# Tool Manifest

> Last Verified: 2026-02-17 | Owner: John

# Tools Manifest

**CHECK THIS BEFORE CREATING NEW TOOLS.** If a tool exists, use it. If you create a new tool, add it here.

**TOOL-FIRST PROTOCOL:** `~/system/rules/tool-first-protocol.md` Redoslijed: Naši alati → Naši skillovi → Naša baza (HiveMind) → Internet → Ažuriraj bazu

**Last audit:** 2026-02-13 — Spring cleaning: 22 deprecated tools archived, 3 empty DBs deleted, 1 broken daemon unloaded, MEMORY.md trimmed 229→184 lines.

# Task Management

| Tool | Command | Description |
|---|---|---|
| task.sh | `~/system/tools/task.sh list\|add\|start\|done\|block` | Task CLI using Taskwarrior 3 (cross-session) |
| mc.js | `node ~/system/tools/mc.js list\|add\|start\|done\|show\|routes` | Mission Control - Task management with agent routing |
| mc.js routes | `node ~/system/tools/mc.js routes` | List available task routes (backend, frontend, devops, qa, bizdev, general) |
| mc.js add --route | `node ~/system/tools/mc.js add "Task" --route backend` | Create task with route - auto-spawns agent on start |

**Task → Agent Routing:** MC tasks can be tagged with routes that automatically spawn appropriate Ollama agents when task starts.

- Routes: backend (dev), frontend (designer+dev), devops (devops), qa (auditor), bizdev (marketer), general (dev)
- Agent output is captured and stored in task.agent_output field
- Visible in `mc.js show <id>` command
- If Ollama unavailable, gracefully degrades (logs error, doesn't block task)

- Agent runs in background via exec() - non-blocking
- Logs to HiveMind on spawn/completion/error

# Briefings & Analysis

| Tool | Command | Description |
|---|---|---|
| council-briefing.js | `node ~/system/tools/council-briefing.js` | AI Council: 4 personas (Growth, Revenue, Skeptic, Ops) analyze business data via Ollama. Posts to Slack #exec. Nightly at 22:00. |
| meeting-prep.js | `node ~/system/tools/meeting-prep.js [--ics file.ics] [--date YYYY-MM-DD]` | Calendar-aware meeting prep: ICS parsing, CRM attendee lookup, pipeline context, contextual notes. |
| council-briefing.js | `node ~/system/tools/council-briefing.js --model 70b` | Use 70b model for deeper analysis |
| council-briefing.js | `node ~/system/tools/council-briefing.js --dry-run` | Gather data only, no Ollama/Slack |
| john-morning.sh | `bash ~/system/tools/john-morning.sh` | Morning routine: Quran, tasks, HiveMind, health, daily synthesis. Daily at 07:00. |
| memory-synthesizer.js | `node ~/system/tools/memory-synthesizer.js daily [date]` | Summarize day's intel → HiveMind memo. Auto in morning-routine. |
| memory-synthesizer.js | `node ~/system/tools/memory-synthesizer.js weekly` | Synthesize week → HiveMind memo. Auto Sundays 23:00. |
| memory-synthesizer.js | `node ~/system/tools/memory-synthesizer.js promote` | Promote weekly → long-term knowledge |
| memory-synthesizer.js | `node ~/system/tools/memory-synthesizer.js prune` | Delete daily memos >30 days |
| memory-synthesizer.js | `node ~/system/tools/memory-synthesizer.js view [tier]` | View tiered memory (daily/weekly/longterm) |

# Meeting & Transcript Processing

| Tool | Command | Description |
|---|---|---|
| transcript-to-tasks.js | `node ~/system/tools/transcript-to-tasks.js <file>` | Extract action items from meeting transcript → MC tasks via Ollama |
| transcript-to-tasks.js | `node ~/system/tools/transcript-to-tasks.js <file> --preview` | Preview extracted actions (no task creation) |
| transcript-to-tasks.js | `node ~/system/tools/transcript-to-tasks.js <file> --owner john` | Assign all extracted tasks to owner |

**Formats:** .txt, .md, .srt, .vtt. Tasks prefixed with [TRANSCRIPT].

# Health & Quality

| Tool | Command | Description |
|---|---|---|
| md-health.js | `node ~/system/tools/md-health.js` | Markdown health scanner: broken links, TODOs, empty files, stale dates. Integrated in AgentForge. |
| md-health.js | `node ~/system/tools/md-health.js --json` | JSON output (for programmatic use) |
| md-health.js | `node ~/system/tools/md-health.js --fix-todos` | List all TODOs across codebase |
| md-health.js | `node ~/system/tools/md-health.js ~/path` | Scan specific path |
| doc-index.sh | `bash ~/system/tools/doc-index.sh [--output file.json] [--verbose]` | Document indexer — scans ~/projects, ~/ALAI, ~/companies for all markdown files. Creates JSON index with metadata (path, category, size, modified). Output: ~/system/databases/doc-index.json |
| doc-index.sh | `bash ~/system/tools/doc-index.sh --verbose` | Verbose mode — shows progress and breakdown by category |

# API Utilities

| Tool | Command | Description |
|---|---|---|
| api-fallback.js | `require('./api-fallback')` | Tiered API fallback + caching. `fetchWithFallback(key, tiers, opts)` tries each tier, caches result. |
| api-fallback.js | `node ~/system/tools/api-fallback.js cache-stats` | Show cache stats |
| api-fallback.js | `node ~/system/tools/api-fallback.js cache-clear` | Clear API cache |

**Cache:** `~/system/cache/api-fallback/` (file-based, per-key, TTL-aware)

# Usage Tracking

| Tool | Command | Description |
|---|---|---|

| | | |
|---|---|---|
| usage-tracker.js | `node ~/system/tools/usage-tracker.js log <agent> <model> <in> <out>` | Log AI call usage (auto-hooked in agent-runner.js + council-briefing.js) |
| usage-tracker.js | `node ~/system/tools/usage-tracker.js stats` | Usage summary (today, month, all-time) |
| usage-tracker.js | `node ~/system/tools/usage-tracker.js stats --agent <name>` | Per-agent breakdown |
| usage-tracker.js | `node ~/system/tools/usage-tracker.js stats --month` | Daily breakdown this month |
| usage-tracker.js | `node ~/system/tools/usage-tracker.js top` | Top agents by cost |
| usage-tracker.js | `node ~/system/tools/usage-tracker.js recent [limit]` | Recent calls |

**DB:** `~/system/db/usage.db` (SQLite). Auto-logged from agent-runner.js (Ollama) and council-briefing.js.

# Session Tracking

| Tool | Command | Description |
|---|---|---|
| session-ledger.sh | Auto (Stop/PreCompact hook) | Deterministic session extraction (files, commands, topics, errors, git) |
| session-search.sh | `bash ~/system/tools/session-search.sh topic\|file\|task\|keyword\|errors\|recent` | Search sessions |
| daily-consolidate.sh | `bash ~/system/tools/daily-consolidate.sh [YYYY-MM-DD]` | Consolidate day's sessions into daily log |
| weekly-digest.sh | `bash ~/system/tools/weekly-digest.sh [YYYY-MM-DD]` | Generate weekly summary |

**Session files:** `~/system/memory/sessions/YYYY-MM-DD-HHMM-sessionid.md`

# Memory

| Tool | Command | Description |
|---|---|---|
| hivemind.js | `node ~/system/agents/hivemind/hivemind.js read [agent] [limit]` | Read shared intelligence (replaces memory-lookup.js) |
| hivemind.js | `node ~/system/agents/hivemind/hivemind.js post <agent> <type> <msg>` | Post intel |

| Tool | Command | Description |
|---|---|---|
| hivemind.js | `node ~/system/agents/hivemind/hivemind.js query <search>` | Search intel |
| hivemind.js | `node ~/system/agents/hivemind/hivemind.js memo save\|get\|search\|list` | Key-value memory store |
| memory-indexer.py | `python ~/system/tools/memory-indexer.py` | Index memory for search |

