctrl+S frequently while editing to create temp files. When a logical milestone is reached (finished edits for a product or before closing the app), press Ctrl+Shift+S to upload to cloud.
- Why: The upload process writes timestamped files and preserves last_export metadata; frequent uploads reduce the risk of data loss or DB corruption.
- Extra: Keep local backups of data/ periodically (copy to a different drive). If the cloud file becomes corrupted, use Restore from Local Backup.

```
    - Step 2: Choose the input Excel file and choose an output filename.
    - Step 3: Click "Analyze" to read file headers and populate the reorder-by radio
    buttons.
    - Step 4: Select which column to reorder by (e.g., `SKU` or `New SKU`).
    - Step 5: Paste or load the SKU list and click Process. When complete, accept to open
    the output file.
```

8) If you need to restore from a local backup to the cloud
- Step 1: File → Restore from Local Backup to Cloud (or run restore_from_local_backup()).
- Step 2: Select the SKU backup JSON file (validated) and optional cost_db file.
- Step 3: Confirm the restore; the app will create timestamped copies locally and upload them to the cloud service.
- Step 4: If the upload fails, consult the error dialog and the console log; try again after repair.

9) If you need to revert to a previous local database file (rollback)
- Step 1: File → Revert to Backup Database and select the desired timestamped sku_database_*.json file from data/ or Old DB/.
- Step 2: Confirm the revert — the app will load that file as the current working database.
- Step 3: After revert, re-run Recalculate/Generate as needed to refresh preview tables.

10) If you need to run reports (usage, price changes)
- Step 1: Reports → choose the desired report (Base SKU Summary, Associated SKU Usage, Price Change reports).
- Step 2: Provide any dialog options (date range, filters) and click Generate. Export the result to Excel/PDF if prompted.

11) If you need to diagnose a failing save/upload
- Step 1: Reproduce the failure and copy the exact error message shown in the dialog.

- Step 2: Open a terminal and run the app from Python to capture console output (the app prints DEBUG messages during save/load/upload).
- Step 3: Attach the console output and the exact sku_database_*.json file to your support message.

12) If the app window disappears or shows as maximized but invisible
- Step 1: Press Ctrl+R to restore the main window to a visible location.
- Step 2: If that doesn't work, use Task Manager to bring app to front or restart the application.

13) If you need to update the pricing database from an external Excel
- Step 1: File → Import Pricing Database... or use the Import Pricing Data utility in the Configure panel.
- Step 2: Use the file import UI to map Part Number and Price columns, then Save to Cost DB.
- Step 3: After saving, the app will call update_option_costs_from_cost_db() and then recalculate_all_pricing() automatically.

14) If you want to safely close the app and ensure everything is uploaded
- Step 1: If you have unsaved changes, the app will prompt on Exit. Choose Save if you want a local copy.
- Step 2: When asked about uploading, choose Yes to run the upload dialog; wait until it finishes before confirming exit.
- Step 3: The app will only shut down the helper web service after uploads complete.

Utilities quick reference (one-liners)

• Quick save: Ctrl+S

• Save + upload: Ctrl+Shift+S

• Restore window: Ctrl+R

• Re-generate SKUs: Options → Manual Controls → Generate New SKUs

• Export visible SKUs: File → Export Current SKU Breakout to Excel...

Appendix: Common troubleshooting messages and actions

• "Load a CSV first." — Open File → Import CSV from BigCommerce... and ensure the CSV has an "Options" column.

• "Missing Base" on export — Fill Base Price & Base Weight before exporting.

• "Database Corruption" — Use Restore from Local Backup or run try_repair_json_file() when prompted.