MEMORANDUM

To: Wakai Stakeholders

From: Dirk Harms-Merbitz

Date: Feb 14, 2026

Re: Unix for AI Ops — The Ecosystem Play

Unix for AI Ops

The Original and the Reimagined

The original Toyota FJ40 was built in the 1960s. Simple, rugged, mechanical. It did exactly what it needed to do with nothing wasted. Fifty years later, Toyota reimagined it. Same spirit, same philosophy, built for modern terrain.

That is what we are doing for Unix.

The Unix pioneers got the ideas right. Small tools. Pipes. Text streams. Composability. But the implementation carries fifty years of cruft. Escaping hell. Quoting nightmares. Sed expressions that break differently in every shell. Tools designed for a world where text was typed by humans, not generated by machines.

AI changes what flows through pipes. The volume is higher. The text is more complex. The commands are generated, not hand-typed. The old plumbing leaks under the new pressure.

We are rebuilding the plumbing.

Not a Tool. A Toolkit.

The first memo described toast — AI as a Unix pipe. Text in, text out. That was the starting point. But toast alone is just one tool in what needs to be an entire toolkit.

We wrote imessage from scratch in C. A clean binary that reads and sends iMessages from the command line. No frameworks. No dependencies. Minimal flags. It composes perfectly with toast and with everything else in the Unix ecosystem:

```
    imessage -e 'imessage | toast | imessage'
```

An iMessage bot in one line of code. Not because we built an "iMessage bot framework." Because we built two tools — imessage and toast — that each do one thing, and pipes handle the rest.

This is the pattern. For every system boundary that AI needs to cross, we build a clean, composable, C-native tool that handles it. Each one is simple. Each one pipes. Together they form a toolkit that makes AI operational — not in theory, but in production, today.

The Shell That Doesn't Escape

Every developer has lost hours to escaping. Nested quotes. Backslashes that need backslashes. Dollar signs that expand when they shouldn't. Sed expressions that work in bash but break in zsh. Single quotes inside double quotes inside command substitutions.

This was tolerable when humans typed every command. It is intolerable when AI generates them. LLMs produce commands with quotes and special characters constantly, and they get escaping wrong constantly. An entire class of errors exists only because the shell was designed in an era when text was simpler.

We are building a shell that treats strings as first-class citizens. No escaping gymnastics. Sed actually works. AI-generated commands execute on the first try. The shell becomes what it always should have been — transparent plumbing, not a puzzle.

This is not a patch. It is a foundation. Every tool in our ecosystem runs better when the shell beneath it handles text cleanly.

Lego Blocks

Each tool we build is a block. Small, self-contained, designed to connect. The value is not in any single block. The value is in what you build when you snap them together.

toast — AI text transformation. imessage — messaging pipe. The new shell — clean execution. Each one written in C. Each one minimal. Each one a perfect pipe citizen.

The roadmap adds more blocks. More messaging systems. More data sources. More output targets. Each one following the same design: text in, text out, composes with everything.

The more blocks we ship, the more combinations become possible. The more combinations, the more use cases. The more use cases, the more valuable the ecosystem. This is a flywheel, not a feature list.

The Moat

Anyone can wrap an LLM in a CLI. Dozens of tools do this. They compete on prompt quality and model selection — advantages that erode quarterly.

Nobody else is rebuilding the surrounding Unix tools from scratch in C, designed from the ground up for AI composability. Nobody else is building the shell. Nobody else is creating an ecosystem where every piece is engineered to snap together.

The moat is not toast. The moat is the toolkit. The moat is that a DevOps engineer using our ecosystem can build in one line what takes a framework thirty files to accomplish. And every new block we add makes every existing block more valuable.

Conclusion

Toast was the first tool. It will not be the last. We are building an ecosystem of composable, AI-native Unix tools — each one small, each one clean, each one written in C, each one a perfect pipe citizen. Together they form a toolkit that makes AI operational for the people who run infrastructure.

The moat is not any single tool. The moat is the ecosystem. Every new block makes every existing block more valuable. Every new combination unlocks use cases nobody planned for.

Unix for AI Ops. Put text in, get text out. The rest is plumbing.

https://linuxtoaster.com