# Communication

| Tool | Command | Description |
|---|---|---|
| slack.js | `node ~/system/tools/slack.js send <channel> "msg"` | Send message to Slack channel |
| slack.js | `node ~/system/tools/slack.js read <channel> [limit]` | Read recent messages from channel |
| slack.js | `node ~/system/tools/slack.js channels` | List all Slack channels |
| slack.js | `node ~/system/tools/slack.js create-channel <name>` | Create new channel |
| slack.js | `node ~/system/tools/slack.js unread` | Check unread messages |
| slack.js | `node ~/system/tools/slack.js users` | List workspace users |
| slack.js | `node ~/system/tools/slack.js status` | Check Slack connection |
| slack-bot.js | `node ~/system/tools/slack-bot.js` | Slack bot daemon — Claude Haiku via CLI (Socket Mode). AI backend: API → CLI → Ollama |
| slack-bot.js | `node ~/system/tools/slack-bot.js --test` | Test AI backend connection |
| email-to-task.js | `node ~/system/tools/email-to-task.js --from "x" --subject "y" --message-id "z" --class ACTION [--priority high]` | Auto-create MC tasks from ACTION emails with deduplication |
| email-to-task.js | `node ~/system/tools/email-to-task.js --status` | Show email classification stats |
| email-inbox.js | `node ~/system/tools/email-inbox.js status` | SQLite-backed email inbox — per-account stats (john, info, alai) |
| email-inbox.js | `node ~/system/tools/email-inbox.js pending` | List unanswered ACTION emails |
| email-inbox.js | `node ~/system/tools/email-inbox.js search "keyword"` | Full-text search in subject/from/sender name |
| email-inbox.js | `node ~/system/tools/email-inbox.js mark <id> responded\|archived\|read\|ignored` | Update email status |

| Tool | Command | Description |
|---|---|---|
| email-inbox.js | `node ~/system/tools/email-inbox.js stale [hours]` | Show emails unanswered > N hours (default 48) |
| email-inbox.js | `node ~/system/tools/email-inbox.js insert --message-id "x" --account john --from-addr "x" --subject "x" --classification ACTION --priority high` | Insert email into inbox DB |

| **MCP email** | `mcp__email__emails_find` | Search emails (sender, subject, date, folder). Account: "john" or "info" | | **MCP email** | `mcp__email__email_send` | Send emails (to, subject, body, HTML, attachments) | | **MCP email** | `mcp__email__email_respond` | Reply/forward with proper threading | | **MCP email** | `mcp__email__emails_modify` | Mark read/unread, flag, archive, move | | **MCP email** | `mcp__email__folders_list` | List all email folders |

**EMAIL PRAVILO:** SVE email operacije koriste **MCP email tools** (custom: email-mcp-bridge.js).

- Dva accounta: john@basicconsulting.no (account="john"), info@basicconsulting.no (account="info")
- Server: `~/system/tools/email-mcp-bridge.js` (ImapFlow + Nodemailer, wraps our proven stack)
- Konfigurisano u ~/.claude/mcp.json mcpServers.email
- Credentials: `~/system/config/mail-credentials.json` + `mail-credentials-info.json`

**Slack:** alai-talk.slack.com (channels: ops, development, client-support, exec)

# Password Sharing & Credential Management

| Tool | Command | Description |
|---|---|---|
| password-share.js | `node ~/system/tools/password-share.js create\|retrieve\|list\|cleanup\|audit` | Secure one-time password sharing with clients |
| client-vault.js | `node ~/system/tools/client-vault.js init\|add\|list\|get\|rotate\|check-rotation` | Per-client encrypted credential storage |

# Agent Infrastructure

| Tool | Command | Description |
|---|---|---|

| | | |
|---|---|---|
| agent-reporter.js | `node ~/system/tools/agent-reporter.js --task <id> --agent <name> --status <status> --summary <text>` | Structured agent output — validates against schema, stores in mission-control.db, emits events, posts to HiveMind |
| agent-reporter.js | `node ~/system/tools/agent-reporter.js --help` | Show usage and examples |
| agent-reporter.js | `node ~/system/tools/agent-reporter.js --task 937 --agent B1 --status completed --summary "..." --deliverables '[...]'` | Full structured report with deliverables, metrics, evidence |
| schema-validator.py | PostToolUse hook on TaskUpdate | Validates agent output JSON against agent-output-schema.json, logs violations to /tmp/schema-violations.log (warning-only, never blocks) |
| goal-verifier.js | `node ~/system/tools/goal-verifier.js --task <id>` | Automated goal verification — reads goal-schema.json, runs verification commands, updates statuses, stores in goals.db, emits events |
| goal-verifier.js | `node ~/system/tools/goal-verifier.js --help` | Show usage, goal types, and operators |
| goal-verifier.js | `node ~/system/tools/goal-verifier.js --task 937 --verbose` | Run verification with detailed output per goal |
| goal-verifier.js | `node ~/system/tools/goal-verifier.js --task 937 --dry-run` | Preview what would be verified without running commands |
| agent-worker.js | `node ~/system/tools/agent-worker.js` | Autonomous agent worker — polls MC every 5min, picks safe tasks, spawns Claude Code subagents, reports results |
| agent-worker.js | `node ~/system/tools/agent-worker.js --once` | Run single cycle then exit |
| agent-worker.js | `node ~/system/tools/agent-worker.js --dry-run` | Show next task without executing |
| agent-worker.js | `node ~/system/tools/agent-worker.js --status` | Show worker status and config |
| agent-worker.js | `node ~/system/tools/agent-worker.js --stop` | Stop daemon gracefully |

**Agent Output Schema:** `~/system/specs/agent-output-schema.json` (JSON Schema draft-07) **DB Table:** `mission-control.db.agent_reports` (task_id, agent, status, summary, report_json) **Event:** `agent.report` emitted to event bus on report submission **Created:** 2026-02-15 (MC #937 Phase 1)

**Goal Schema:** `~/system/specs/goal-schema.json` (JSON Schema draft-07) **DB:** `~/system/databases/goals.db` (goals, goal_history tables) **Verification:** verification-gate.py enforces goal verification for H/M priority tasks (if goal-schema.json present) **Events:** `goal.verified`, `goal.failed` emitted to event bus **Created:** 2026-02-15 (MC #937 Phase 4)

# Subagents (~/.claude/agents/)

| Agent | Role | Description |
|---|---|---|
| builder.md | Build | Implements ONE task using GOTCHA, self-validates, reports via agent-reporter.js or TaskUpdate |
| validator.md | Verify | Read-only GOTCHA compliance check + acceptance criteria, reports via agent-reporter.js |

# Local AI (Ollama on Mac Studio M3 Ultra)

## 2 Tools — Executor + Orchestrator

| Tool | Command | Description |
|---|---|---|
| agent-runner.js | `node ~/system/tools/agent-runner.js <agent> --task "X"` | **Executor** — sends ONE task to Ollama with agent identity + state |
| agent-runner.js | `node ~/system/tools/agent-runner.js list` | List all agents with status |
| agent-scheduler.js | `node ~/system/kernel/agent-scheduler.js spawn <agent> <task>` | **Orchestrator** — forks agent-runner.js as child processes for parallel execution |
| team-coordinator.js | `node ~/system/kernel/team-coordinator.js assign\|execute\|status\|message\|sync` | **Team Orchestrator** — multi-team coordination (Backend/Frontend/DevOps/QA) with cross-team messaging |

**Relationship:** agent-scheduler.js spawns agent-runner.js. Runner = single agent. Scheduler = multi-agent. team-coordinator.js uses scheduler for team execution. **What agents do:** Generate text responses via Ollama. They don't execute anything. **State:** `~/system/agents/state/*.json` (persists between runs) **Identities:** `~/system/agents/identities/*.md` (15 agents)

| offline-mode.js | `node ~/system/tools/offline-mode.js status` | **Offline Mode** — check Ollama readiness for Claude fallback | | offline-mode.js | `node ~/system/tools/offline-mode.js run "task"` | Route task to best local model (auto-detects type) | | offline-mode.js | `node ~/system/tools/offline-mode.js run "task" --agent dev` | Use specific agent identity | | offline-mode.js | `node ~/system/tools/offline-mode.js run "task" --text-only` | Text-only mode (no tool execution) | | offline-mode.js | `node ~/system/tools/offline-mode.js queue` | Show outputs waiting for Claude review | | offline-mode.js | `node ~/system/tools/offline-mode.js capabilities` | What local models

can/can't do | | offline-mode.js | `node ~/system/tools/offline-mode.js batch tasks.txt` | Run tasks from file (one per line) | | offline-mode.js | `node ~/system/tools/offline-mode.js enable\|disable` | Toggle offline mode on/off | | offline-mode.js | `node ~/system/tools/offline-mode.js whitelist` | Show safe read-only commands allowed offline | | offline-mode.js | `node ~/system/tools/offline-mode.js check "command"` | Check if command is whitelisted for offline use |

**Offline Mode:** When Claude API hits usage limits, switch to local Ollama models. Auto-routes tasks to best model (qwen-coder for code, 70b for reasoning, 8b for trivial). All outputs saved to `~/system/offline-queue/` with NEEDS_REVIEW status. Claude reviews when back online. Capability matrix built in — knows what local models can/can't do. Created 2026-02-12.

