**8. Files on Disk**

📁 *Reference for Workbench Artifacts*

This section documents the files used and generated by the Mango Workbench and cryptographic engine. It defines naming conventions, directories, purposes, and usage constraints. It is intended for both internal development and advanced users creating ML pipelines or automation scripts.

**🧬 File Naming Convention**

Most generated files follow a structured pattern:

php-template

CopyEdit

<Name>,-L<N>-P<P>-D<T>-M<M>-S<S>.<ext>

**🔤 Components:**

| **Tag** | **Meaning** |
| --- | --- |
| Name | File category: Contenders, MungeFailDB, State, etc. |
| L<N> | Munge Level — e.g., L1 to L5 (sequence depth) |
| P<P> | PassCount threshold (metrics a sequence must pass) |
| D<T> | Input/Data Type: DC = Combined, DN = Natural, DR = Random, DS = Sequence, DU = UserData |
| M<M> | Evaluation Mode: C = Cryptographic, E = Exploratory |
| S<S> | Scoring Model: SP = Practical Score, SM = Metric Score |

**📂 Directory Layout & File Roles**

**🔸 Output/ — Writable, Ephemeral Workspace**

This directory holds all **runtime-generated** results from Munge, BTR, and Workbench jobs. It is **user- and tool-writable**, and may be overwritten or cleaned between runs.

**🔹 Contender Files**

Contenders,-L4-P6-DC-MC-SP.txt

* Stores best sequences per InputType (e.g., DC = Combined)
* Sorted by score — top line is the winner

**🔹 State Files**

State,-L5-P6-DN-MC-SP.json

* Periodic Munge snapshots
* Enables automatic resume on rerun

**🔹 Fail Databases**

MungeFailDB,-P6-DR-MC-SP.db  
BTRFailDB,-P0-DS-MC-SP.db

* SQLite logs of sequences that failed
* Prevents redundant attempts during future jobs

**🔸 Contender Archive/ — Read-Only, Dev-Populated**

This directory holds **official, shipped results** created by the developer (e.g., top contenders, gold-standard failure DBs). It is intended for **analysis, comparison, or machine learning**, **not** for ongoing writing by users or processes.

⚠️ There is no mechanism to regenerate or overwrite this directory from the Workbench — by design.

**Examples:**

* Prebuilt contender files:  
  Contenders,-L5-P6-DR-MC-SP.txt
* Final fail DBs:  
  MungeFailDB,-P6-DR-MC-SP.db
* Preserved Lx snapshots:  
  State,-L5-P6-DN-MC-SP.json

**🔸 Data/ — Static Inputs**

* Frankenstein.txt / .bin – Natural text corpus
* randoms.bin – Pre-generated entropy
* UserData.bin – Optional user-provided input

These power the InputType settings (DN, DR, DU) and are **read-only** during scoring.

**🔸 Docs/ — User & Dev Documentation**

* 01\_Introduction.docx → 13\_Interpreting Mango's Practical Scoring Model.docx
* Live reference for commands, scoring, transforms, usage

**🔸 Runtime/ — Published Build Output**

Post-publish binaries for:

* Workbench/
* MangoAC/, MangoBM/, MangoZI/

No editing expected — these are used for distribution and deployment.

**🔸 Workbench State & Settings**

* ConsoleState.json – Shell history and preferences
* GlobalSettings.json – Central config: rounds, mode, scoring, etc.
  + 📌 Always lives next to the executable (AppContext.BaseDirectory)
* MangoConfig.txt – Signal file for job status (e.g., “Munge in progress”)

**📌 Final Notes**

* Output/ is a live workspace — tools may overwrite files here.
* Contender Archive/ is fixed — don’t write to it in normal flows.
* Auto-resume, fail avoidance, and scoring depend on these conventions.

For reproducible results, ensure your automation respects the roles of each file and folder.