

OPERATOR EDITION

myOrganizer: CURATOR End User Manual

Setup, guided operation, pipeline stages, and output review

Collection Utility for Retro Archive Tracking, Organization, Repository

Updated: October 11, 2026 (v2.5.0-beta.1)

Created, architected, and managed by Miguel A. Ortiz

Development and coding assistance: OpenAI ChatGPT

Third-party tools, authority data, and trademarks remain credited under their respective notices.

Copyright © 2026 Miguel A. Ortiz. All rights reserved except as granted by the applicable license.

Contents

Section	Topic
1	What CURATOR does
2	Quick start
3	Before you run
4	Gateway menu reference
5	Running the pipeline
6	Stage reference
7	Understanding results
8	Review items
9	Archives, CHD files, and disc media
10	Safe operation
11	Troubleshooting
12	Glossary
Current	Current collection behavior
Appendix A	Tool and dependency references
Appendix B	Credits, sources, and third-party attributions

Critical Safety Warnings

These warnings apply throughout this guide. Read them before configuring folders or starting a run.

PROTECT YOUR SOURCE FILES - Run CURATOR against copies, not irreplaceable originals. Keep a verified backup, use one primary source folder per run, and inspect the organized output before replacing or moving source material.

CLEAR CAN REMOVE OUTPUT - Clear replaces the existing contents of the configured collection folder. Choose it only when that folder is safe to replace. Choose Append when existing output must remain and its receipt evidence is intact; invalid retained collection evidence stops safely.

PRESERVE INCOMPLETE SETS - Keep this folder and its contents in the collection output. Manually removing retained incomplete-set archives can prevent a later Append or Resume run from completing.

DO NOT ALTER ACTIVE EVIDENCE - While CURATOR is running, do not move, rename, edit, or delete relational authority, active run state, workspace, logs, reports, placement receipts, or manifests. Cleanup options can remove recovery and audit history; use them only when no stage is active and you understand what will be lost.

AUTHORITY AND RUNTIME LIMITS - CURATOR is deliberate and evidence-driven, not a fast filename sorter. Deterministic matching can take hours. MAME support in myOrganizer: CURATOR is a VERY EARLY, experimental implementation. It must NOT be relied upon as a proven means of organizing MAME sets. Continue using clrmame as the definitive MAME ROM manager. During Scan Preparation it attempts supported CDI, NRG, and MDF/MDS decoding; unsuccessful images are logged, preserved unchanged, and continue to ordinary matching and review.

CDI DAT MATCHING REMAINS UNVERIFIED - No tested Dreamcast or Jaguar CD CDI has matched a loaded DAT, either as the original image or through its decoded tracks. Successful decoding or conversion does not confirm DAT identity or playback compatibility. Review unmatched CDI files and their generated outputs.

1. What CURATOR Does

CURATOR is a PowerShell application that prepares a retro game collection from local source files, DAT authority files, and platform reference data. It is designed for deterministic collection work: it uses configured rules, DAT authority data, file identity, hashes, and recorded lineage instead of guessing from file names.

CURATOR can read platform and DAT reference files, build a compact authority registry, discover local files, stage loose files and archive contents, hash prepared files, match them against DAT authority, separate review items, and publish organized output when the final Organizer stage is used.

CURATOR IS NOT A FAST WAY TO CURATE A COLLECTION. CURATOR is not a fast way to curate a collection. It is deliberately slow and meticulous: it checks evidence, preserves review items, and protects output decisions instead of guessing or rushing files into place.

Design principle

If CURATOR cannot prove a result deterministically, it preserves the evidence for review instead of silently forcing a match.

2. Quick Start

Use this path for a first guided run:

1. Open PowerShell in the CURATOR project folder.
2. Start the gateway: `pwsh -File .\Curator.ps1`

On a clean installation, `database\CURATOR.db` does not yet exist. The Gateway shows only P — Prepare DAR. Other menu choices are hidden and their keys are not accepted until the database exists.

Choose P and confirm the Prepare DAR run. Prepare DAR does not request Clear or Append and does not prepare collection output. It runs Stages 1–3: Stage 1 creates and initializes the database and prepares platform references, Stage 2 discovers DAT files, and Stage 3 publishes the validated DAR authority.

The full menu becomes available when the Gateway refreshes and finds the database. Its appearance does not prove that all three preparation stages succeeded. If Prepare DAR fails, correct the reported problem and run P again; choose O — Organize only after Prepare DAR completes successfully. Stage 2 hashes every original DAT/XML and ingests only new files or files whose content hash changed. Application or mapping changes alone do not re-ingest unchanged files. Stages 1 and 3 retain their own source and policy checks. RunIDs identify execution and provenance; a new RunID alone does not invalidate retained authority.