# Tier Routing (CC Rate Limit Optimization)

| Tool | Command | Description |
|---|---|---|
| ollama-engine.js | `require('./ollama-engine')` | **Centralized Ollama API** — generate(), classify(), healthCheck(). Consolidates duplicated Ollama HTTP code from 5+ files. |
| ollama-engine.js | `node ~/system/tools/ollama-engine.js test` | Run health check + generate test |
| tier-router.js | `require('./tier-router')` | **Central AI Router** — classify(caller, task) → {tier, engine, model}. Routes tasks to Ollama (free) or CC based on complexity. |
| tier-router.js | `node ~/system/tools/tier-router.js test` | Run routing tests |
| tier-router.js | `node ~/system/tools/tier-router.js classify <caller> <task>` | Test classification for caller+task |
| tier-router.js | `node ~/system/tools/tier-router.js stats` | Show routing stats (ollama vs cc) |
| ollama-tool-agent.js | `node ~/system/tools/ollama-tool-agent.js --task "X" --model Y` | **Ollama + Tools** — multi-turn agent with read-only tools (read_file, glob, grep, list_dir, run_cmd). Replaces CC for explore/validate tasks. |
| ollama-tool-agent.js | `node ~/system/tools/ollama-tool-agent.js --task "X" --verbose` | Verbose mode (show tool calls) |

**Tier Routing Architecture:**

- **Tier 1** (Ollama 8b): classify, filter, extract, triage
- **Tier 2** (Ollama 72b): summarize, draft, analyze, research, review
- **Tier 2c** (Ollama coder:32b): code review, debug, simple fix
- **Tier 3** (CC Sonnet): multi-file coding, architecture
- **Tier 4** (CC Opus): interactive sessions only
- **Config:** `~/system/config/tier-routing.json` (caller→tier mapping, keywords, fallback)
- **Integration:** agent-worker.js routes tasks through tier-router before execution

- **Fallback:** Ollama failure → auto-escalate to CC
- **Created:** 2026-02-16

# Models

| Model | Size | Use For |
|---|---|---|
| qwen2.5-coder:32b | 19GB | Coding, debugging, refactoring |
| llama3.1:70b | 40GB | Research, writing, analysis |
| llama3.1:8b | 5GB | Fast validation, simple queries |

# Routing & Decision

| Tool | Command | Description |
|---|---|---|
| route.js | `node ~/system/tools/route.js project <name>` | Lookup project (internal/external) |
| route.js | `node ~/system/tools/route.js query "<request>"` | Match request to company by routes |
| route.js | `node ~/system/tools/route.js list` | List all projects and companies |
| route.js | `node ~/system/tools/route.js add <name> <type>` | Add project to registry |

**Registry:** `~/system/databases/projects.json`

# Event Bus

| Tool | Command | Description |
|---|---|---|
| event-bus.js | `node ~/system/tools/event-bus.js emit <type> <json> [--publisher X]` | SQLite event bus — async emit/subscribe/dispatch. Decouples tools from point-to-point execSync. |
| event-bus.js | `node ~/system/tools/event-bus.js list [--type X] [--status X] [--limit N]` | List events (supports * wildcard for type) |
| event-bus.js | `node ~/system/tools/event-bus.js show <id>` | Show event details with payload |
| event-bus.js | `node ~/system/tools/event-bus.js replay <id>` | Re-process a failed/completed event |
| event-bus.js | `node ~/system/tools/event-bus.js dead-letter list\|resolve\|replay` | Dead letter queue management |
| event-bus.js | `node ~/system/tools/event-bus.js stats` | Event bus statistics (counts, last 24h by type) |

| Tool | Command | Description |
|---|---|---|
| event-bus.js | `node ~/system/tools/event-bus.js subscriptions list\|register\|seed` | Manage handler subscriptions |
| event-bus.js | `node ~/system/tools/event-bus.js dispatch [--once] [--interval N]` | Start dispatch loop (default 2s) |
| event-handlers.js | `require('./event-handlers.js')` | All subscriber handlers — task, lead, invoice, draft, email, job events |

**Event Bus Architecture (Transactional Outbox Pattern):**

- **Domain tools** (mc.js, sales-pipeline.js, invoice-generator.js, drafts.js) write events to **outbox table** in their own domain DB — same transaction as domain data. Atomic: if domain write succeeds, event is guaranteed.
- **Daemon tools** (email-agent.js, job-hunter-agent.js) use direct `bus.emit()` — no domain DB, fire-and-forget.
- **Dispatcher** daemon (event-dispatcher.js, 2s poll):
  1. **Relay:** reads outbox tables from 4 domain DBs → inserts into events.db → marks outbox processed
  2. **Dispatch:** claims pending events from events.db → calls registered handlers
- **Handlers** in event-handlers.js process events (Slack, HiveMind, Planka, leads, MC tasks, etc.)
- **Retry:** 3 attempts with backoff (0s → 30s → 2min) → dead letter queue → Slack alert
- **DB:** `~/system/databases/events.db` (central store, separate from domain DBs)
- **Outbox tables:** mission-control.db, leads.db, invoices.db, drafts.db
- **Daemon:** com.john.event-dispatcher (KeepAlive=true)
- **13 event types:** task.status_changed, task.created, lead.created, lead.stage_changed, lead.lost, invoice.created, invoice.overdue, invoice.paid, draft.created, draft.auto_approved, email.action_required, job.scored_perfect, job.scored_good
- **Integrated tools:** mc.js, sales-pipeline.js, invoice-generator.js, drafts.js (outbox), email-agent.js, job-hunter-agent.js (direct emit)

# GOTCHA Core

| Tool | Command | Description |
|---|---|---|
| utils.js | `require('~/system/lib/utils')` | Shared utility library (log, file, path, time, validate) |
| sales-pipeline.js | `node ~/system/tools/sales-pipeline.js add\|list\|show\|advance\|stats\|forecast\|auto-actions` | Lead CRM — tracks leads from prospect to won/lost. Auto-actions: archive old leads (lost >30d), escalate stale proposals (>14d no activity) |

| Tool | Command | Description |
|---|---|---|
| outbound.js | `node ~/system/tools/outbound.js start\|list\|stats` | Cold outreach prospecting — 3-email sequence (Day 1 intro, Day 3 follow-up, Day 7 final). Creates lead (cold_email), drafts intro email (LOW risk), schedules Day 3+7 reminders. Tags leads with outbound-seq. |
| email-to-contact.js | `node ~/system/tools/email-to-contact.js backfill` | Auto-populate contacts.db from email classifications. Creates contacts, logs interactions, skips spam/own. |
| email-to-contact.js | `node ~/system/tools/email-to-contact.js stats` | CRM import statistics (auto-imported vs manual, interactions) |
| contacts.js | `node ~/system/tools/contacts.js add\|list\|show\|search\|update\|log\|tag\|stats` | Central contact database — all partners, clients, brokers, vendors |
| contacts.js | `node ~/system/tools/contacts.js export-n8n` | Export n8n-monitored emails for Known Contact workflow |
| contacts.js | `node ~/system/tools/contacts.js import-leads` | Import contacts from leads.db |
| unified-crm.js | `node ~/system/tools/unified-crm.js pipeline\|client\|search\|dashboard` | READ-ONLY integration layer across 5 databases (contacts, leads, invoices, tickets, MC tasks) |
| contract-manager.js | `node ~/system/tools/contract-manager.js add\|list\|show\|renew\|terminate\|renewal-check\|status` | Contract lifecycle management — tracks contract status (draft→sent→signed→active→expired→terminated), auto-renewal alerts, MC task creation, Slack notifications. DB: contracts.db. Types: NDA, DPA, contract, SLA, MSA. |
| contract-manager.js | `node ~/system/tools/contract-manager.js renewal-check [--dry-run]` | Check for contracts expiring within 30 days, create MC renewal tasks (auto-renew only), send Slack alerts to #ops |
| document-store.js | `node ~/system/tools/document-store.js store <client> <type> <file>` | Document storage & retention system — organizes business documents with retention policies. Standard path: ~/ALAI/clients/{client}/documents/{type}/. Types: contract (10y), nda (5y), invoice (5y), proposal (2y), dpa (10y), agreement (10y), signed (10y). DB: documents.db |
| document-store.js | `node ~/system/tools/document-store.js list [client] [--type TYPE]` | List documents with optional filters |
| document-store.js | `node ~/system/tools/document-store.js find <search>` | Search documents by client/filename/notes |
| document-store.js | `node ~/system/tools/document-store.js retention-check` | Flag documents past retention period (non-destructive) |

