Blueprint Liveness Gate

Blueprint Liveness Gate

Owner: CodeCraft **Implemented:** 2026-04-16 (System Evolution Phase 2, MC #8026) **Files:** `~/system/tools/mc.js` (migration + add handler + done gate)

Purpose

Blueprints drift. `BUILD-BLUEPRINT.md` files describe architecture, stack, integrations — but they're written once and then ignored. Six months later the code has migrated (Hono → Kotlin, Vite → Next.js) and the blueprint still shows the old stack. New agents reading the blueprint get misled.

The **liveness gate** forces a blueprint to be touched during the task that changes it. If a task is tagged with `--blueprint-ref`, it cannot be marked `done` unless the referenced blueprint file has been modified since the task started.

How it works

```
mc.js add "Switch Plock to Next.js" --blueprint-ref ALAI/products/plock/BUILD-BLUEPRINT.md
  → stores blueprint_ref in tasks table
mc.js start <id>                    → records started_at
(developer works, updates code + blueprint)
mc.js ready <id> "tested X"
mc.js done <id> "completed"
  → gate reads task.blueprint_ref, fs.statSync(bp).mtimeMs
  → if mtime < started_at → REJECT with ZAKON error
  → if mtime ≥ started_at → PASS
```

Usage

Add a task bound to a blueprint:

```
node ~/system/tools/mc.js add "Add GraphQL to Drop" \
  --blueprint-ref ALAI/products/drop/BUILD-BLUEPRINT.md \
  --priority H
```

Check which tasks have blueprint refs:

```
sqlite3 ~/system/databases/mission-control.db \
  "SELECT id, title, blueprint_ref FROM tasks WHERE blueprint_ref IS NOT NULL ORDER BY id DESC
LIMIT 10"
```

Retrofit an existing task:

```
sqlite3 ~/system/databases/mission-control.db \
  "UPDATE tasks SET blueprint_ref='ALAI/products/plock/BUILD-BLUEPRINT.md' WHERE id=5126"
```

Emergency bypass

Use only when the blueprint genuinely shouldn't change (e.g. a typo fix, dependency bump that doesn't alter stated architecture).

```
node ~/system/tools/mc.js done <id> "outcome" --force "reason why blueprint unchanged"
```

The `--force` reason is logged to HiveMind for audit.

Adding a new blueprint

1. Create `<project>/BUILD-BLUEPRINT.md` with sections: Scope, Stack, Ownership, Change Protocol, Last Updated, Related MC Tasks
2. Path in `--blueprint-ref` is **relative to** `$HOME` (gate uses `path.resolve(os.homedir(), task.blueprint_ref)`)
3. Run `bash ~/system/tools/zakon-plan-lint.sh <file>` as a sanity check on the document structure (the linter accepts blueprints too)

Retrofit pattern (Plock example)

`~/ALAI/products/plock/BUILD-BLUEPRINT.md` got a `## 11. Stack Compliance` section appended:

```
## 11. Stack Compliance

| Layer    | Current              | CEO Standard          | Status              |
|----------|----------------------|-----------------------|---------------------|
| Backend  | Kotlin + Ktor        | Kotlin + Ktor         | COMPLIANT           |
| Frontend | Vite MFE             | Next.js 15            | MIGRATION — MC #5126|
| DB       | PostgreSQL + Flyway  | PostgreSQL + Flyway   | COMPLIANT           |
| Testing  | JUnit + Playwright   | JUnit + Playwright    | COMPLIANT           |
| Monorepo | Turborepo + pnpm     | Turborepo + pnpm      | COMPLIANT           |
```

Do not refactor the whole blueprint at once — one compliance table is enough to surface the drift.

Related

- Plan: `~/system/specs/system-evolution-plan.md` Phase 2 Task 11
- Companion gate: Proveo gate in `mc.js done` (see `mc-done-auto-writeback.md`)
- Companion linter: `zakon-plan-lint.sh` (see `zakon-plan-linter.md`)
- Felles shared-configs blueprint: `~/felles/shared-configs/BUILD-BLUEPRINT.md`

Revision #3
Created 2026-04-16 21:44:10 UTC by John
Updated 2026-06-21 20:02:49 UTC by John