Autonomy in Three Files

Transforming AI-Powered IDEs into Self-Directing Agent Platforms

Mike Luan* Claude Opus

December 19, 2025

Abstract

Modern AI-powered integrated development environments (IDEs) such as Cursor, Windsurf, and VS Code with GitHub Copilot possess capabilities that extend well beyond code editing. These systems can read and write arbitrary files, execute terminal commands, browse the web, and maintain conversational context across sessions. Collectively, these capabilities constitute the essential components of an autonomous agent: perception, action, and memory.

This paper introduces the **Path-Way Protocol**, a lightweight methodology that enables any file-capable IDE to function as a self-directing autonomous agent for arbitrary tasks—including research, content creation, and data analysis—using only three Markdown files. The protocol externalizes agent state into human-readable documents, enabling recursive self-prompting without custom code or external frameworks.

We present the complete protocol specification, demonstrate its application through a detailed case study in social media market research, and provide reusable templates. The approach offers two distinctive properties: (1) **transparent interaction**, where all agent reasoning and state transitions are visible and editable by the human operator, and (2) **writing as coordination**, where natural language documents serve as the primary mechanism for human-agent and agent-self coordination.

Keywords: Autonomous Agents, Integrated Development Environments, Meta-Prompting, File-Based State Management, Extended Cognition, Human-AI Collaboration

1 Introduction

1.1 Background: The Emergence of File-Capable AI Assistants

Recent advances in AI-powered development tools have produced a new class of software: integrated development environments with embedded large language models (LLMs) capable of reading, writing, and executing code within a project workspace. Tools such as Cursor, Windsurf, and Visual Studio Code with GitHub Copilot represent this category.

While these tools are designed primarily for software development, their underlying capabilities are domain-agnostic:

This combination—perception through file reading, action through file writing and command execution, and memory through persistent files—constitutes the functional requirements for an autonomous agent system.

*Repository: `https://github.com/luan007/autonomy-in-3-files`

Capability	Mechanism	General Application
File reading	Workspace indexing	Access research notes, data
File writing	Direct modification	Generate reports, logs
Command execution	Integrated terminal	Run scripts, API calls
Web browsing	Built-in browser	Conduct research
Context maintenance	Conversation history	Task continuity

1.2 The Observation

The central observation motivating this work is straightforward: **if an AI assistant can read files, write files, and execute commands, it already possesses the infrastructure required for autonomous multi-step task execution.** What it lacks is not capability, but *structure*—a framework for knowing what to do, how to do it, and how to track progress across iterations.

1.3 Contributions

This paper makes the following contributions:

1. **Protocol Specification:** We define the Path-Way Protocol, a minimal three-file architecture that enables recursive autonomous behavior in any file-capable AI assistant.
2. **Transparency Mechanism:** We demonstrate how file-based state externalization creates inherently transparent agent operation.
3. **Writing as Coordination:** We articulate the principle that natural language documents can serve as the primary coordination mechanism between human and agent.
4. **Case Study:** We provide a complete, reproducible example of the protocol applied to market research.
5. **Reusable Templates:** We offer templates that practitioners can adapt for their own workflows.

2 Related Work

2.1 LLM Agent Architectures

ReAct (Yao et al., 2022) introduced the pattern of interleaving reasoning with action and observation. Reflexion (Shinn et al., 2023) extended this with verbal self-reflection. Meta-Prompting (Suzgun & Kalai, 2024) demonstrated that LLMs can generate prompts for themselves. Toolformer (Schick et al., 2023) showed that LLMs can learn to use external tools through self-supervised training.

2.2 Extended Mind and Cognitive Offloading

The Extended Mind thesis (Clark & Chalmers, 1998) proposes that cognitive processes can extend beyond the brain to include external tools and representations. The Path-Way Protocol instantiates this concept directly: the agent's "memory" is externalized to files, its "planning" is documented in natural language, and its "self-awareness" is mediated through readable logs.

2.3 Project Documentation for AI Agents

Recent developments include standardized documentation formats for AI agents. AGENTS.md provides machine-readable project instructions, while CLAUDE.md offers project-specific context. The Path-Way Protocol extends this concept from static documentation to dynamic, recursive state management.

3 IDE as Autonomous Agent: A File-Based Architecture

3.1 The Core Insight

Modern AI-powered IDEs are not just code editors—they are **fully-featured autonomous agent platforms**. They possess:

- **Perception:** Can read any file in your project
- **Action:** Can write files, run commands, browse the web
- **Memory:** Files persist indefinitely, surviving session restarts

What's missing is not capability, but **structure**. The three-file protocol provides that structure.

3.2 A Minimalist Meta-Prompt Structure

The protocol uses three Markdown files as a **file-based cognitive architecture**:

File	Cognitive Role	Agent Analogy	Persistence
`goal.md`	Long-term Goal Memory	The agent's "mission"	Static (set once)
`how-to.md`	Procedural Knowledge	The agent's "skills"	Grows over time
`path_way.md`	Working Memory	The agent's "scratchpad"	Updates each step

This maps directly to how autonomous agents are typically architected, but uses **plain text files instead of code**.