| Tool | Command | Description |
|---|---|---|
| document-store.js | `node ~/system/tools/document-store.js stats` | Storage statistics by type and client |
| send-signing-email.js | `node ~/system/tools/send-signing-email.js send\|send-single\|test\|check` | ALAI branded document signing — creates DocuSeal submission + sends ALAI branded email with embedded logo via SMTP. Standard for all contracts/NDAs/DPAs. Always test first with `test` command. |
| nda-generator.js | `node ~/system/tools/nda-generator.js create <email> --name "Name" --company "Company"` | NDA PDF generator + DocuSeal signing flow — generates ALAI-branded NDA PDF via Puppeteer, uploads to DocuSeal, creates submission, sends ALAI branded signing emails. Flags: --preview (local PDF only), --test (send to post@alai.no), --orgnr, --address, --phone, --project. |
| fiken.js | `node ~/system/tools/fiken.js status\|companies\|invoices\|contacts\|balances\|dashboard` | Fiken API v2 integration — invoices list/show/sync, contacts list/show/sync, bank balances, CEO dashboard data. Syncs to invoices.db + contacts.db. |
| invoice-generator.js | `node ~/system/tools/invoice-generator.js create\|list\|show\|pay\|pdf\|send\|remind\|check-overdue\|auto-remind\|dashboard\|stats` | Invoice CRUD with VAT, PDF/HTML generation, MCP email draft creation, auto-reminders (3 levels: friendly/firm/urgent), automatic escalation system (Day 7/14/30+) |
| invoice-generator.js | `node ~/system/tools/invoice-generator.js auto-remind [--dry-run]` | Automatic invoice reminder escalation — Day 7: friendly (LOW risk draft), Day 14: firm (LOW risk draft + Slack), Day 30+: HIGH MC task + URGENT Slack. Norwegian templates. |
| support-ticket.js | `node ~/system/tools/support-ticket.js create\|list\|show\|update\|assign\|comment\|stats` | Support ticket system with SLA tracking (P1-P4) |
| email-to-ticket.js | `node ~/system/tools/email-to-ticket.js --sender "email" --subject "subject" --body "body" --uid uid` | Email → ticket bridge — detects support emails, creates tickets, generates ACK drafts, Slack + HiveMind notifications |
| ticket-sla-checker.js | `node ~/system/tools/ticket-sla-checker.js` | SLA breach detector — monitors open tickets, escalates to Slack #ops, generates escalation drafts, HiveMind logs |
| ticket-resolve-notify.js | `node ~/system/tools/ticket-resolve-notify.js --ticket-id TKT-12345` | Resolution notifier — generates client resolution email draft, HiveMind log |

| Tool | Command | Description |
|---|---|---|
| team-coordinator.js | `node ~/system/tools/team-coordinator.js teams\|assign\|handoff\|block\|unblock\|sync\|status` | Cross-team orchestration |
| onboard-client.js | `node ~/system/tools/onboard-client.js new\|status\|list\|timeline\|undo` | One-command client onboarding — orchestrates project scaffold, sales pipeline, support, teams, routing, welcome email, pipeline events, HiveMind |
| expansion-dashboard.js | `node ~/system/tools/expansion-dashboard.js [--compact]` | Aggregate view: companies, pipeline, invoices, support, teams |
| proposal-gen.js | `node ~/system/tools/proposal-gen.js create\|edit\|pdf\|send\|list\|show\|approve\|reject` | Professional proposal generator — auto-populates from leads, generates PDF, sends via SMTP (3 templates: standard, landing-page, webapp) |
| pipeline-events.js | `node ~/system/tools/pipeline-events.js check-reminders` | Stage transition event handlers — auto-triggered by sales-pipeline.js on advance/lose, generates drafts (→ drafts.db), creates reminders (~/system/reminders/), logs to HiveMind, sends Slack notifications. Handlers: onQualified, onProposal, onNegotiating, onWon, onActive, onLost |
| follow-up.js | `node ~/system/tools/follow-up.js check [--auto]` | Follow-up reminder processor — scans ~/system/reminders/ for due reminders, generates language-aware follow-up drafts (NO/EN/BS), 3 escalation levels (day 3/7/14), Slack alert on day 14 |
| follow-up.js | `node ~/system/tools/follow-up.js list` | List all pending follow-up reminders with due dates and escalation levels |
| follow-up.js | `node ~/system/tools/follow-up.js add <lead_id> <type> <days>` | Manually create follow-up reminder (types: proposal, inquiry) |
| drafts.js | `node ~/system/tools/drafts.js list\|show\|approve\|reject\|send\|stats` | Draft approval workflow — 3-level risk classification (low/medium/high), content-based pattern matching, smart auto-approval |
| drafts.js | `node ~/system/tools/drafts.js process-auto [--dry-run]` | Auto-classify and process all pending drafts (LOW→approve+send, MEDIUM→approve+Slack+send, HIGH→manual) |
| drafts.js | `node ~/system/tools/drafts.js auto-approve [--type type1,type2]` | Auto-approve low-risk drafts (optional type filter) |
| drafts.js | `node ~/system/tools/drafts.js mark-sent <id> [--message-id mid]` | Mark draft as sent (updates linked invoice status) |

| Tool | Command | Description |
|---|---|---|
| drafts.js | `node ~/system/tools/drafts.js import` | Import JSON drafts from ~/system/drafts/ |
| intake-analyzer.js | `node ~/system/tools/intake-analyzer.js detect-lang "text"` | Language detection (NO/EN/BS) via character markers + word frequency |
| intake-analyzer.js | `node ~/system/tools/intake-analyzer.js analyze "text"` | Request analysis via Ollama — extracts category/scope/urgency, generates 3 pricing options from Vizu pricing.md |
| intake-analyzer.js (module) | `const { detectLanguage, analyzeInquiry, generateOptions } = require('./intake-analyzer')` | Module API for client intake pipeline |

**intake-analyzer.js:** Language detector (æøå→NO, ćčšžđ→BS, word frequency lists) + request analyzer (Ollama llama3.1:8b JSON extraction) + option generator (reads ~/ALAI/pipeline/Vizu/finance/pricing.md, maps category→packages, generates A/B/C options). Heuristic fallback when Ollama unavailable. Pure Node.js, no dependencies. Created: 2026-02-13 (MC #840).

**follow-up.js:** Automated follow-up reminder system. Proposal reminders: day 3 (gentle), day 7 (nudge), day 14 (final + Slack). General inquiry: day 5. Language-aware templates (NO/EN/BS) extracted from lead intake analysis. Idempotent processing (marks reminders as processed). Legacy reminder migration: infers missing escalation_level and lang fields from due date and lead notes. Wired into gotcha-health.sh (runs every 15 min). Reminder format: JSON files in ~/system/reminders/ with fields: id, lead_id, type, due_date, escalation_level, created_at, processed, lang. Created: 2026-02-13 (MC #840).

# Image Generation

| Tool | Command | Description |
|---|---|---|
| image-gen.js | `node ~/system/tools/image-gen.js --prompt "desc" --output path.png` | Generate image via Gemini (free) or Together.ai |
| image-gen.js | `node ~/system/tools/image-gen.js --setup gemini YOUR_KEY` | Save API key to config |
| image-gen.js | `node ~/system/tools/image-gen.js --prompt "desc" --count 4` | Generate multiple images |

**Providers:** Gemini (default, free, no CC), Together.ai (FLUX, free tier) **Keys:** `~/system/config/image-gen.json` or env vars `GEMINI_API_KEY`, `TOGETHER_API_KEY` **Get key:** https://aistudio.google.com/apikey (2 min, no credit card)

| brand-compositor.js | `node ~/system/tools/brand-compositor.js all` | Deterministic brand asset generator — resize/composite REAL logo (profile-pic.png) onto social banners, profiles, favicons. No AI generation. | | brand-compositor.js | `node ~/system/tools/brand-compositor.js`

`profile\|avatar\|banner-linkedin\|banner-twitter\|og-image\|favicon` | Generate specific asset type | | design-engine.js | `node ~/system/tools/design-engine.js render <template> --data '{}' --output path.png` | Puppeteer-based HTML/CSS template rendering engine — pixel-perfect typography with Inter font, retina quality | | design-engine.js | `node ~/system/tools/design-engine.js list` | List available templates |