3. After Prepare DAR succeeds and the full menu appears, use F — Manage folders to review Source files, Organized output, and Working area. Then choose 9 — Check setup and confirm the paths and stage scripts are ready before organizing.
4. Return to the main menu.
5. Choose O — Organize after successful preparation and setup checks. On later runs, repeat P — Prepare DAR when authority inputs change. CURATOR uses the committed relational authority; there is no separate DAR-file selection step.

DAT discovery reports completed DATs and the sets and members read for the current platform. Reference preparation reports its reuse check and summary loading. Checking unchanged sources may still require reading their contents.

Stage 10 processes copy-only outputs first, then single-file ZIPs (including one-member sets), multi-file ZIPs, and remaining conversions. Each ZIP group shows one completed-file count bar for outputs requiring creation. ZIP output uses native 7-Zip Normal compression, level 5, with a default ceiling of 24 archive workers and one native thread per archive.

6. When prompted for Stage 10 output behavior, choose Clear only if the configured output folder can be replaced. In paused Gateway runs (options 2, 4, 6 and 8), press Enter before displaying the Stage 10 summary and again before returning to the Gateway run summary. Acknowledge that final Gateway summary separately. Choose Append to preserve existing output; known unchanged targets are skipped using discovery and retained database receipts. Keep output unchanged during the run; ordinary Append does not detect same-size external changes.

7. Type YES when CURATOR asks for confirmation.

3. Before You Run

CURATOR expects PowerShell 7.6.1 or later. The project folder should contain the application folders, local data folders, and generated-output folders listed below.

Folder	Purpose
config	Main configuration and console/document templates.
core	Shared CURATOR runtime modules.
scripts	Gateway and stage scripts.
templates	Templates for manifests, receipts, reports, warnings, and retained file-based contracts.
tools	Command-line tools such as 7-Zip, CHDMan, DolphinTool, and optional BinMerge.
dependencies	Required DLL libraries used by the application.
DATs	DAT authority files supplied locally by the user.
platforms	Platform XML/reference files supplied locally by the user.
samples	Optional samples; Input is selected through Gateway.
workspace	Temporary staged working area.
_temp	Temporary runtime files.
database	Persistent CURATOR.db authority and relational stage state.

The Gateway shows the effective Source, Collection, and Workspace paths. Use Manage folders to configure these operator paths, and keep all three outside the CURATOR application folder.

WARNING: VERY EARLY MAME Implementation

MAME support in myOrganizer: CURATOR is a VERY EARLY, experimental implementation. It must NOT be relied upon as a proven means of organizing MAME sets. Continue using clrmame as the definitive MAME ROM manager.

MAME output uses SPLIT sets. Use clrmame to manage MAME collections:

<https://mamedev.emulab.it/clrmamepro/>

CURATOR accepts content-validated MAME -listxml exports for coin-operated Arcade ROM and CHD authority. It recognizes the MAME XML structure and metadata rather than requiring the file to begin with MAME. The current output contract is split: each machine keeps its own ROM members, while parent, BIOS, and device dependencies remain separate sets.

For MAME content, internal CHD SHA-1 authority takes precedence over Redump-derived optical-track evidence. CURATOR excludes nodump entries and retains incomplete authority sets for review under Incomplete Sets instead of publishing them as complete.

Supported MAME software-list XML contributes ROM and CHD authority, but does not establish reliable MAME set organization. Use clrmame to organize, audit, and rebuild MAME sets; retain backups before repairs or removals.

Preparing MAME Software-List Authority

MAME support in myOrganizer: CURATOR is a VERY EARLY, experimental implementation. It must NOT be relied upon as a proven means of organizing MAME sets. Continue using clrmame as the definitive MAME ROM manager.

MAME software lists are official XML authority files in MAME's  hash folder. For Atari 2600, CURATOR requires these matching-release files in one local folder:

-  a2600.xml - cartridge software authority
 -  a2600_cass.xml - cassette software authority
1. Download the intended official MAME release from <https://www.mamedev.org/release.html> and extract it. Use one release consistently; do not mix software-list files from different MAME releases.
 2. Open the extracted MAME hash folder and locate the official per-list XML files.
 3. Copy  a2600.xml and  a2600_cass.xml into the appropriate authority folder, for example  DATs\Atari\MAME\Atari 2600. CURATOR reads the per-list XML without loading an external DTD.
 4. Run Prepare DAR, or, after database initialization, run Stages 1, 2, and 3 separately from the full menu. Review Stage 2 and Stage 3 evidence to confirm both software lists were accepted and published to Core.Database.
 5. When updating MAME, replace both XML files from the same new release and rebuild Stages 1-3. Retain the MAME release number as provenance for the authority baseline.

