# **Technical Research Report: A Stateful Coding Agent with a Final Execution Report**

Version: 25.0

Date: 18 June 2025

Status: Approved

1. Abstract

This report addresses a critical flaw discovered in the v7 Coding Agent: a lack of intra-execution state persistence. While the agent's "Read-Think-Write" cycle was logically sound for a single modification, it failed to persist changes between steps within the same execution run. This resulted in the agent overwriting its own work, as each step re-read the original, unmodified file from disk. The solution is to refactor the agent to maintain an in-memory representation of the codebase during its execution. Furthermore, to complete our symmetrical communication protocol, the Coding Agent will now conclude its run by generating a final, structured coding\_execution\_report.json. This artifact will contain the complete, final proposed content for all modified files, serving as an unambiguous contract for review and for the future Master Project Manager Agent.

2. Research: The Failure of Stateless Execution

The v7 execution log clearly showed that the agent was not building on its previous work. It would "implement" the first method for coding\_agent.py, then, for the second step, it would re-read the original file and implement the second method, discarding the first.

This is because the agent's state was only passed between graph *nodes*, not between *iterations within the same node*. The code\_execution\_node was a loop, but the state of the files it was modifying was not being maintained within that loop. It was relying on an external, unchanging source (the file system) for its context at each step.

3. The Solution: In-Memory State and a Final Report

The v8 architecture solves this with two key enhancements to the Coding Agent:

1. In-Memory File Cache: The code\_execution\_node will now maintain a local dictionary (file\_contents\_in\_memory). When a step begins, it first checks this cache for the target file. If not found, it reads from the disk and populates the cache. After the "Think" step generates new code, it updates the cached version. All subsequent operations on that file within the same run will use the updated, in-memory version. This ensures changes are incremental and preserved.
2. Structured Execution Report (coding\_execution\_report.json): After the execution loop completes, a new finalize\_coding\_report node will be called. This node takes the final state of the file\_contents\_in\_memory cache and compiles it into a clean JSON artifact. This report will contain the full, final proposed code for every file the agent intended to change, providing a perfect, auditable summary of its work for the next agent in the pipeline (the Master PM).

This architecture creates a more intelligent and robust Coding Agent that correctly handles multi-step modifications and adheres to the project's core principle of structured, agent-to-agent communication.