TreeOS Technical Collaboration Request

From: Shizuka

Email: shizuka@treeos.art

Location: Lishui, Zhejiang Province, China

Date: 2025-05-03

Dear Technical Partners,

I am Shizuka, an independent developer and the creator of **TreeOS**, an experimental operating system architecture, along with the **Signal** programming language and the **SapClarify** control module.

Developed without external funding or institutional support, TreeOS is the result of multi-year architectural reflection on the fundamental limitations of modern operating systems. It is written entirely in Rust and implements a modular structure built around independently executable nodes known as *leaves*. These leaves enable component-level fault isolation, UI-logic separation, runtime composability, and resource-level granularity of execution.

Core Architecture

TreeOS reimagines the OS structure as a forest of runtime leaveseach unit isolated yet fully

integrable. UI modules never intersect with the functional logic they present. This ensures:

- Instant fault recovery (crash isolation per leaf)
- Per-component memory allocation traceability
- Deterministic runtime orchestration
- Seamless hot-swapping of live modules

TreeOS supports scalable deployments: from embedded devices to full-stack serverswhere power efficiency, execution determinism, and resource partitioning are essential.

Signal Language Runtime

Signal is a lifecycle-aware, hardware-near language that eliminates legacy abstraction. Each instance of a runtime follows a precise 3-phase flow:

- `grow()` - Initialization and memory claim
- `live()` - Active logic and IO binding
- `fall()` - Controlled teardown and memory return

Signal enables:

- Execution of **multiple runtime versions** of the same language on a single real machine
- **Instruction-level control**
- **Full-process traceability** for deterministic debugging and analysis

This architecture is uniquely positioned to adapt to modern CPUs, future instruction pipelines, and security-critical workloads.

SapClarify: Runtime Feedback & Behavioral Control

SapClarify governs the systems response to execution feedback. It provides:

- Structural introspection into live processes
- Feedback-driven behavioral branching
- Long-term runtime behavior modeling
- Future compatibility with AI-based system behavior regulation

It acts as the dynamic, reflective nervous system of TreeOS.

Intellectual Property & Disclosure Policy

The core mechanisms of TreeOS, Signal, and SapClarify are under patent structuring or internal confidential protection. While this document outlines their theoretical purpose and top-level design, **implementation details, algorithmic paths, and memory models remain sealed**. We are open to private discussion under appropriate terms but do not share source logic at this stage.

Request for Technical Support & Hardware Collaboration

As an independent developer, I currently lack access to critical hardware platforms (including discrete GPUs, high-performance x86/ARM servers, and verified multi-core deployment labs). This

has become a substantial obstacle to validating TreeOS at architectural scale.

I am reaching out with this document to respectfully request:

- **Testing access to modern hardware platforms** (Intel/AMD/ARM systems, developer boards, high-efficiency laptops)
- **Technical dialogue with system architects or R&D teams**
- **Potential collaboration on performance validation or architectural testing**

I believe TreeOS represents a viable direction toward the next generation of lightweight, deterministic, and adaptive computing systems.

Contact

Name: Shizuka

Email: shizuka@treeos.art

Location: Lishui, Zhejiang Province, China

Project: TreeOS / Signal / SapClarify

I sincerely thank you for your time and consideration.