Use the official per-list XML files from the MAME hash folder for these inputs. A separate DTD file is not required.

Official references: MAME software-list guidelines, <https://docs.mamedev.org/contributing/softlist.html>; MAME command-line reference,

<https://docs.mamedev.org/commandline/commandline-all.html>; MAME releases, <https://www.mamedev.org/release.html>.

Preparing Atari 8-Bit Preservation Authority

Unsupported HyperList inputs

Do not use the Atari 2600 HyperList XML or its derived DAT as authoritative ROM definitions, including copies from older packages.

The source XML does not declare complete member identities. Values inferred from other DATs are not declarations from HyperList.

CURATOR recognizes Atari 8-bit Preservation exports by their contents:

-  Atari 8-bit Software Preservation Initiative export - physical-release provenance with machine-readable dump hashes from <https://www.a8preservation.com/v2>
 1. Generate or download the current machine-readable export from the authority's official site. Keep the unmodified export and record its retrieval date and source URL.
 2. Place each export anywhere under CURATOR's  DATs folder. Use a clear Atari 8-Bit subfolder; filenames are descriptive only because Stage 2 recognizes the format from its content. CURATOR public packages include the eight individually allowlisted CURATOR-created authority DATs: the Atari 8-bit Preservation DAT, six Atarimania GameBase DATs for Atari 2600, 5200, 7800, Jaguar, Lynx, and ST, and the Atari Lynx Dragnet supplemental DAT; all other DAT payload remains user-supplied and excluded. Sets from GameBase-derived DAT sources are placed beneath `<Output>\<Company>\<Platform>\GameBase\`. Category, revision and alternate-dump subfolders follow that bucket. Existing output is not automatically reorganized into the new bucket.
 3. Run Prepare DAR, or, after database initialization, run Stages 1, 2, and 3 separately from the full menu. Stage 2 must identify the A8P profile, and Stage 3 must publish its sets and members into Core.Database.
 4. When the export changes, replace that authority file, rebuild Stages 1-3, and retain the source date as provenance for the new authority baseline.

Use the available authoritative sources when evaluating Atari 8-bit coverage. Absence from No-Intro, TOSEC, Libretro, or MAME alone does not establish that a file is unknown, invalid, or corrupt.

4. Gateway Menu Reference

Start the user-facing gateway from the project root with:

pwsh -File .\Curator.ps1

The following table describes the full menu. On a clean installation without database\CURATOR.db, only P — Prepare DAR is displayed and accepted. Complete preparation before using Organize.

Choice	Menu item	Use it when
P	Prepare DAR	Run Stages 1-3 to refresh platform and DAT authority.
O	Organize	Run Stages 4-10 against the committed relational authority revision.
1	Full run	Run all configured stages without intermediate stage pauses. Final-stage and Gateway summary acknowledgements remain.
2	Full run with pause	Run all configured stages. Press Enter before and after each stage summary, including the final stage; acknowledge the Gateway run summary separately.
3	Run one stage	Run one selected stage.
4	Run one stage with pause	Run one selected stage. Press Enter before and after its stage summary; acknowledge the Gateway run summary separately.
5	Run from stage	Run from a selected stage through the end.
6	Run from stage with pause	Run from a selected stage through the end. Press Enter before and after each stage summary, including the final stage.
7	Run through stage	Run from Stage 1 through a selected stage.
8	Run through stage with pause	Run from Stage 1 through a selected stage. Press Enter before and after each stage summary, including the final selected stage.
9	Check setup	Check project paths, configuration, dependencies, and stage readiness.
10	Current status	Show the current configured folder and latest run status.
R	Collection coverage report	Open current Reporting-generated company coverage results.
V	Full physical collection audit	Hash every collection artifact and review differences before any manifest update.
H	Clear histories/temp/workspace	Run the confirmed maintenance cleanup workflow.
D	Generate API document	Regenerate the Core API Developer Handbook source document.
S	Clear run states	Clear supported saved run-state records after confirmation.
F	Manage folders	Change Source, Collection, and Workspace operator paths.
X	Advanced options	Open advanced maintenance and diagnostic options.
Q	Exit	Close the gateway.

5. Running the Pipeline

Full Run

A full run executes all configured stages in order. Stages 1-3 maintain persistent authority, while Stages 4-10 use one disposable relational collection run bound to the committed authority revision. CURATOR asks you to type YES before it starts.

Full Run With Pause

This mode uses the same stage order as a full run. Press Enter once to display each stage summary and again to continue, including after the final stage. The Gateway run summary has its own acknowledgement. The same before-and-after stage pauses apply to Run one stage with pause, Run from stage with pause and Run through stage with pause (options 4, 6 and 8). Use paused runs when reviewing a new setup, checking progress behavior, or learning what each stage produces.

