Keep the originals. Recheck the whole history.

One local workspace for later jobs that reuse pieces from older planned stock groups.

Start with original sources

Open Composed-Stock-History.html. Choose an original planned-sequence input journal, or a previously saved composed-history file. Every job needs its original numeric input and complete compatible raw-length report. Base links must also be present.

The workspace accepts at most four jobs in total: base jobs plus appended composed jobs. A four-job base can be reviewed and saved, but cannot add another job. This first version selects finite known intervals only. Abstract finite or unlimited original supply remains separate.

A deliberate six-step route

1. Reopen. Check original history. Every original layout and legacy link replays; no saved “checked” flag is trusted.

2. Choose. Select 1–20 whole known groups. Each key contains both its job number and group ID. Equal numerical lengths merge into finite stock rows.

3. Prepare. Enter fresh demand, kerf, start trim, end trim and new residual minimum. Check the selection, then download the exact new planner input.

4. Plan. Run that exact input deliberately outside the workspace. A platform may charge for planning. The offline workspace never starts a run.

5. Append. Import the complete matching report. Geometry, every part, stock quantity and original-root partition must verify before the job is appended.

6. Save. Save original history. Reopening recomputes every check from those original sources. Nothing is stored automatically.

The checked record and all-root HTML are useful explanations, not accepted history inputs. A numerical assignment is a proposal: it does not identify a physical bar or write stock.

An older parked group can come back.

Worked example: examples/parked-groups-history.json plus its selection and complete report.

Two groups can share one group ID

After the original three jobs, the known pieces are 14 mm from job 2, plus 14 mm and 34 mm from job 3. The two 14 mm groups both use residual-14000. Their full keys differ: (2, residual-14000) and (3, residual-14000). A group ID alone would be ambiguous.

Selecting all three groups prepares 14 mm × 2 and 34 mm × 1: 62 mm in finite known pieces. Origins retain canonical source order regardless of checkbox order. The older job-2 origin is the first 14 mm origin, so it is the proposed origin when the fresh report uses one 14 mm bar.

New demand: four 10 mm parts; kerf 1 mm; fresh trims 1 mm at each end; new residual minimum 2 mm. The included complete report uses one 14 mm piece and the 34 mm piece. The other 14 mm piece remains unused exactly once.

Fresh stock use: **48 = 40 parts + 3 kerf + 4 trims + 1 residual** (mm). The selected 62 mm includes 14 mm not used by this fresh report. A prepared input alone does not establish that allocation.

The used 14 mm piece, in fresh local coordinates



0–1 start trim · 1–11 part · 11–12 kerf · 12–13 residual · 13–14 end trim. Teal shows finished part; amber shows losses; pale green shows remaining length. Tables preserve exact coordinates.

What survives the step

The selected old group leaves the old frontier. Its unused suffix, if any, returns under a fresh stock-row group once; fresh residuals get their new creation job and minimum. Unselected groups retain their previous keys and origins. No piece can remain under both an old and a new frontier entry.

The example ends with a 14 mm unused piece and a 1 mm residual. Original-root accounting is **300 = 238 prior retired + 47 new retired + 14 unused + 1 residual**. The complete root tables show the original interval assigned to every use.

All reports in this pack were generated locally as synthetic numerical cases. They are not platform runs, execution observations, inventory reservations or earnings.

A second composition retains the full path.

A separate precision example appends two compositions after a two-job base.

From two original jobs to a third

examples/base-history.json contains 300 mm of original roots. After its two jobs, 218 mm is retired into parts/losses and 82 mm remains known: one 14 mm piece and two 34 mm pieces. third-selection.json prepares four 10 mm parts with fresh 1 mm kerf/trims and a 2 mm residual minimum.

The third report uses 48 mm and retires 47 mm, leaving a 1 mm residual. Another 34 mm piece stays unused. The 1 mm residual is below the 2 mm minimum supplied in its creation job; this classification is retained when the history is saved.

The fourth job needs an explicit choice

fourth-selection.json explicitly permits the below-creation-minimum piece. It supplies three 0.25 mm parts, zero kerf, fresh 0.001 mm start/end trims, and a new minimum of 0. The arithmetic is **1 = 0.750 parts + 0.002 trims + 0.248 residual** (mm). These tiny values test numerical precision; they are not machining recommendations.



Exact local intervals: start trim [0, 0.001); three adjacent parts ending at 0.751; residual [0.751, 0.999); end trim [0.999, 1). No new kerf is added when the supplied kerf is zero.

Translate once into the original root

The selected 1 mm interval is [95, 96) in row-1-proposed-1. Adding that offset maps the final residual to **[95.751, 95.999)**. Its proposed path is 1:1 → 2:1 → 3:1 → 4:1: job and bar number at each use. Earlier finished parts and losses remain retired.

Whole-history conservation: **300 = 218 + 47 + 0.752 + 34 + 0.248**. The final known length is 34.248 mm in two pieces. The 34 mm piece was created in job 1 and retained unused through later group containers; its creation job was never relabelled.

Reopen examples/four-job-history.json and select the first original root. The final partition table and all four job diagrams preserve the coordinates and complete path.

Save useful files. Reject uncertain sources.

The standard-library Python route performs the same complete original-source replay.

Portable commands

python history.py replay examples/four-job-history.json

python history.py prepare examples/base-history.json examples/third-selection.json

python history.py append examples/base-history.json examples/third-selection.json examples/third-report.json

Commands print JSON to standard output; redirect it to your own chosen file if needed. Invalid data yields a generic rejection and exit code 2. The tool does not echo rejected source values or failed paths. There are no packages to install, network calls or API keys.

Which output to keep

Original history. Reopen this to replay every source pair, link, explicit selection and composed report.

New planner input. Unallocated numeric demand and selected finite stock rows. It starts no run.

Explicit selection. Records fresh allowances and below-creation-minimum permission for that new input.

Checked record / all-root HTML. Explanatory numerical results. HTML contains all original-root interval tables without scripts. Neither replaces original history.

CSV tables. Export final known origins, retired intervals, remaining groups, original source bindings or every planned bar interval. Historical residual rows are intermediate states, not extra final stock. Tables overlap; do not add them together. An empty quantity with unlimited_assumption=true is an unlimited supply assumption, not zero. The CSV formatter alone does not replay or verify a record.

What causes rejection or invalidation

Changed source geometry, mismatched report, duplicate JSON key, unknown schema, supplied derived frontier, forged lineage, fifth total job, invalid Unicode, nonfinite number or negative zero rejects. Limits: 12 MiB aggregate JSON; 2 MiB per original source; 64 KiB CSV; depth 100; one million decoded values. New demand is capped at 100 rows / 500 parts.

Changing the source file clears the checked history and printable record. Changing demand, selection, permission or allowances clears the prepared input and pending report. Stop terminates the disposable local worker; its stale response cannot restore a result. A check has a 20-second limit.

Check material, profile and condition outside the tool. Positive orientation is assumed. No economic value, price transfer, inherited optimum proof, reservation, concurrency lock, machine control or observed physical identity is established. Hashes bind decoded numerical values; they are not file-byte authentication or signatures.