

# Research Flow Notebook

AITP Protocol v3: From Compliance Officer to Adversarial Collaborator  
`code_method lane`

AITP Protocol v4

2026-04-24

## Abstract

This notebook records the complete research trajectory: sources, conventions, derivation attempts, validated results, and open questions. Mode: `explore`.

## Contents

<b>1</b>	<b>Research Question</b>	<b>2</b>
<b>2</b>	<b>Source Landscape</b>	<b>2</b>
<b>3</b>	<b>Conventions &amp; Notation</b>	<b>3</b>
<b>4</b>	<b>Mode &amp; Session History</b>	<b>5</b>
<b>5</b>	<b>Derivation Journey</b>	<b>5</b>
<b>6</b>	<b>Synthesis &amp; Claims</b>	<b>5</b>
<b>7</b>	<b>Validation</b>	<b>5</b>
<b>8</b>	<b>Negative Results &amp; Open Questions</b>	<b>5</b>

# 1 Research Question

## Bounded Question

Refactor AITP from 50-tool compliance-checking protocol to ~15-tool adversarial-collaborator protocol: Python handles only file persistence + semantic search, all physics judgment lives in skill files as Socratic/role-based prompts.

**Scope:** MCP server `brain/mcp_server.py` and `brain/state_model.py`, plus skill files. Does NOT include: UI, external MCP integrations, Zotero/Obsidian sync, numerical computing infrastructure. Study mode pipeline is in-scope because it feeds L2. **Target quantities:** 1) Tools: cut from 50 to ~15. 2) Gate evaluators: remove content-quality pretensions, keep only file-existence checks. 3) L2 search: semantic from substring. 4) Skill files: rewritten as role-based adversarial prompts. 5) Autonomous loops: removed or gated behind human review. 6) All existing tests must still pass (behavior preserved where tools remain).

# 2 Source Landscape

Source	Title	Fidelity	Relation
adversarial-collaborator-design	Adversarial Collaborator Design: Skills as Socratic Interloc	local_source	
current-mcp-server-codebase	Current AITP MCP Server v2: 50 tools, <code>brain/mcp_server.py</code> +	local_source	
protocol-critique-20260424	26-Problem Protocol Critique from Theoretical Physicist Pers	local_source	

## Source Basis Analysis

### Source Basis

#### Core Sources

1. **current-mcp-server-codebase:** `brain/mcp_server.py` (3273 lines, 50 `@mcp.tool()`), `brain/state_model.py` (912 lines, gate evaluators, templates, constants). This is the substrate to be refactored. Every tool function, every gate evaluator, every template is candidate for deletion or rewrite.
1. **protocol-critique-20260424:** The 26-problem analysis from a theoretical physicist's perspective. Organized by: gate quality (4 problems), L2 knowledge representation (6), mathematical verification (4), trust model (4), workflow (5), scaling (3). This is the "what's broken" reference.
1. **adversarial-collaborator-design:** Design direction note establishing: Python = storage + search only; Skills = role-based adversarial prompts; L2 = conversationally curated; no autonomous LLM-self-validation loops. This is the "where we're going" reference.

## Peripheral Sources

- Existing 7 AITP topics (real usage data showing what's working and what's not)
- Test suite ~5000 lines (what behaviors are currently promised)
- `.claude/skills/` v2 skill files (reference for what good skills look like)

## Source Roles

Source	Role
<code>mcp_server.py</code> + <code>state_model.py</code>	Primary substrate what gets cut/rewritten
<code>protocol-critique</code>	Requirements document what must be fixed
<code>adversarial-collaborator-design</code>	Architecture vision how it should work
Test suite	Compatibility constraint what must not break
Existing 7 topics	Migration constraint data must survive

## Reading Depth

All sources are local, relatively short (< 5000 lines total), and already deeply analyzed. No need for extended reading phase move quickly to design (L3).

## Why Each Source Matters

- **codebase**: You can't refactor what you haven't read
- **critique**: 26 specific bugs/gaps, each is a design constraint
- **design note**: The positive vision if the critique says "don't do X", the design note says "do Y instead"

# 3 Conventions & Notation

## Convention Snapshot

### Notation Choices

- **Protocol layers**: L0 (discover) → L1 (read/frame) → L2 (cross-topic knowledge graph) → L3 (derive, research or study mode) → L4 (validate) → L5 (write)
- **Tool naming**: `aitp_<verb>_<noun>` e.g., `aitp_create_l2_node`, `aitp_query_l2`
- **Skill naming**: `skill-<stage>-<subplane>.md` for stage-specific, descriptive names for cross-cutting skills
- **Adversarial collaborator**: The persona embedded in each skill file that asks Socratic/-critical questions modeled after a skeptical but constructive peer reviewer or co-author
- **"Python never judges"**: A design invariant no Python function returns a content-quality verdict, trust assessment, or physics correctness claim

## Unit Conventions

- Python 3.10+, FastMCP framework
- pytest for testing, PyYAML for frontmatter parsing
- All persistence: Markdown files with YAML frontmatter (no JSON, no SQL, no embeddings DB)
- File naming: snake\_case for slugs, kebab-case for artifact directories

## Sign Conventions

- **Remove, don't add:** Default action is deleting tools, not creating new ones
- **Skill over Python:** When in doubt, put logic in skill files (LLM-mediated), not in Python (deterministic but dumb)
- **Storage is sacred:** File I/O must be atomic and reliable this is Python's only real responsibility

## Metric Or Coordinate Conventions

- Code measured in: tool count (~15 target), lines of Python (targeting significant reduction from 3273), test coverage (must not regress)
- Quality measured in: how well the protocol enables a real physics research conversation, not in checkboxes checked

## Unresolved Tensions

1. Semantic search can we get good enough results without embedding models? (Pure Python: token overlap + concept aliases + LaTeX normalization)
2. Gate evaluation if Python doesn't check content, what signals "you're not ready to advance"? Answer: skill-level adversarial questions that the agent must answer honestly
3. Backward compatibilit

**Notation:** Protocol layers: L0(discover)→L1(read)→L2(cross-topic KG)→L3(derive)→L4(validate)→L5(write)  
Tool naming: aitp\_<verb>\_<noun>. Skill naming: skill-<stage>-<subplane>.md. Adversarial collaborator: the agent-in-skill that asks Socratic questions, not checklist items.

**Units:** Code: Python 3.10+, FastMCP framework, pytest, PyYAML. Storage: Markdown files with YAML frontmatter (no JSON, no SQL). File naming: snake\_case for slugs, kebab-case for artifact directories.

## 4 Mode & Session History

Axis	Value
Topic slug	aitp-protocol-v3
Status	new
Mode	explore
Lane	code_method
L3 mode	research

### Session Log

**Session Log: AITP Protocol v3: From Compliance Officer to Adversarial Collaborator**

### Sessions

## 5 Derivation Journey

(No structured derivation recorded.)

## 6 Synthesis & Claims

## 7 Validation

(No L4 reviews submitted.)

## 8 Negative Results & Open Questions

(No negative results or open questions formally recorded.)