Selected Stage Runs

Selected-stage runs do not rebuild unrelated state. Use them only when the required persistent authority and prior relational stage state still match the data you intend to process.

Stable authority baseline: If your platform files and DAT files already cover the collection you plan to curate, Stages 1-3 usually need to run again only when that authority data changes. For normal curation batches against the same authority revision, begin with File Discovery unless a valid disposable run is being resumed.

Stage 10 Output Choice

Choice	Meaning	Practical advice
Clear	Remove existing top-level output folder contents before publishing.	Use only when the configured output folder can be replaced.
Append	Keep existing output folder contents. Known unchanged targets are skipped using discovery and retained database receipts. Same-size external changes are not detected by ordinary Append.	Use when preserving existing output and its validated receipt evidence matters.

6. Stage Reference

No.	Name	What it does	
1	Platform Preparation	Reads platform reference files and builds platform mapping for later stages.	Relational platform authority

No.	Name	What it does	
2	DAT Discovery	Reads DAT files, identifies DAT profiles, associates DATs with platforms, and publishes DAT inventory.	Relational DAT authority
3	DAT Authority Resolution	Builds the deterministic authority registry used for matching.	Relational DAR authority
4	File Discovery	Scans the configured input folder and records discovered source files.	Relational input inventory
5	Scan Preparation	Stages loose files, archive contents, CHD-derived files, and supported optical-media inputs for scanning.	Relational prepared state
6	Deterministic Matching	Hashes prepared files and matches them using size, CRC, SHA1, and RVZ effective-disc evidence when applicable.	Relational matching state
7	DAT Validation	Validates Matching output against DAR authority and scan lineage.	Relational validation state
8	Collection Assembly	Builds the assembled collection plan from validated files.	Relational assembly state
9	Collection Completeness Validation	Checks planned output readiness, review items, duplicates, and artifacts.	Relational frozen plan and receipt
10	Final Collection Output	Publishes organized ZIP, CHD, direct RVZ, review, and preserved output according to the Stage 9 plan and output decision.	Relational execution receipts

7. Understanding Results

Each stage commits its required relational state and writes logs to its numbered stage folder. Final manifests and reports remain file-based where their contracts require them. The standard stage log files are:

 **Run summary** RUN.log

 **Operational detail** DETAIL.log

 **Diagnostic detail** DEBUG.log

 **Completion summary** — The completion screen provides the most useful user-facing summary.

Summary value	Meaning
Matched files	Files deterministically matched to DAT authority.
Unmatched files	Files that could not be matched to the current DAT authority.
Ambiguous files	Files with more than one possible authority path.

Summary value	Meaning
Hash failures	Files that could not be hashed successfully.
Warning or review files	Valid records that need manual review before final collection placement.
Downstream ready	Later stages can continue from the current output.
GO_WITH_REVIEW_ITEMS	The plan can continue, but review items exist.
GO_FOR_PHYSICAL_OUTPUT	Organizer has enough valid planned output to publish physical files.

8. Review Items

Review items are not discarded. They are tracked so the user can inspect them after the run. Review items are common when DAT coverage is incomplete, files are unusual, dumps are bad, or inputs do not belong to the current DAT set.

A file may require review because no matching DAT authority was found, a hash did not match, the match was ambiguous, or the file is valid evidence but not eligible for final official output.

9. Archives, CHD Files, and Disc Media

When CURATOR traverses an archive, the unmatched container itself is not copied to needs_attention if its child lineage was processed. Each extracted descendant is evaluated independently. This avoids redundant archive copies and .zip.zip output names.

CURATOR treats archives as containers. During Scan Preparation, archive contents are staged into the workspace while preserving source lineage.

Scan Preparation decodes and hashes logical CHD tracks without writing BIN files. Matching reuses bounded header prefixes; applicable normalization and pending output operations extract tracks through the configured Core tools. Copying an unchanged whole CHD requires no track extraction. All input classes use compact hash-addressed workspace roots, avoiding path-length growth without changing source identity. CURATOR uses recorded logical-track evidence for matching and can later rebuild official CHD output artifacts when the plan says that is appropriate. RVZ files are handled as optical-media sources through DolphinTool evidence; matched official RVZ outputs remain direct .rvz files, use the DAT-authoritative basename with the .rvz physical extension, and are not zipped. When recovery supplies an authoritative corrected CUE, Stage 9 reconciles it against the complete originating CHD membership and suppresses only obsolete recovered-CUE review records. Genuinely unresolved source evidence remains in NeedsAttention.

