**8. Files on Disk**

📁 *Preliminary Reference for Workbench Artifacts*

This section documents the files generated by the Mango Workbench and cryptographic engine. It defines naming conventions, purposes, and usage contexts. While intended as an internal reference, it also serves as a foundational guide for users extending or automating Mango workflows.

**🧬 File Naming Convention**

Most generated files follow a structured pattern:

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

**🔤 Components:**

| **Tag** | **Meaning** |
| --- | --- |
| **Name** | File category: Contenders, MungeFailDB, etc. |
| **L<N>** | Munge Level — e.g., L1 to L5, denoting sequence depth |
| **P<P>** | PassCount threshold — number of metrics a sequence must pass |
| **D<T>** | InputType/DataType: DC = Combined, DN = Natural, DR = Random, DS = Sequence |
| **M<M>** | Evaluation Mode: C = Cryptographic, E = Exploratory |
| **S<S>** | Scoring Model: SF = Practical Score, ST = Metric Score |

Tags are joined using commas and dashes for clarity and machine parsing.

**📄 Key File Types and Their Purpose**

**🔹 Contender Files**

**Example:**  
Contenders,-L4-P6-DC-MC-SF.txt

* Stores top-ranked sequences discovered during Munge.
* One file per InputType (e.g., DC = Combined).
* Sorted by descending score — the first sequence is the winner.

**🔹 State Files**

**Example:**  
State,-L5-P6-DN-MC-SF.json

* Periodic snapshots of Munge progress.
* Enables automatic resumption after interruption (e.g., Ctrl-C or power loss).
* Just re-run Munge — it will pick up where it left off.

**🔹 Fail Databases**

**Examples:**  
MungeFailDB,-P6-DR-MC-SF.db  
BTRFailDB,-P0-DS-MC-SF.db

* SQLite databases that log failing sequences.
* Used to skip known underperformers in future runs.
* Especially helpful during multi-day L5 Munge jobs.

**🔹 Benchmarking Files**

**Examples:**

* TransformBenchmarkResults.json
* TransformBenchmarkResults.txt
* TransformProfileResults.json
* Contains profiling data: average runtime per transform, system timings, etc.
* Used internally for estimating completion times and profiling bottlenecks.

**🔹 Input Datasets**

**Files:**

* Frankenstein.txt / Frankenstein.bin — literary corpus for Natural InputType
* randoms.bin — pre-generated random byte arrays for Random InputType

**Notes:**

* Combined input is built dynamically from all three classes.
* Sequence input is algorithmically generated (e.g., counters, monotonic ramps).
* Canned input ensures score stability between runs.

**🔹 Workbench State & Settings**

**Files:**

* ConsoleState.json — Command history and last-used console input
* GlobalSettings.json — All config values (Rounds, PassCount, Mode, etc.)
* MangoConfig.txt — Runtime execution flag used during long jobs (e.g., Munge in progress)

**📌 Final Notes**

This document is a living reference and will expand as new tooling, export formats, and configuration systems are introduced.

For best results, all automated systems (CI pipelines, regression tests, Munge schedulers) should adhere to these conventions for interoperability and reproducibility.