**Brand Compositor:** Uses sharp (npm) for deterministic resize + composite. Same pixels every time. Source: `~/system/context/branding/alai/social/profile-pic.png`. Output: `~/system/context/branding/alai/social/`. Options: `--source <file>`, `--output <dir>`. **Design Engine:** Uses Puppeteer (headless Chrome) to render HTML templates with professional typography (kerning, ligatures, OpenType). Templates: linkedin-banner (1584x396), twitter-banner (1500x500), og-image (1200x630), profile-card (400x400), favicon (180x180). Uses {{mustache}} placeholders. Reuses browser for batch rendering. Module export: `require('./design-engine')`. Options: `--data '{"key":"value"}'`, `--output path.png`, `--scale 2`. **Created:** 2026-02-10

# Intel & News Aggregation

| Tool | Command | Description |
|---|---|---|
| intel-briefing.js | `node ~/system/tools/intel-briefing.js` | Full daily briefing — fetch RSS + HN, summarize via Ollama, deliver to Slack #exec + HiveMind |
| intel-briefing.js | `node ~/system/tools/intel-briefing.js --preview` | Preview briefing in terminal |
| intel-briefing.js | `node ~/system/tools/intel-briefing.js --fetch` | Fetch only — list items without summarization |
| intel-briefing.js | `node ~/system/tools/intel-briefing.js --hours 48` | Custom lookback period (default: 24h) |

**Sources (7):** Anthropic News, Anthropic Engineering, Claude Code Changelog, OpenAI News, TechCrunch AI, Simon Willison, Hacker News API **Summarization:** Ollama llama3.1:8b (local, $0 cost) **Delivery:** Slack #exec channel + HiveMind + ~/system/logs/intel-briefing-{date}.md **Daemon:** com.edita.intel-briefing (daily 7:00 AM) **MCP RSS:** @missionsquad/mcp-rss added to Edita MCP config for live RSS queries **Created:** 2026-02-11

# Tender Hunting & Public Procurement

| Tool | Command | Description |
|---|---|---|
| tender-hunter-agent.js | `node ~/system/daemons/tender-hunter-agent.js` | **Doffin (Norway)** — TED API scanner for Norwegian IT tenders. Analyzes via Ollama, scores company fit (ALAI), stores in tenders.db. NO Puppeteer, NO Finn.no, NO TheHub. |

| Tool | Command | Description |
|---|---|---|
| tender-hunter-agent.js | `node ~/system/daemons/tender-hunter-agent.js --briefing` | Generate briefing from tenders.db (HOT/WARM summary) |
| tender-hunter-agent.js | `node ~/system/daemons/tender-hunter-agent.js --dry-run --verbose` | Test mode with detailed logging |
| bih-tender-hunter.js | `node ~/system/daemons/bih-tender-hunter.js` | **BiH Tender Hunter** — TED API (primary) + ejn.gov.ba (secondary) scanner for BiH IT tenders. Analyzes via Ollama, scores company fit (SnowIT), stores in bih-tenders.db. |
| bih-tender-hunter.js | `node ~/system/daemons/bih-tender-hunter.js --briefing` | Generate briefing from bih-tenders.db |
| bih-tender-hunter.js | `node ~/system/daemons/bih-tender-hunter.js --pages 5` | Custom page count (default: 3) |
| bih-tender-hunter.js | `node ~/system/daemons/bih-tender-hunter.js --source ted\|ejn` | Filter by data source (default: all) |
| bih-tender-hunter.js | `node ~/system/daemons/bih-tender-hunter.js --help` | Show usage and options |

**Doffin Agent:**

- **Data Source:** TED API (buyer-country = "NOR")
- **Keywords:** Norwegian + English IT terms
- **Scoring:** 0-100 (75+ HOT, 55-74 WARM, <55 COLD) — remote, English, tech stack match, framework, team size bonuses; security clearance, on-site, Norwegian-only penalties
- **DB:** ~/system/databases/tenders.db (tenders + outbox tables)
- **Events:** tender.hot, tender.warm → event bus
- **Delivery:** Slack #exec
- **Daemon:** com.john.tender-hunter (30 min interval)
- **Created:** 2026-02-15

**BiH Agent:**

- **Data Sources:** Tier 1 (TED API buyer-country = "BIH"), Tier 2 (ejn.gov.ba — TODO: needs Puppeteer)
- **Keywords:** Bosnian + English IT terms (digitalizacija, e-usluge, softver, etc.)
- **Scoring:** 0-100 (75+ HOT, 55-74 WARM, <55 COLD) — BiH-specific bonuses: digitalizacija (+15), transport/railway sector (+10), BAM currency (+10)
- **DB:** ~/system/databases/bih-tenders.db (tenders + outbox tables with source field: 'ted' or 'ejn')
- **Events:** tender.hot, tender.warm → event bus
- **Delivery:** Email reports (primary) + Slack #exec (fallback)
- **Daemons:** com.snowit.bih-tender-hunter (30 min), com.snowit.bih-tender-briefing (daily 07:30)
- **Created:** 2026-02-16 (MC #1057)

# Reporting & Analytics

| Tool | Command | Description |
|---|---|---|
| auto-report.js | `node ~/system/tools/auto-report.js daily` | Daily brief — revenue, pipeline, tasks, decisions, alerts. Generates email draft in ~/system/drafts/ |
| auto-report.js | `node ~/system/tools/auto-report.js weekly` | Weekly report — revenue summary, pipeline progress, team performance, achievements. Email draft with ALAI branding |
| auto-report.js | `node ~/system/tools/auto-report.js preview` | Preview report in terminal without generating draft |
| client-status-update.js | `node ~/system/tools/client-status-update.js generate [--dry-run]` | Weekly client status updates — queries MC for completed tasks per project, matches to client contacts, generates ALAI-branded HTML email drafts (MEDIUM risk). LaunchAgent: Mondays 08:00. |
| client-status-update.js | `node ~/system/tools/client-status-update.js list` | Show recently generated status update drafts |

**Auto-Report Features:**

- Aggregates data from: invoice-generator, sales-pipeline, mc.js, support-ticket, decisions doc
- ALAI brand styling (dark #09090b, accent #00E5A0)
- Mobile-friendly HTML emails
- Text + HTML versions in JSON draft
- Daemon config: ~/system/daemons/auto-report-config.json
- Recipient: alembasic@gmail.com
- Schedule: Daily 7:00 AM, Weekly Monday 8:00 AM

# Dashboards

| Dashboard | URL | Description |
|---|---|---|
| Mission Control | http://localhost:3030 | Task management, sessions, active work |
| CEO Dashboard | http://localhost:3030/ceo | Executive metrics — revenue, pipeline, projects, decisions, alerts |
| Client Portal | http://localhost:3030/client?token=XXX | Client-facing project status — tasks, tickets, SLA. Token-authenticated. |

**CEO Dashboard Features:**

- Revenue Overview: MRR, outstanding invoices, 3-month trend, next due date
- Pipeline Funnel: Visual funnel from prospect to won (data from sales-pipeline.js)
- Active Projects: Kanban board (active/pending/stalled) from MC tasks
- Decisions Pending: GO/NO-GO decisions from ~/system/specs/alem-decisions-2026-02.md
- Alerts Panel: Overdue invoices, SLA breaches, stale tasks (>7 days)
- Upcoming Timeline: Next 14 days deadlines from MC tasks
- Dark theme (ALAI brand: #09090b background, #00E5A0 accent)
- Auto-refresh: 60 seconds
- Mobile responsive

**Client Portal Features:**

- Token auth: `POST /api/client/tokens` (localhost only) to generate tokens
- Summary: active tasks, completed count, open tickets, blocked items
- Task list: filtered by client project, shows priority/status
- Ticket list: from tickets.db, shows SLA compliance
- ALAI dark theme, auto-refresh 60s, mobile responsive
- Token management: create/list/revoke via localhost API