Disc descriptor files such as .cue are handled cautiously. CURATOR records lineage and matching evidence instead of assuming platform identity from the descriptor file name alone. Complete Atari Jaguar CD authority sets publish as ZIP. After Stage 4, normal processing

uses the Workspace, CURATOR-owned temporary files, and the frozen Stage 9 plan. An already-valid source CHD is the intentional exception: when its content satisfies the accepted authority set, Stage 10 copies the CHD directly from the original configured Input folder instead of spending time generating an identical CHD. CURATOR first verifies that the recorded physical CHD path agrees with the configured Input folder; a disagreement stops safely rather than copying from an unexpected location. Direct copy does not alter the original CHD. Keep the Input folder connected and unchanged until Stage 10 completes. Other complete optical authority sets publish as newly created CHD files. The source container does not choose the final optical format.

CURATOR uses bundled managed libraries for this work. You do not need to install, launch, or configure Aaru separately. Keep the complete application dependencies folder intact.

MDF/MDS NOTICE - Some MDF/MDS pairs cannot be opened by the bundled decoder because implementations of the format differ. CURATOR logs the problem, retains both original files, and moves on through its standard container-handling workflow.

A failed MDF/MDS decode is not proof of corruption. Expect the preserved files to remain unmatched and appear in review when no authoritative content match is available.

10. Safe Operation

- Keep a backup of source files before large collection work.
- Use Full run with pause when reviewing a new setup.
- Use Append when existing output must be protected and retained receipt evidence is available. Append uses output discovery and retained database receipts to skip known unchanged targets without repeating content hashes or archive inspection in Stage 10. Keep the output folder unchanged during the run. Ordinary Append does not detect same-size external content changes or changes after discovery; use the supported full physical audit after manual changes. Valid unchanged output is not rewritten. Eligible incomplete-set ZIPs can be replaced when verified new members are added.
- Preserve the 00_Manifest folder, CURATOR's database, and Organizer receipt evidence. CURATOR_Collection_Manifest.json is published from the permanent Official and Unofficial generation in Core.Database, while receipt evidence proves which existing files Append may safely reuse. Reporting reads the permanent database generation rather than treating company-presence manifests as collection authority. Stages 1-3 may refresh authority without discarding collection progress. Starting Stage 4 clean warns before replacing that progress and requires Stages 5-10 to run again. Stage 10 recovery remains available for an interrupted physical run, but a completed Stage 10 plan cannot run again; rebuild Stages 4-9 for a new collection reality. Deleting or changing the database, current collection manifest, or trusted receipts can break continuity and require supported recovery or a deliberate rebuild.

- Keep review files that you want to reconsider with newly acquired source material. The needs_attention folder is disposable: if it or individual files are absent, the next run can create review output again. Old database receipts remain retained but missing review files do not reserve destinations or authorize reuse. Existing files still undergo integrity checks. Review output remains outside permanent collection truth.
- Use Clear only when the configured output folder can be replaced. After manual collection changes, use Gateway option V for a full SHA-256 audit and approve only manifest differences you intend to accept.
- Do not manually edit generated JSON files.
- Review unmatched and ambiguous files instead of deleting them blindly.
- Run CURATOR in an interactive PowerShell window for normal progress display and user-facing validation.

11. Troubleshooting

Problem	Likely cause	What to do
DAT files are missing	The DATs folder does not contain the required DAT files.	Add DAT files to the configured DAT folder and run again.
Platform files are missing	The platforms folder does not contain required platform reference files.	Add platform XML/reference files and run again.
A later stage reports missing prior state	The earlier required relational stage state does not exist or is not ready.	Run the earlier stages first, or use a full run.
Matching reports unmatched files	The current DAT authority does not cover those files, or the files differ from known hashes.	Review the files and DAT coverage. This is not always a failure.
Organizer routes output to review or blocks output	Append stops safely on missing or mismatching receipt evidence; other blockers include readiness issues, missing sources, invalid planned paths, or planned-output collisions.	Read the Completeness and Organizer summaries before rerunning. Use Clear only if replacing the output folder is intended.
Progress display looks wrong	The console is captured or non-interactive.	Use an interactive PowerShell window for normal user-facing runs.

Antivirus may interrupt processing

Antivirus and other security software may interrupt myOrganizer: CURATOR by temporarily blocking access to files it needs. This can cause delays or prevent an operation from completing. If this happens, review your security software's alerts and activity history. An access error alone does not confirm that antivirus software caused it.

DolphinTool warnings

If DolphinTool displays a blocking Warning dialog, read its message before dismissing it with OK. A warning alone does not establish its cause.