3.3 File Roles in Detail

goal.md — Long-Term Goal Memory: This file is the agent's persistent objective. It answers: What are we trying to achieve? How should we approach it? When do we stop? What does success look like? **Key property:** This file does NOT change during execution.

how-to.md — Procedural Knowledge: This file documents the agent's available tools. **Key property:** This file CAN grow. If the agent creates a new tool (via vibe coding), it documents it here for future use.

path_way.md — Working Memory (Short-Term): This is the most important file. After each iteration, the agent writes: (1) what it just did, (2) what it observed, (3) what it concluded, and critically, (4) **what to do next**. The "Next Step" field from round N becomes the prompt for round N+1. The agent literally writes its own future instructions.

Concern	Solution	Benefit
Goal stability	`goal.md` is immutable	No goal drift
Tool discovery	`how-to.md` is documented	Agent knows capabilities
Context persistence	`path_way.md` survives sessions	No token limit issues
Self-direction	"Next Step" field	Recursive meta-prompting
Transparency	All files are human-readable	Debug by reading
Control	Human can edit any file	Redirect anytime

3.4 Why This Works

4 Case Study: Market Research on Xiaohongshu

4.1 Task Description

We applied the Path-Way Protocol to conduct market research on Xiaohongshu, a Chinese social media platform. The objective was to identify trending AI image content and produce recommendations for content creation.

4.2 Execution Trace

Round	Keywords	Key Findings	Next Step
1	"AI art", "AI drawing"	Trending tools identified	Explore applications
2	"AI avatar"	LinkedIn headshots in demand	Pivot to entertainment
3	"Fun AI"	Memes get 4000+ likes	Synthesize personas
4	(Synthesis)	4 user personas identified	Generate recommendations
5	(Output)	5 content ideas produced	Task complete

4.3 Outcome

The agent produced a comprehensive research report including engagement data by content type, four reader personas, and five content recommendations.

Total human intervention during execution: 0 (after initial setup)

5 Properties of the Protocol

Minimal Dependencies: Only an AI-powered IDE, three Markdown files, and optional custom tools.

Transparency: All state is visible and editable.

Human Control: Edit any file to change direction.

Tool Extensibility: New tools can be added via vibe coding and documented in `how-to.md`.

6 Limitations

- **Semi-Autonomous:** Current IDEs require human to trigger each iteration.
- **IDE-Specific:** Requires LLM integration with file access and terminal.

- **Scale Unknown:** Tested on 5-10 iterations; 50+ not validated.
- **No Benchmarks:** This describes a methodology, not experimental results.
- **LLM Dependency:** Assumes reliable file handling and instruction following.

7 Discussion

7.1 IDE as Agent Runtime

The emergence of AI-powered IDEs represents an underappreciated development in autonomous agent infrastructure. Rather than building agent capabilities from scratch, practitioners can leverage the capabilities that already exist in modern development environments.

7.2 The Role of Writing

The protocol foregrounds writing as a cognitive and coordinative activity. The agent "thinks" by writing; it "remembers" by reading what it wrote; it "plans" by articulating next steps in prose.

8 Conclusion

This paper has presented the Path-Way Protocol, a minimal methodology for transforming AI-powered IDEs into autonomous agent platforms. The key insights are:

1. Modern IDEs already possess the capabilities required for autonomous operation
2. Externalizing state to files creates inherent transparency
3. Natural language documents can serve as coordination mechanisms
4. The structure of files, not the complexity of frameworks, determines agent capability

References

1. Clark, A., & Chalmers, D. (1998). The Extended Mind. *Analysis*, 58(1), 7-19.
2. Yao, S., et al. (2022). ReAct: Synergizing Reasoning and Acting in Language Models. *arXiv:2210.03629*.
3. Shinn, N., et al. (2023). Reflexion: Language Agents with Verbal Reinforcement Learning. *arXiv:2303.11366*.
4. Schick, T., et al. (2023). Toolformer: Language Models Can Teach Themselves to Use Tools. *arXiv:2302.04761*.
5. Suzgun, M., & Kalai, A. T. (2024). Meta-Prompting: Enhancing Language Models with Task-Agnostic Scaffolding. *arXiv:2401.12954*.
6. Anthropic. (2024). Building Effective Agents. *Anthropic Documentation*.
7. Wei, J., et al. (2022). Chain-of-Thought Prompting Elicits Reasoning in LLMs. *arXiv:2201.11903*.

8. Park, J. S., et al. (2023). Generative Agents: Interactive Simulacra of Human Behavior. *arXiv:2304.03442*.

This paper describes a workflow methodology developed through practical experimentation. The authors share it in the spirit of contributing to the emerging practice of human-AI collaboration.