# Testing & Verification

| Tool | Command | Description |
|---|---|---|
| smoke-test.js | `node ~/system/tools/smoke-test.js` | Run all smoke tests (Docker, Slack, daemons, MC, HiveMind) |
| smoke-test.js | `node ~/system/tools/smoke-test.js report` | Run all + post report to Slack #ops |
| smoke-test.js | `node ~/system/tools/smoke-test.js slack\|docker\|daemons\|mc\|hivemind` | Test specific suite |
| smoke-test.js | `node ~/system/tools/smoke-test.js api <url>` | Test specific API endpoint |
| health-check.js | `node ~/system/tools/health-check.js` | Monitor all services (Docker, HTTP, system, daemons) with human/JSON output |
| health-check.js | `node ~/system/tools/health-check.js --quick` | HTTP endpoints only (fast check) |
| health-check.js | `node ~/system/tools/health-check.js --json` | JSON output for programmatic use |
| daemon-health.js | `node ~/system/tools/daemon-health.js` | Daemon heartbeat monitor — checks all com.john.* LaunchAgents, reports PID/exit/status, detects unloaded plists |
| daemon-health.js | `node ~/system/tools/daemon-health.js --quick` | Quick status only |

| Tool | Command | Description |
|---|---|---|
| daemon-health.js | `node ~/system/tools/daemon-health.js --json` | JSON output for dashboards |
| auto-fix.js | `node ~/system/tools/auto-fix.js <service> <issue>` | Automated service recovery (restart loop prevention: max 3/hour) |
| ops-watchdog.js | `node ~/system/daemons/ops-watchdog.js` | Master watchdog daemon — health checks every 120s, auto-recovery via auto-fix.js, Slack alerts, event bus integration. Config: ~/system/config/ops-watchdog.json |
| cold-start.sh | `bash ~/system/ops/cold-start.sh` | Bring entire system up from fresh boot — 5-layer startup (infra→docker→core→business→workers→enrichment), pre-flight checks, verification |
| planka-sync.js | `node ~/system/tools/planka-sync.js test\|status\|sync <mc-id>` | MC↔Planka bidirectional sync — auto-moves cards on mc.js start/done/pause/resume |
| **MCP playwright** | `mcp__playwright__*` (nativni Claude toolovi) | Browser automation — navigate, click, fill, screenshot |

**Reports:** `~/system/reports/smoke-test-*.json` **Protocol:** Smoke test BEFORE + AFTER infra changes. Playwright for UI. `npm test` for code.

# Test Quality

| Tool | Command | Description |
|---|---|---|
| test-auditor.js | `node ~/system/tools/test-auditor.js <project-dir>` | Scan test suite for weak validation — detects "no crash" without rejection, missing stupid-user inputs, unused chaos strings |
| test-auditor.js | `node ~/system/tools/test-auditor.js <dir> --json` | JSON output for pipeline integration |

**Detects:** (1) Chaos tests with "no crash" but no rejection assertion, (2) Form fields missing stupid-user inputs (numbers in names, letters in phones), (3) CHAOS_STRINGS defined but unused. Exit: 0=clean, 1=findings. **Rule:** `~/system/rules/testing.md` (Mandatory Input Rejection Tests section)

# Plan Enforcement

| Tool | Command | Description |
|---|---|---|

| | | |
|---|---|---|
| plan-advance-step.js | `node ~/system/tools/plan-advance-step.js` | Manually advance to next plan step with gate checks (for builder agents) |
| plan-adherence-report.js | `node ~/system/tools/plan-adherence-report.js <task-id>` | Post-execution adherence report — did agent follow the plan? Shows step execution, violations, summary |

**Plan Enforcement Architecture:**

- **Hook:** `~/.claude/hooks/plan-enforcer.py` (PreToolUse) gates Write/Edit/Bash based on current plan step
- **Plan files:** `/tmp/plan-{task-id}.json` (machine-readable plan), `/tmp/plan-state-{task-id}.json` (execution state)
- **Audit log:** `/tmp/plan-audit-{task-id}.jsonl` (every hook decision logged)
- **Graceful degradation:** If no plan file exists, hook warns but allows (not all tasks have plans)
- **Manual step advance:** Builder calls plan-advance-step.js when ready to move forward
- **Validator check:** Validator runs plan-adherence-report.js to verify compliance
- **Created:** 2026-02-13 (MC #845)

# Build Pipeline

| Tool | Command | Description |
|---|---|---|
| build-project.js | `node ~/system/tools/build-project.js prep "Name" "type" "Description"` | Scaffold + CLAUDE.md + onboard + spec + task |
| build-project.js | `node ~/system/tools/build-project.js deploy "Name"` | Vercel deploy |
| build-project.js | `node ~/system/tools/build-project.js status "Name"` | Check project state |
| assert-log.sh | `source ~/system/tools/assert-log.sh` | Structured assertion library for deterministic verification (Phase 1) |
| gate-pre-claim.sh | `bash ~/system/tools/gate-pre-claim.sh --spec spec.json --workdir /path` | Pre-claim verification gate — file exists, hash changed, forbidden patterns (Phase 2) |
| gate-pre-claim.sh | `bash ~/system/tools/gate-pre-claim.sh --snapshot --workdir /path` | Snapshot file hashes before build |
| gate-pre-deploy.sh | `bash ~/system/tools/gate-pre-deploy.sh --project-dir /path` | Pre-deploy verification gate — tests, build, artifacts, TODO check (Phase 4) |

| pipeline-controller.js | `node ~/system/tools/pipeline-controller.js create\|status\|advance\|gate\|gate-pass\|abort\|resume\|history\|list\|dashboard` | Central pipeline orchestrator — tracks projects through 13 lifecycle phases (lead→support), automated gate checks, phase history, abort/resume. DB: pipeline.db | | pipeline-watchdog.js | `node ~/system/tools/pipeline-watchdog.js scan\|status [--auto-resume] [--notify]` | Detects stalled pipelines (2h threshold), orphan Claude team tasks (1h), stale MC tasks. Marks stalled, auto-

resumes, Slack alerts (2h cooldown). Skips aborted. | | rollback.js | `node ~/system/tools/rollback.js tag\|list\|rollback\|status <project>` | Git tag-based deployment rollback — tag deploys, list history, one-command rollback. Projects in ~/projects/. | | post-mortem.js | `node ~/system/tools/post-mortem.js generate\|create\|list\|show` | Incident post-mortem management — generate from ticket, create blank, list/show. Template: ~/system/template/post-mortem.md. Output: ~/system/reports/post-mortems/ |

**Types:** `landing-page` | `nextjs-app` | `api-backend` **Templates:** `~/system/template/types/<type>/CLAUDE.md` + `spec.md` **CI/CD:** `~/system/template/github-actions/ci.yml` (copied by scaffold.sh), `~/system/template/docker-compose.staging.yml` **Deploy:** `--platform vercel|railway|fly` (auto-detects from type if omitted) **Pipeline Gates:** Part of Zero-Hallucination Deterministic Build Pipeline

# Client Interaction & Design Review

| Tool | Command | Description |
|---|---|---|
| preview-share.js | `node ~/system/tools/preview-share.js start\|stop\|status\|list` | Client preview sharing — starts local dev server + Cloudflare tunnel for public URL. Auto-detects build output dirs. |
| design-approval.js | `node ~/system/tools/design-approval.js create\|list\|approve\|reject\|show\|stats` | Design review workflow — tracks design approval from draft→sent→reviewing→approved/rejected→implemented. DB: design-reviews.db |
| design-board.js | `node ~/system/tools/design-board.js create\|list\|stop\|restart` | Client-facing design review board — ALAI-branded web page with design options, feedback form, approve/reject. Cloudflare tunnel (http2 protocol) for public URL. Health check endpoint. Integrates with design-reviews.db. |
| client-signoff.js | `node ~/system/tools/client-signoff.js create\|status\|checklist\|check\|request-signoff\|complete\|list` | UAT + client sign-off — full acceptance testing workflow with per-type checklists, client approval gate, delivery tracking. DB: design-reviews.db |

**UAT Template:** `~/system/template/uat-checklist.md` (per project type: webapp, landing-page, api-backend) **DB:** `~/system/databases/design-reviews.db` (reviews + signoffs tables)

# File Editing

| Tool | Command | Description |
|---|---|---|

| | | |
|---|---|---|
| smart-edit.js | `node ~/system/tools/smart-edit.js view <file> [start-end]` | Show file lines with line numbers |
| smart-edit.js | `node ~/system/tools/smart-edit.js replace <file> <start-end> <content>` | Replace line range with new content |
| smart-edit.js | `node ~/system/tools/smart-edit.js insert <file> <after> <content>` | Insert content after line number |
| smart-edit.js | `node ~/system/tools/smart-edit.js delete <file> <start-end>` | Delete line range |
| smart-edit.js | `node ~/system/tools/smart-edit.js append <file> <content>` | Append content to end of file |

**Why:** Line-number based editing is more reliable than str_replace (exact match failures). Inspired by The Harness Problem. Reduces edit fail rate from ~15-20% to ~5%. **Backup:** Auto-creates `.bak` before each edit. Use `--no-backup` to skip. **Stdin:** Use `-` as content arg to pipe content via stdin (for multi-line edits). **Lines:** 1-indexed, inclusive ranges (10-15 = lines 10 through 15). **Workflow:** `view` to see lines → `replace`/`insert`/`delete` by line number.