12. Glossary

Term	Meaning
Authority	The selected DAT-backed record CURATOR uses as the official identity for a file or set.
Canonical	CURATOR's normalized internal record shape for reliable stage-to-stage evidence.
Lineage	The record of where a file came from, such as a loose file, archive member, or CHD-derived member.
Hash	A file identity value used for deterministic matching. CURATOR uses size, CRC, and SHA1.
Artifact	A planned or produced output file, such as a ZIP or CHD.
Review item	A valid record that CURATOR preserved for inspection instead of placing into official output.

Document Notes

This manual is organized for user tasks first, then reference. Procedures use numbered steps for actions, tables only where lookup or comparison helps, and short explanatory sections for concepts that affect safe operation.

myOrganizer: CURATOR v2.5.0-beta.1 is in BETA. It is intentionally strict: if a required file, tool, folder, or authority record is missing, CURATOR is expected to stop rather than guess.

Current collection behavior

Complete Redump Sega Naomi disc sets can produce standalone CHD output using authoritative filenames. DAT-declared publication files are retained even when an image-related folder name is present. XML game identifiers remain separate from authoritative filenames.

Start collection stages through the Gateway. Clear or Append is selected before processing; Clear replaces prior output once. Resume retains the interrupted run identity and completed output without clearing again. Blocking completeness findings stop final output. Receipts are saved before generated workspace sources are removed.

Recognized catalogue categories and format folders are resolved independently. USA-inclusive sets stay at the platform root regardless of language tags; recognized alternate editions use Alternates. Original member directories remain inside ZIPs. Only a file named exactly _ with zero bytes is excluded as a placeholder.

For a bound single-member set, a missing release classification can use recognized DAT set tags such as Beta or Unlicensed. An already recognized member classification keeps precedence. Matching, payloads and set memberships are preserved.

CDI DAT matching remains unverified. No tested Dreamcast or Jaguar CD CDI has matched a loaded DAT, either as the original image or through decoded BIN tracks. CURATOR can attempt CDI decoding and preserve the original image, but successful conversion does not identify the game or confirm playback compatibility. Review unmatched originals and generated outputs before using them in your collection.

Completeness follows the selected DAT set. A DAT may describe one disk as a complete set, even when the game needs several disks. When every required member of that DAT set is found and matched, the set follows its normal destination rules.

Whole-game estimates are informational and do not determine output placement. They do not establish that a game is playable or that all editions, disks or companion files are present.

When required DAT members are missing, the available matched members remain under Incomplete Sets, grouped by their authoritative DAT set. Similar names do not combine different DAT sets. Append may add verified members to an eligible existing incomplete-set ZIP, but even a completed accumulated ZIP is not automatically moved or converted into the normal collection.

Append can add verified new members to an eligible existing ZIP in Incomplete Sets. CURATOR validates the retained archive and freezes a combined member recipe before replacing it; repeated members are skipped without rewriting the archive. Earlier contents,

source history and set memberships are preserved. Completed accumulated sets remain in Incomplete Sets; automatic completion assessment and final relocation or conversion are not available.

WARNING: Preserve Incomplete Sets. Do not manually delete or move the Incomplete Sets folder or its contents. Later Append or Resume processing that needs a previously recorded incomplete-set archive will stop if that archive is missing. Recreating an empty folder is insufficient. Keep the folder and its contents in place until this limitation is addressed.

Choose Resume from the Gateway to continue the saved stage without clearing successful output. Ordinary output failures are recorded while later outputs continue; Resume retries unfinished work and preserves successful outputs. Catastrophic failures can stop processing. ZIP groups show one completed-file count bar; optical preparation uses one shared display. Collection clearing uses simple messages. Recognized cracked variants use the Cracked folder label for newly planned output; existing folders are not renamed.

Incomplete-set save details are recorded in DETAIL.log. A log file that remains unavailable can stop processing. Review the logs if cleanup reports that working files remain. Stage 10 retains workspace contents until output creation and collection catalogue publication finish, then clears the contents once while preserving the workspace folder. If publication or final cleanup fails, choose Resume to finish the remaining work without creating a duplicate collection generation. Final cleanup announces the total file and folder count before deletion and completion afterward, without a cleanup progress counter.

Execution.Workspace.CleanOnFinish must remain true.

v2.5.0-beta.1 accepts existing v2.5.xx and v2.0.0-beta.6 databases at schema 26 and preserves their creation identity. This compatibility does not establish that every unfinished run can resume. Let the Gateway check recovery eligibility. Keep input and working files unchanged until processing finishes.

Stage 10 streams ZIP/Copy receipts without rechecking source content after planning or rereading fresh output. Existing Append and recovery checks remain.

Authoritative member names determine ZIP entries; configured package naming also uses authoritative set names. Naming is separate from duplicate equality. Existing CHD, CDI and RVZ formats retain their physical extensions; CHDs remain standalone and never belong inside ZIPs.

