Project Workflow Guide

Version: 0.30.30-Beta Date: 2025-08-05

--- Revision History ---

[0.30.30-Beta] - 2025-08-05

- No functional changes. Synchronizing version numbers.

[0.30.10-Beta] - 2025-08-05

- Added Section 4.3 to formally define the "File State Algorithm" for

determining the definitive version of all project files.

- Amended Section 8 to clarify that all AI communication is to be

treated as "technical writing," prioritizing precision over conversational style.

[0.30.9-Beta] - 2025-08-05

- Amended Section 9.1 to require bundle status for single-bundle updates.

[0.30.8-Beta] - 2025-08-05

- Amended Section 9 to require bundle status

updates.

[0.30.7-Beta] - 2025-08-05

- Amended Section 8 to clarify the "technical writing" rule.

[0.30.6-Beta] - 2025-08-05

- Renamed "Unit Testing Protocol" to "Pre-Flight Check Protocol" (Section 12).

[0.30.5-Beta] - 2025-08-05

- Amended Section 4.2 to specify a concise checksum verification report.

[0.30.4-Beta] - 2025-08-05

- Added Section 12: Unit Testing Protocol.

- Added Section 13: File Naming Convention Protocol.

- Amended Section 5 (Development Protocol) with rule 5.3.

[0.30.2-Beta] - 2025-08-05

- Expanded Section 8.1 to include all standard SI prefixes.

[0.30.1-Beta] - 2025-08-05

- Added section 8.1 to explicitly define technical prefixes.

[0.30.0-Beta] - 2025-08-05

- Initial release of Version 0.30.0-Beta.

[cite_start]This document outlines the standard operating procedures for the collaborative development of the Contest Log Analyzer. [cite: 2312] [cite_start]**The primary audience for this document is the Gemini AI agent.** [cite: 2313]

[cite_start]Its core purpose is to serve as a persistent set of rules and context. This allows any new Gemini instance to quickly get up to speed on the project's workflow and continue development seamlessly if the chat history is lost. [cite: 2313] [cite_start]Adhering to this workflow ensures consistency and prevents data loss. [cite: 2314]

Guiding Principles

1. [cite_start]**Trust the User's Diagnostics.** When the user reports a bug, their description of the symptoms (e.g., "the multipliers are too high," "the report is all zeros") should be treated as the ground truth. [cite: 2315] [cite_start]The AI's primary task is to find the root cause of those specific, observed symptoms, not to propose alternative theories about what might be wrong. [cite: 2316]
2. [cite_start]**Debug "A Priori" When Stuck.** If an initial bug fix fails, do not simply try to patch the failed fix. [cite: 2317] [cite_start]Instead, follow the "a priori" method: discard the previous theory and re-examine the current, complete state of the code and the error logs from a fresh perspective. [cite: 2318] [cite_start]When in doubt, the most effective next step is to add diagnostic print statements to gather more data. [cite: 2319]
3. [cite_start]**Prefer Logic in Code, Not Data.** The project's design philosophy is to keep the `.json` definition files as simple, declarative maps. [cite: 2320] [cite_start]All complex, conditional, or contest-specific logic should be implemented in dedicated Python modules (e.g., `cq_160_multiplier_resolver.py`). [cite: 2321] [cite_start]This makes the system more robust and easier to maintain. [cite: 2322]

---

1. Project File Input

[cite_start]All project source files (`.py`, `.json`) and documentation files (`.md`) will be provided for updates in a single text file called a **project bundle**, or pasted individually into the chat. [cite: 2323] [cite_start]The bundle uses a simple text header to separate each file: [cite: 2324]