# Daemons (LaunchAgents)

| Daemon | Interval | Description |
|---|---|---|
| com.john.slack-bot | always | Slack bot — Claude Haiku via Socket Mode. AI: API → CLI → Ollama. Needs SLACK_BOT_TOKEN + SLACK_APP_TOKEN |
| com.john.mc-dashboard | always | Mission Control web dashboard (port 3030) — includes CEO Dashboard at /ceo route |
| com.john.mc-session-worker | on session events | Session state extraction |
| com.john.pipeline-watcher | 60 sec | Pipeline event dispatcher + invoice auto-reminder daemon — checks unsigned proposals, triggers invoice escalation (Day 7/14/30+ reminders) |
| com.john.event-dispatcher | always | Event bus dispatcher daemon — polls events.db every 2s, routes to handlers, retry with backoff, dead letter queue |
| com.john.ops-watchdog | always | Master watchdog — health checks every 120s, auto-recovery, Slack alerts, event bus. Config: ~/system/config/ops-watchdog.json |
| com.john.client-status-update | Monday 08:00 | Weekly client status update generator — queries MC for completed tasks, generates ALAI-branded email drafts per project |

**Ops Documentation:** `~/system/ops/` — service catalog, dependency map, 15 runbooks, cold-start script, ops README. **Ops Dashboard:** http://localhost:3030/ops (status page), /api/ops/health (JSON), /api/ops/history (events)

**Env Vars (both profiles):**

- `enableToolSearch=true` — lazy-load MCP tools
- `CLAUDE_CODE_EXPERIMENTAL_AGENT_TEAMS=true` — agent teams
- `DISABLE_AUTOUPDATER=1` — prevent auto-update breaking custom setup
- `CLAUDE_CODE_DISABLE_AUTO_COMPACT=true` — manual compaction control

# Boards (Planka — Kanban)

| Tool | URL | Description |
|---|---|---|
| Planka | https://boards.basicconsulting.no | Kanban boards per project (Trello-like) |
| Planka local | http://localhost:3100 | Direct local access |

**Admin:** john / BasicAS2026! **User:** alem / Alem2026! **Password reset:** `node ~/system/tools/planka-admin.js reset-password <username> <new-pass>` **Add user:** `node ~/system/tools/planka-admin.js add-user <email> <username> <name> <pass>` **SMTP:** Configured (send.one.com:465, john@basicconsulting.no) — za notifikacije **Docker:** ~/system/services/planka/docker-compose.yml **Projects:** Wizard NUF, Ren Drom, Riad Basic, Drop Fintech, ALAI Internal, BasicAS Operations **Tunnel:** Cloudflare (boards.basicconsulting.no → localhost:3100)

# Setup & Backup

| Tool | Command | Description |
|---|---|---|
| syslog.sh | `bash ~/system/tools/syslog.sh add "opis"` | System Changelog — logira promjene za oba agenta |
| syslog.sh | `bash ~/system/tools/syslog.sh today` | Današnje changelog entries |
| syslog.sh | `bash ~/system/tools/syslog.sh recent [N]` | Zadnjih N entries |
| setup-backup.sh | `bash ~/system/tools/setup-backup.sh "opis"` | Backup setup files + changelog |
| sync-to-mini.sh | `bash ~/system/tools/sync-to-mini.sh [--execute]` | Sync GOTCHA to Mac Mini |
| daemon-manager.js | `node ~/system/daemons/daemon-manager.js list\|start\|stop\|status` | Manage persistent background services |

| Tool | Command | Description |
|---|---|---|
| team-cleanup.sh | `bash ~/system/tools/team-cleanup.sh [--force] [--days N]` | Clean stale Agent Teams task/team dirs (default 7d) |

# Company Management

| Tool | Command | Description |
|---|---|---|
| company.sh | `~/system/tools/company.sh list\|info\|add` | Company registry management |

# Skills (Claude Code Slash Commands)

| Command | Description |
|---|---|
| `/plan-with-team` | Creates plan with builder/validator teams |
| `/build-plan` | Executes approved plan using TaskList |
| `/code-review` | Systematic GOTCHA code review (security, quality, performance) |
| `/debugging` | Systematic bug investigation and resolution |
| `/security-audit` | OWASP Top 10 + config + infra security review |
| `/design-system` | AI-powered design generator — multi-tool (v0.dev, Google Stitch, Figma Make, Codia AI). Prompt templates per tool. Brief → kickass design + code. |
| `/figma-design` | Figma WebSocket bridge operations — populate design systems, create screens programmatically |

**Workflow:** `/plan-with-team "task"` → plan → approval → `/build-plan` → execution **Design:** `/design-system "brief"` → AI tool selection → optimized prompts → Figma + code **Review:** `/code-review <file>` or `/security-audit <target>` **Debug:** `/debugging "<bug description>"`

# Vector & Semantic Search

| Tool | Command | Description |
|---|---|---|
| vector-db.js | `node ~/system/tools/vector-db.js help` | Hybrid Vector DB: SQLite + vector columns for semantic search. Reusable module. |

| Tool | Command | Description |
|---|---|---|
| vector-db.js (module) | `const { VectorDB } = require('./vector-db')` | Module API: createCollection(), insert(), search(), hybridSearch(), bulkInsert() |
| vector-db.js search | `node ~/system/tools/vector-db.js search <db> <collection> <query>` | Semantic search via Ollama nomic-embed-text (768-dim) |
| vector-db.js hybrid | `node ~/system/tools/vector-db.js hybrid <db> <col> <query> --where "cond"` | SQL filter + vector ranking combined |
| knowledge-base.js | `node ~/system/tools/knowledge-base.js add <url-or-file> [--tag t]` | KB: drop URL/file → chunk → vector store. Semantic search over all docs. |
| knowledge-base.js | `node ~/system/tools/knowledge-base.js search <query> [--tag t]` | Semantic search across knowledge base documents |
| humanizer.js | `echo "text" \| node ~/system/tools/humanizer.js [--deep]` | Remove AI patterns from text. Quick (regex) or deep (Ollama rewrite). Module: require('./humanizer') |
| hourly-backup.sh | `bash ~/system/tools/hourly-backup.sh [--dry-run\|--list]` | Hourly auto-commit to 'auto-backup' branch across all repos. LaunchAgent: com.john.hourly-backup. |
| db-backup.sh | `bash ~/system/tools/db-backup.sh [--list\|--restore]` | Daily SQLite backup (14 DBs). sqlite3 .backup, tar.gz, 30-day rotation. LaunchAgent: com.john.db-backup (03:00). |
| cron-notify.sh | `bash ~/system/tools/cron-notify.sh "job" "OK\|ERROR" "details"` | Post cron results to Slack #ops channel. Used by db-backup, hourly-backup. |
| memory-indexer.py | `python3 ~/system/tools/memory-indexer.py index\|search` | LanceDB vector search over MD files (Python, sentence-transformers) |

**Vector Pattern:** Embeddings stored as BLOB (Float32Array) in SQLite. Cosine similarity computed in JS. Model: nomic-embed-text (768-dim, local Ollama). Batch embedding supported (32/batch). Usage tracked via usage-tracker.js.

# Databases (~/system/databases/)

| Database | Description |
|---|---|
| leads.db | Sales pipeline / Lead CRM — use `sales-pipeline.js` |
| invoices.db | Invoice tracking — use `invoice-generator.js` |
| contracts.db | Contract lifecycle management — use `contract-manager.js` |
| documents.db | Document storage & retention — use `document-store.js` |

| Database | Description |
|---|---|
| tickets.db | Support tickets with SLA — use `support-ticket.js` |
| teams.db | Cross-team coordination — use `team-coordinator.js` |
| strategy-tracker.db | Strategic goals |
| alem-directives.db | Alem's direct orders |
| projects.db | Project lifecycle (phases, milestones, metrics) |
| hivemind.db | Agent shared intelligence |
| drafts.db | Email draft approval workflow — use `drafts.js` |
| events.db | Event bus store — use `event-bus.js` |
| projects.json | Routing registry — use `route.js` |
| company-registry.json | Company information registry |

# Enforcement Hooks (~/.claude/hooks/)

| Hook | Matcher | Description |
|---|---|---|
| security-guard.py | `.*` (all tools) | Blocks forbidden paths, dangerous commands, delete protection, business-critical doc enforcement |
| agent-protocol-enforcer.py | `Task` | CORE PROTOCOL enforcement for subagent spawning |
| gotcha-enforcer.py | `Write\|Edit\|NotebookEdit\|Bash` | Boot flag + MC active task enforcement |
| gate-pre-commit.py | `Bash` | Pre-commit validation |
| hallucination-detector.py | `Write\|Edit` | Phantom tools, phantom paths, wrong ports, phantom require/import detection |
| teammate-quality-gate.py | `TeammateIdle` | Quality gate for agent teammates — checks TODO/FIXME markers, syntax errors in recent files. Exit 2 = keep working |

**Global:** All hooks apply to ALL agents (parent + subagents) via `~/.claude/settings.json`. **ZAKON #1:** AI bez enforcement-a ne radi. Hooks su deterministički enforcement.