Original DAT definitions and set memberships remain preserved. Unidentified ordinary files go into folders named after their source extensions, such as A26, BIN and JPG, within needs_attention. Files without an extension remain under Unknown File. Unidentified optical discs keep their existing whole-disc review handling. Filename collisions receive numbered suffixes before the extension. Missing MD5 alone does not make a file unresolved.

Duplicate sets are compared by their complete member contents, including repeated-member counts: Size, CRC and SHA1, with MD5 compared when both declarations supply it. A missing MD5 does not split otherwise matching content and cannot bridge conflicting supplied values. Names, member order, DAT filename, format, source, platform and ranking are excluded from equality. Configured source hierarchy selects one whole-set winner; distinct sets that share only some members remain separate. Preserve every original declaration and membership. Filename comparison is case agnostic.

Unmatched review plans bind their hashes to the selected physical source, including whole CHDs and raw CDI tracks. These physical hashes are separate from normalized matching identity and from final output SHA-256 receipts.

Organizer can publish an empty permanent collection manifest. Standard Reporting and FullAudit accept empty collections and continue to report populated collections. Physical file counts and fulfilled-definition counts can differ because one physical file can retain several memberships.

RVZ creation and recovery of converted media have limited support. MAME support is very early and experimental; use clrmame to manage MAME collections. MAME output uses SPLIT sets.

Appendix A: Tool and Dependency References

Citations for the current CURATOR tool and dependency set. Local configuration remains authoritative for required paths and required/optional status.

PowerShell 7+: Runs the CURATOR gateway and stage scripts. Citation: Microsoft Learn: <https://learn.microsoft.com/en-us/powershell/scripting/overview?view=powershell-7.5>

AlphaFS: Helps CURATOR handle file-system work such as long paths. Citation: AlphaFS GitHub: <https://github.com/alphaleonis/AlphaFS>

HashLib4CSharp: Optional SHA-256 services. Core.Archive owns CRC32/SHA-1 streaming through 7z.dll. Citation: <https://www.nuget.org/packages/HashLib4CSharp>

Newtonsoft.Json: Reads and writes JSON evidence files. Citation: Json.NET: <https://www.newtonsoft.com/json>

Newtonsoft.Json.Schema: Supports JSON schema validation. Citation: Json.NET Schema: <https://www.newtonsoft.com/jsonschema>

Spectre.Console and Spectre.Console.Ansi: Displays CURATOR progress and console UI. Citation: Spectre.Console: <https://spectreconsole.net/> ; GitHub: <https://github.com/spectreconsole/spectre.console>

Microsoft.Bcl.TimeProvider: Supports timing behavior used by progress rendering. Citation: Microsoft Learn: <https://learn.microsoft.com/en-us/dotnet/api/system.timeprovider?view=net-9.0>

7-Zip / 7z.dll: Handles archive extraction and validation support. Citation: 7-Zip: <https://www.7-zip.org/> ; License: <https://www.7-zip.org/license.txt>

Aaru managed libraries: Read supported CDI, NRG, and MDF/MDS images in process. CommonTypes uses the MIT license; the remaining bundled Aaru assemblies use LGPL-2.1-only. Project: <https://www.aaru.app/>

Modified MAME chdman: Handles CHD extraction, verification, and CHD output creation. Citation: MAME docs: <https://docs.mamedev.org/tools/chdman.html>

DolphinTool: Handles RVZ verification/header evidence for official RVZ output support. Citation: Dolphin GitHub: <https://github.com/dolphin-emu/dolphin>

The CUE-authority resolver accepts a corrected CUE descriptor only when every extracted media member matches one unambiguous authoritative set. This check does not replace general-purpose CHD integrity verification.

clrmame / clrmamepro: Recommended external scanner and rebuilder for MAME-native ROM-set management. Use current clrmame for new setups; legacy clrmamepro remains available. Citation: Official project: <https://mamedev.emulab.it/clrmamepro/> ; Current documentation: <https://mamedev.emulab.it/clrmamepro/binaries/readme.html>

BinMerge: Optional BIN/CUE helper retained in the tools folder. Citation: BinMerge GitHub: <https://github.com/putnam/binmerge> ; Releases: <https://github.com/putnam/binmerge/releases>

Appendix B: Credits, Sources, and Third-Party Attributions

CURATOR was created, architected, and is managed by Miguel A. Ortiz. Copyright © 2026 Miguel A. Ortiz. Development, coding, and documentation assistance was provided by OpenAI ChatGPT; that assistance does not transfer project ownership or third-party rights.

Bundled components remain the work of their respective authors and contributors and remain governed by their own licenses. The concise credits below supplement—not replace—the controlling upstream notices and license texts.

