System Regression Suite (weekly)

System Regression Suite

Owner: Angie Jones / Proveo **Implemented:** 2026-04-16 (System Evolution Phase 3, MC #8036)
Script: `~/system/tools/system-regression.sh`

Purpose

Catch system drift early. The ALAI "system that builds systems" accumulates state — LightRAG ingest, HiveMind intel, daemons, blueprints — and every small change risks breaking a seam. The regression suite runs 10 checks in under 10 seconds and exits non-zero if any critical component regresses.

The 10 Checks

#	Check	Tool	PASS condition
1	Tools health	`discover.js --verify`	exit 0
2	MC smoke	`mc.js list --limit 1`	exit 0
3	LightRAG docker	`docker inspect lightrag`	`.State.Health.Status == "healthy"`
4	LightRAG HTTP	`curl /health`	JSON with `.status` field
5	HiveMind readable	`sqlite3 … SELECT 1`	exit 0
6	HiveMind growing	count vs baseline	count ≥ baseline
7	Outbox exists	`test -f mc-task-outcomes.jsonl`	file present
8	ZAKON PLAN drift	`zakon-plan-lint.sh *-plan.md`	WARN if any FAIL, not total fail
9	Dead daemons	`launchctl list`	count of exit≠0 in ALAI namespace < 5

#	Check	Tool	PASS condition
10	Cost tracker	`cost-tracker.js summary today`	Input tokens ≥ 0 (non-error)

Runtime target: < 2 minutes. Measured: **9 seconds** on an idle machine.

Reading the output

- `PASS` (green) — check cleared
- `FAIL` (red) — check did not clear; suite will exit 1
- `WARN` (yellow) — degraded but not blocking (e.g. legacy plans failing linter)
- `BOOTSTRAPPED` — baseline file did not exist, first run recorded it

Summary line at end:

```
FAILED=X, WARNINGS=Y, PASS=Z, TOTAL=10
```

Exit codes:

- `0` — all critical checks passed (warnings allowed)
- `1` — at least one check failed

Usage

Manual run

```
bash ~/system/tools/system-regression.sh
# exit code visible in $?
```

Scheduled (weekly via launchd)

Create `~/Library/LaunchAgents/com.alai.system-regression.plist`:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE plist PUBLIC "-//Apple//DTD PLIST 1.0//EN" "http://www.apple.com/DTDs/PropertyList-1.0.dtd">
<plist version="1.0">
<dict>
    <key>Label</key><string>com.alai.system-regression</string>
```

```
    <key>ProgramArguments</key>
    <array>
        <string>/bin/bash</string>
        <string>/Users/makinja/system/tools/system-regression.sh</string>
    </array>
    <key>StartCalendarInterval</key>
    <dict>
        <key>Weekday</key><integer>1</integer>
        <key>Hour</key><integer>6</integer>
        <key>Minute</key><integer>0</integer>
    </dict>
    <key>StandardOutPath</key><string>/Users/makinja/system/logs/system-
regression.log</string>
    <key>StandardErrorPath</key><string>/Users/makinja/system/logs/system-
regression.err</string>
</dict>
</plist>
```

Load with:

```
launchctl load ~/Library/LaunchAgents/com.alai.system-regression.plist
launchctl start com.alai.system-regression
```

Schedule: every Monday 06:00 CEST. Output tailed to `~/system/logs/system-regression.log`.

Baseline management

Checks 6 (HiveMind growth) stores `/tmp/regression-baseline-hivemind.txt`. If `/tmp` gets cleared (reboot), the next run re-bootstraps. This is intentional — growth check is soft-enforced, warns on first run, tracks from second onwards.

If you want a durable baseline, move the file:

```
mkdir -p ~/system/state
mv /tmp/regression-baseline-hivemind.txt ~/system/state/
# then edit the script: BASELINE_FILE="$HOME/system/state/regression-baseline-hivemind.txt"
```

Adding a new check

1. Open `~/system/tools/system-regression.sh`
2. Find the `# --- Check N ---` pattern
3. Copy a block, increment counter, write the check
4. Update `TOTAL` at bottom
5. Re-run; confirm new check shows up in output
6. Commit + update this runbook

Keep checks **fast** (< 2s each) and **independent** (no check should depend on another passing).

Known current failures (2026-04-16)

These fail but are tracked as pre-existing debt, not regressions:

- Check 3 + 4: LightRAG probe timeout under heavy ingest load (87K files batch processing). Workaround: increase timeout to 30s in Dockerfile or switch to simpler probe.
- Check 9: 43 dead daemons in ALAI namespace — the actual list of broken daemons (b2-offsite-backup, bookstack-sync, learning-agent, model-warmup, forge-watchdog, ollama-warmup, tldr-briefing, autowork, health-monitor, apply-knowledge, tool-sync-audit, critical-tools-healthcheck). Needs FlowForge/Kelsey sweep.

Related

- Plan: `~/system/specs/system-evolution-plan.md` Phase 3 Task 13
- Main doc: `~/system/docs/system-evolution-2026-04-16.md`
- Evidence: `~/system/evidence/system-evolution-2026-04-16/v8-regression.txt`

Revision #3
Created 2026-04-16 21:44:13 UTC by John
Updated 2026-06-21 20:02:49 UTC by John