# Design & Figma

| Tool | Command | Description |
|---|---|---|

| | | |
|---|---|---|
| figma-extract.js | `node ~/system/tools/figma-extract.js extract-tokens <file-key>` | Extract design tokens (colors, typography, effects) from Figma file |
| figma-extract.js | `node ~/system/tools/figma-extract.js extract-components <file-key>` | List components with metadata and variants |
| figma-extract.js | `node ~/system/tools/figma-extract.js frame-to-prompt <file-key> <node>` | Generate implementation prompt from Figma frame |
| figma-extract.js | `node ~/system/tools/figma-extract.js file-info <file-key>` | File metadata and pages |
| figma-to-react.js | `node ~/system/tools/figma-to-react.js <file-key> <node-id> --output Login.tsx` | **Figma → React + Tailwind** — generates production React TSX from Figma frame via REST API (Auto Layout→Flexbox, fills→bg, typography→text classes, shadows→shadow-*) |
| figma-to-react.js | `node ~/system/tools/figma-to-react.js <file-key> <node-id> --component Name` | Custom component name (default: derived from frame name) |
| figma-to-react.js | `node ~/system/tools/figma-to-react.js <file-key> <node-id>` | Output to stdout (pipe to file or preview) |
| figma-validate.js | `node ~/system/tools/figma-validate.js compare <file-key> <node-id> <url> --output /tmp/validate/` | Visual validation tool — compare built page vs Figma design via pixel diff. Exit: 0=PASS 1=FAIL 2=ERROR. Enforces ZAKON 0.1 |
| figma-validate.js | `node ~/system/tools/figma-validate.js compare ... --threshold 0.05 --viewport 1920x1080` | Custom threshold (default 0.1=10%) and viewport (default 375x812) |
| figma-token-sync.js | `node ~/system/tools/figma-token-sync.js <file-key> --output ./tokens/ --format all` | **Figma Variables → Design Tokens** — extracts Variables API → W3C DTCG JSON + Tailwind theme + CSS custom properties. Supports modes (light/dark). |
| figma-token-sync.js | `node ~/system/tools/figma-token-sync.js <file-key> --format tailwind --output ./tailwind-tokens.js` | Single format: tailwind, css, w3c, json, or all |
| figma-populate.js | `bun ~/system/tools/figma-populate.js <channel-id>` | Populate Figma with design tokens (colors, typography, spacing, radius, buttons) via WebSocket bridge |
| v0-generate.js | `node ~/system/tools/v0-generate.js generate "prompt"` | v0.dev Platform API wrapper — prompt → React+Tailwind code. Also generates optimized prompts for manual use. |
| v0-generate.js | `node ~/system/tools/v0-generate.js generate --brief Name --screen login --industry fintech --primary "#hex"` | Structured brief → optimized prompt |
| v0-generate.js | `node ~/system/tools/v0-generate.js prompt --brief Name --industry fintech` | Output prompt only (no API call) — for copy-paste into v0.dev or Google Stitch |

| | | |
|---|---|---|
| v0-generate.js | `node ~/system/tools/v0-generate.js setup <api-key>` | Save v0.dev API key |
| design-to-code.js | `node ~/system/tools/design-to-code.js assemble --stitch-code <html> --assets-dir <dir> --target-page <tsx>` | Assemble Stitch HTML + Figma assets → Next.js TSX. Converts HTML→JSX, inline styles→Tailwind, integrates assets, optional logic preservation. |
| design-to-code.js | `node ~/system/tools/design-to-code.js assemble ... --preserve-logic` | Extract and keep business logic (useState, handlers) from existing page |
| **MCP figma** | `mcp__figma__*` (native Claude tools) | Figma MCP integration — direct Figma access from Claude |

**Config:** `~/system/config/figma.json` or `FIGMA_TOKEN` env var **v0 Config:** `~/system/config/v0.json` or `V0_API_KEY` env var **File key:** From Figma URL — `figma.com/design/<FILE-KEY>/...` **Node ID:** From Figma URL (select frame, copy link) or use `figma-extract.js list-nodes <file-key>` **Figma bridge:** WebSocket on port 3055 (bun). Channel ID from Figma Desktop → Plugins → Claude MCP Plugin. **External AI tools:** v0.dev ($20/mo), Google Stitch (free: stitch.withgoogle.com), Figma Make (native), Codia AI (Figma plugin) **Design output:** `~/system/design-output/` **Created:** 2026-02-12 (figma-extract), 2026-02-13 (figma-populate, v0-generate, /design-system skill), 2026-02-14 (figma-to-react, figma-validate, figma-token-sync)

# Archived (NE POSTOJE — samo za referencu)

| Tool | Status | Note |
|---|---|---|
| ~~session-save.sh~~ | REMOVED (2026-02-07) | Orphaned code, never hooked, conflicts with session-ledger.sh |
| ~~memory-lookup.js~~ | REMOVED | Zamijenjeno HiveMind-om |
| ~~memory-search.js~~ | REMOVED | Zamijenjeno HiveMind-om |
| ~~mail.js~~ | NEVER EXISTED | Haluciniran |
| ~~mail-filter.js~~ | NEVER EXISTED | Haluciniran |
| ~~security.js~~ | NEVER EXISTED | Haluciniran — pravi enforcement = ~/.claude/hooks/ |
| ~~secure-config.js~~ | NEVER EXISTED | Haluciniran |
| ~~keychain-helper.js~~ | NEVER EXISTED | Haluciniran |
| ~~design-enforcer.js~~ | NEVER EXISTED | Haluciniran |
| ~~optimize-images.js~~ | NEVER EXISTED | Haluciniran |
| ~~strategy-tracker.js~~ | NEVER EXISTED | Haluciniran |

| Tool | Status | Note |
|---|---|---|
| ~~deploy-strategy-tracker.js~~ | NEVER EXISTED | Haluciniran |
| ~~prompt-tester.js~~ | NEVER EXISTED | Haluciniran |
| ~~self-improve.js~~ | NEVER EXISTED | Haluciniran |
| ~~send-to-edita.js~~ | NEVER EXISTED | Haluciniran |
| ~~generate-boot.js~~ | NEVER EXISTED | Haluciniran |
| ~~generate-today.js~~ | NEVER EXISTED | Haluciniran |
| ~~solution-finder.js~~ | NEVER EXISTED | Haluciniran |
| ~~docusign.js~~ | NEVER EXISTED | Haluciniran |
| ~~validator.js~~ | ARCHIVED (2026-02-06) | Was orphaned — see ~/system/archive/ |
| ~~laws-enforcer.js~~ | ARCHIVED (2026-02-06) | Was checker-only — see ~/system/archive/ |
| ~~email-smtp-imap-mcp~~ | DEPRECATED (2026-02-11) | Community MCP server — unreliable, replaced by custom email-mcp-bridge.js |
| ~~mcp-email-server (ai-zerolab)~~ | TESTED (2026-02-11) | Python MCP — ClosedResourceError bug, not used |

# brand-package.js

**Purpose:** Generate brand package (guidelines, colors, typography) for company factory pipeline
**Location:** `~/system/tools/brand-package.js`
**Usage:** `node ~/system/tools/brand-package.js "ProjectName" --logo /path/to/logo.png [--colors "primary:#hex,secondary:#hex"] [--output /path/]`
**Dependencies:** None (pure Node.js)
**Output:** Creates brand-guidelines.md, colors.json, typography.json
**Features:** Extracts colors from PNG logo, supports color overrides, generates complete brand identity
**Created:** 2026-02-09

---

Revision #3
Created 2026-02-17 22:14:01 UTC by John
Updated 2026-05-31 20:00:33 UTC by John