Component	Creator / contributor credit	License and authoritative source
Aaru managed libraries	Natalia Portillo and Aaru contributors; CURATOR modifications: Miguel A. Ortiz, with OpenAI ChatGPT assistance	MIT (CommonTypes); LGPL-2.1-only (other bundled Aaru assemblies); https://www.aaru.app/
AlphaFS	Peter Palotas, Jeffrey Jangli, Alexandr Normuradov, and contributors	MIT; https://github.com/alphaleonis/AlphaFS
HashLib4CSharp	Ugochukwu Mmaduekwe and contributors	MIT; https://www.nuget.org/packages/HashLib4CSharp
Microsoft libraries	Microsoft and .NET Foundation contributors	MIT; Microsoft.Bcl.TimeProvider and Microsoft.Data.Sqlite
Newtonsoft libraries	James Newton-King and contributors	Json.NET: MIT; Json.NET Schema: AGPL-3.0-only or commercial license; https://www.newtonsoft.com/
SQLitePCLRaw / SQLite	Eric Sink, SourceGear, LLC, Microsoft Open Technologies, Inc., and contributors; SQLite authors	SQLitePCLRaw: Apache-2.0; SQLite: public domain; https://github.com/ericsink/SQLitePCL.raw ; https://sqlite.org/
Spectre.Console	Patrik Svensson, Phil Scott, Nils Andresen, Cédric Luthi, and contributors	MIT; https://github.com/spectreconsole/spectre.console
7-Zip	Igor Pavlov; retained notices also credit Apple Inc., Facebook, Inc., Yann Collet, and Alexander Roshal (unRAR)	LGPL-2.1-or-later plus unRAR restriction and BSD components; https://www.7-zip.org/license.txt
BinMerge	Chris Putnam and contributors	GPL-2.0-or-later; https://github.com/putnam/binmerge
Modified MAME chdman	MAMEDev and contributors; CURATOR modification copyright 2026 Miguel A. Ortiz	GPL-2.0-or-later; https://docs.mamedev.org/license.html
DolphinTool	Dolphin Emulator project contributors	GPL-2.0-or-later; https://dolphin-emu.org/
Redump cue/GDI metadata	Redump project contributors	Public-domain metadata; https://redump.org/
verifydump	j68k; embedded Python, PyInstaller, OpenSSL and Microsoft contributors	Version 1.0.0; MIT plus incorporated runtime licenses; see LICENSE

Complete packaged records:  **LICENSE** and  **config\dependency-manifest.json**. The dependency manifest records the exact packaged versions, hashes, provenance, and license identifiers.

Authority and data-source credit: No-Intro / DAT-o-MATIC, Redump, TOSEC, Libretro Database, MAME, progetto-SNAPS, LaunchBox, and other operator-selected authorities

remain credited to their respective projects and contributors. Their names and trademarks belong to their respective owners; mention does not imply affiliation, sponsorship, or endorsement.

Modified chdman and source availability

CURATOR bundles tools\chdman.exe as the unofficial modified MAME chdman build 0.289-CURATOR.8-poc. It retains standard chdman commands and adds logical-track hashing. Scan Preparation decodes and hashes logical tracks without writing BIN files and retains bounded header prefixes for Matching. Applicable normalization rules and pending output operations still extract tracks through the configured Core tools. Collection Assembly uses recorded relational source evidence without probing or rehashing collection media. Copying an unchanged whole CHD does not require track extraction. Acceptance is bounded to the documented collection; it does not establish universal compatibility or playback correctness. When recovery supplies an authoritative corrected CUE, Stage 9 reconciles it against the complete originating CHD membership and suppresses only obsolete recovered-CUE review records. Genuinely unresolved source evidence remains in NeedsAttention.

This is not an official MAMEdev build and is not produced, sponsored, approved, supported, or warranted by MAMEdev. MAME is a registered trademark of Gregory Ember.

This v2.5.0-beta.1 release provides matching application, modified chdman, modified Aaru.Images and third-party source ZIPs, a release inventory and SHA-256 sidecars. Provide all source companions beside the application download with equivalent access. Packaged Input, Output and Workspace defaults are exactly C:\; configure appropriate destinations before use. CleanOnFinish remains true and schema remains 26.

Release dependency records identify verifydump 1.0.0, unofficial Aaru.Images 5.4.1-curator-poc1 and unofficial chdman 0.289-CURATOR.8-poc. Json.NET Schema 4.0.1 is supplied under its published AGPL alternative (publisher limit: 1,000 validations per hour). See LICENSE for exact source assets and original terms. CDI DAT matching remains unverified. This beta does not establish universal acceptance of source formats, interruption recovery or ST/GameBase inputs. Review results and preserve originals.