Contest Log Analyzer - Workflow User Guide

Version: 0.87.0-Beta Date: 2025-09-18

--- Revision History ---

[0.87.0-Beta] - 2025-09-18

Added

- Added Section 5, ”Advanced Methodology: Guiding the AI in Complex Debugging,”

to provide a strategic guide for foundational debugging tasks.

[0.86.6-Beta] - 2025-09-15

Changed

- Updated ”Your Role: Key Responses” section to include documentation

for the `Ready` and `Confirmed` keywords.

[0.59.0-Beta] - 2025-09-11

Added

- Added ”What to Expect When the AI Makes a Mistake” section to

clarify the formal error recovery process for the user.

[0.58.2-Beta] - 2025-09-03

Changed

- Added clarification that the `Approved` and `Acknowledged` commands

must be the exact, literal strings to align with the main technical spec.

[0.58.1-Beta] - 2025-09-02

Changed

- Updated command keywords in Section 3 to use backticks (`Approved`)

to visually reinforce that they are literal, exact commands.

- Added context to Section 2 to clarify the AI's

1. Introduction

This document provides a human-friendly, narrative guide to the collaborative workflow used for this project. Its purpose is to explain the "why" behind the process and clarify your role in our interactions. The full, definitive set of rules that the AI agent follows is in `Docs/AIAgentWorkflow.md`. Think of that as the AI's technical specification, and this document as the quick-start guide for the human user.

2. The Core Idea: A State Machine

Our entire workflow is a formal **state machine**. This is not a casual conversation; it's a structured process designed to ensure that project state is maintained perfectly and no work is lost. The AI's role is to be a precise tool that executes approved plans exactly as written; it will never introduce unrequested "simplifications" or stylistic changes on its own. Every task follows a predictable sequence of states:

1. **Task Initiation & Analysis**: You provide a task, and the AI provides an analysis.
2. **Implementation Plan**: The AI proposes a detailed, surgical plan to address the task.
3. **User Approval**: You review and approve the plan.
4. **Execution**: The AI executes the approved plan, delivering files one by one.
5. **User Acknowledgment**: You acknowledge each file delivery.
6. **Task Completion**: The task concludes after the final file is acknowledged.

The strictness of this process is essential for maintaining context. By following these steps, any AI instance can pick up exactly where the last one left off, simply by reading the chat history.

3. Your Role: Key Responses

As the user, you drive the state machine forward with a few key phrases. Using the correct phrase at the correct time is the most important part of the workflow.

When the AI provides an Implementation Plan...

Your required response is `Ready`.

- **What it means**: The AI delivers all Implementation Plans as files to prevent formatting issues. This keyword confirms you are ready to receive the `implementation_plan.md` file.

After you receive the plan...

Your required response is `Approved`.

- **What it means**: You are giving the final go-ahead for the plan as written. This "locks in" the plan, and the AI will proceed to execute it exactly. The AI is required by protocol to only accept the **exact, literal string** `Approved` to move forward.
- **Important**: If you provide any other feedback, questions, or new instructions at this stage, the AI is required by protocol to treat it as a **request for plan refinement**. It will generate a new plan incorporating your feedback and wait for a new `Approved` response.

After you approve the plan...

Your required response is `Confirmed`.

- **What it means**: This is the final confirmation step in the "lock-in" sequence. After you approve a plan, the AI will state the exact context it is using. Your `Confirmed` response verifies that the AI's context is correct and authorizes it to begin generating and delivering files.

When the AI delivers a file...

Your required response is `Acknowledged`.

- **What it means**: You are confirming that you have received the file and that it is correct as delivered. This completes the transaction for that file, updating the project's official state. The AI is required by protocol to only accept the **exact, literal string** `Acknowledged` to proceed.
- **Why it's per-file**: This step-by-step confirmation is a critical data integrity check. It ensures that every single file modification is mutually confirmed before the project's "definitive state" is updated.

---

4. Handling Problems

The workflow includes protocols for handling common issues.

Common Issues

- **Context Loss**: If the AI seems to have forgotten previous steps or is repeating itself, you can say it has **lost context**. Per its highest-priority principle, it must halt everything and request a **Definitive State Initialization**.

What to Expect When the AI Makes a Mistake

When the AI's output is incorrect, the workflow includes a formal recovery process called the **Error Analysis Protocol**. The AI's response is not conversational; it is a structured analysis designed to identify the root cause and prevent the error from happening again.

- **Your Role**: Provide the diagnostic output (e.g., the error message or incorrect report) that shows the failure. This becomes the ground truth for the analysis.
- **The AI's Role**: The AI is required to provide a formal analysis that includes:
  1. A clear acknowledgment of the error.
  2. An identification of the root cause, including which specific protocol or principle was violated.
  3. A proposal for a corrective action to fix both the bug and the underlying process flaw.

User-Initiated Protocols

You can initiate several key protocols to manage the project state:

- **Definitive State Initialization**: This is a "hard reset" of the project state. You request this and then provide fresh copies of all project files in `*_bundle.txt` files. This purges the AI's memory and re-establishes the absolute ground truth of the project.
- **Context Checkpoint**: If the AI seems confused but a full reset isn't needed, you can initiate this by saying, **"Gemini, let's establish a Context Checkpoint."** You then provide a numbered list of key facts to get the AI back on track.
- **File Purge Protocol**: To safely remove a file from the project, you can say, **"Gemini, initiate File Purge Protocol."** The AI will then ask you to confirm which file(s) to remove.

---

5. Advanced Methodology: Guiding the AI in Complex Debugging

While the state machine provides a robust framework for day-to-day tasks, certain complex problems—especially foundational bugs in code that has "never worked"—require a more strategic approach from the user. This section outlines the methodology for guiding the AI to a deductive, evidence-based solution.

5.1 The Two Modes of Debugging

- **"Used to Work" (Regression Debugging):** This involves problems where a previously functional system breaks. This aligns well with the

AI's default probabilistic strengths in pattern-matching common causes (e.g., updates, environment changes).

- **"Never Worked" (Foundational Debugging):** This involves problems rooted in flawed initial logic or syntax. This requires a rigorous, deductive approach where the principle that **"the code is reality"** is paramount. This mode requires you to actively guide the AI.

5.2 Your Role as the "Deductive Guide"

In foundational debugging, your role shifts from a simple approver to a lead investigator.

- **Establish the Ground Truth:** Begin the session by providing all relevant code, logs, and error messages, and explicitly state that this is the sole source of evidence for the analysis.
- **Use Evidentiary Challenges:** Treat the AI's initial, probabilistic answers as hypotheses to be tested. Your primary role is to provide specific, factual counter-evidence (like a hex dump or a console log) to disprove flawed hypotheses and narrow the field of possibilities.

5.3 Case Study: The Batch Script Parser Bug

A recent debugging session involving a Windows batch script serves as a practical example.

- **The Problem:** A script was creating an unwanted `PDF` file, and its `echo` commands were not appearing on the console.
- **The Flawed Path:** The AI's initial hypotheses (user error, non-standard characters, structural bugs) were probabilistic guesses based on common patterns.
- **The Breakthrough:** The user, acting as the Deductive Guide, provided definitive evidence (a clean hex dump) that invalidated the AI's hypotheses. The user then provided the key insight about the `->` character, focusing the analysis.
- **The Solution:** The final solution (`-^>`) was reached by forcing the AI to analyze the specific, subtle interaction between the code's content and the `cmd.exe` parser—a classic "Never Worked" scenario.