AgentOS Advisor Lane

Invitation to Co-Design with Sean and JP-SR

Prepared for: Craig

From: Sean Parsons

Date: August 10, 2026

Status: Invitation and discussion brief — no system access or implementation authority granted

PAGE 1 — The idea and why I am asking you

Craig,

Our breakfast conversation about memory, databases, and AgentOS made me realize that I need more than a one-time answer to a list of technical questions. I would like to ask whether you would help me shape a small feature we are calling the AgentOS Advisor Lane.

The idea is to create a secure, repeatable way for a small number of trusted specialists to review a deliberately limited view of an AgentOS problem, ask follow-up questions, challenge assumptions, and return candid recommendations. The advisor should be able to work iteratively without receiving access to the real operating system, private data, credentials, or production authority.

I would remain the project owner. JP-SR, my coordinating AgentOS assistant, would prepare approved context, support the design process, and record decisions. I am asking you to participate as an external technical co-designer—especially around data architecture, memory, identity, access, auditability, versioning, rollback, and practical maintenance.

Why your input matters

You bring long professional experience in software, engineering-team leadership, and database hosting. You are also someone I know personally and trust to be candid. I want the technical design to remain safe even when the advisor is trusted: personal trust should improve the quality of the discussion, but the platform should still enforce clear boundaries.

Our first real use case: AgentOS memory

AgentOS currently coordinates six computers:

- one coordinating process that reviews and promotes shared long-term memory;
- one more-independent reviewer that proposes additions but cannot self-approve;
- four bounded worker processes that return task results and evidence.

Reviewed institutional knowledge currently lives in a private Markdown notes vault viewed through Obsidian. Project repositories remain authoritative for code and versioned implementation files. We are considering whether Git-based change control, structured data/event storage, hybrid keyword/vector retrieval, RAG, or a graph layer would improve memory and workflow—or merely add complexity.

Instead of asking you to answer that once by email, I would like to use memory as the first test case for designing the Advisor Lane itself.

PAGE 2 — What we want to co-design

Intended advisor experience

1. A named advisor receives a personal invitation for one defined topic.
2. The advisor authenticates and sees the purpose, time window, approved context, exclusions, and requested output.
3. The advisor reviews a short packet and answers guided questions.
4. A bounded chat allows follow-up questions against only the approved context.
5. Answers cite the supplied material and refuse questions outside the approved scope.
6. The advisor ranks risks, recommends changes, and may attach a sketch or voice note.
7. The session expires and can be revoked early by Sean.
8. The original exchange is preserved as advisory evidence.
9. JP-SR separately summarizes each recommendation as accepted, revised, rejected, deferred, or blocked.
10. Nothing becomes policy, memory, code, or operating authority automatically.

Boundaries we expect the design to preserve

The advisor would not receive:

- access to Node 1 or the live control plane;
- live vault, filesystem, GitHub, email, messaging, browser, shell, or node-control tools;
- credentials, configuration, system prompts, private logs, or raw sessions;
- customer, tenant, banking, email, regulated, or unrelated project data;
- authority to modify approved memory or take external action.

The interactive model would initially use a static, sanitized, versioned context bundle. It would have no tools, live filesystem, network access, or persistent memory writes. Advisor input would remain untrusted data until reviewed, even when the person is trusted.

Questions where we especially want your help

1. What information would a professional advisor need to give strong advice without seeing the real system?
2. How should the advisor experience avoid becoming a long survey or homework assignment?
3. Which questions should the bounded system answer dynamically, and which should require Sean’s approval?
4. What should never be disclosed, even to a trusted advisor?
5. How should identity, authentication, session expiration, revocation, and transcript integrity work?
6. How should we prove exactly which version of each document the advisor received?
7. Should the knowledge layer be files, Git, a database/event ledger, retrieval index, or a hybrid?
8. How should recommendations move from external advice into tested, reviewed system changes?
9. What abuse, failure, or recovery tests would you require before granting anyone a login?
10. What is the smallest useful prototype, and what should we deliberately avoid building yet?

PAGE 3 — Proposed working process and first meeting

Proposed roles

- **Sean — Project Owner:** defines the objective, approves disclosed material, makes milestone decisions, and authorizes any external access.
- **JP-SR — Product/Architecture Lead:** prepares context, coordinates requirements and implementation, answers bounded questions, and records decisions and evidence.
- **Craig — External Technical Co-Designer:** challenges the design and helps define a useful, secure advisor experience.
- **JP-JR — Independent Workflow Reviewer:** tests clarity, handoffs, evidence, and the second-agent perspective.
- **Guardian Council — Security Reviewer:** challenges permissions, disclosure, injection, logging, revocation, and failure handling.
- **Builder and QA nodes:** may later create and test a synthetic offline prototype after Sean approves the design scope.

Review gates

1. Invitation: Craig agrees to participate and understands the boundaries.
2. Requirements: Sean, Craig, and JP-SR agree on the advisor experience and threat model.
3. Offline prototype: a synthetic, local-only interface is built and reviewed by screen share.
4. Security/QA: refusal, isolation, expiration, revocation, logging, upload, and recovery behavior are tested.
5. Craig pilot: one identity, one topic, one sanitized bundle, one time window, and no tools.
6. Owner review: Sean decides which recommendations become approved design changes.
7. Reusable lane: only after the Craig pilot succeeds would the pattern be generalized for other trusted specialists.

First meeting proposal

A 45–60 minute call or screen share would be enough to begin. There is no deadline pressure. The agenda would be:

1. Ten minutes — current AgentOS environment and why the Advisor Lane is needed.
2. Fifteen minutes — walk through the proposed advisor experience and security boundaries.
3. Fifteen minutes — use the memory architecture packet as a practical test case.
4. Ten minutes — identify the smallest prototype worth building.
5. Ten minutes — agree on open questions, roles, and the next review gate.

No login or system access is needed for this first meeting. The first objective is to let you help define the collaboration channel before we build or expose it.

What I am asking from you now

Please let me know whether you would be interested in helping co-design this feature and whether a call or screen share sometime in the coming days would work. You do not need to prepare a polished response. A candid initial reaction—especially what feels unnecessary, unsafe, or missing—is exactly what I want.

Thank you,

Sean