Schema Stub Gate + Claim Schema Injector (MC #101065)

Schema Stub Gate + Claim Schema Injector (MC #101065)

MC: #101065 (Deterministic Session Compiler — expanded scope)

Parent: Reality Anchor Doctrine v1
Owner: CodeCraft / Petter Graff + FlowForge / Kelsey Hightower
Date Shipped: 2026-05-16
Components: `~/system/tools/schema-injector.js` + `~/.claude/hooks/schema-stub-gate.sh`

Problem Statement

The claim schema was never pre-registered at task dispatch boundary. When John dispatches UAT, no template exists specifying "expected logins: N, expected a11y violations threshold: T, expected commits: SHA list". The verifier has no baseline to fill — so it fills from John prose (the same LLM surface the system is meant to bypass). This is the root cause of evidence padding incidents (Bilko UAT 2026-05-16: "4/4 logins working" claimed unverified).

> **Petter Graff (unified fix doc):** "Gap today: compiler exists but does not pre-register expected claim schema before dispatch."

Solution: Pre-Dispatch Claim Schema Injection

The system now operates in three phases:

1. **mc.js start** → fires `schema-injector.js` → writes `/tmp/claim-schema-<mc_id>.json` with claim stubs
2. **Verifier/builder work** → runs deterministic probes → fills stubs from probe output JSON
3. **mc.js ready/done** → fires `schema-stub-gate.sh` → BLOCKS if any stub is PENDING or FAILED

Component 1: Schema Injector

File: `~/system/tools/schema-injector.js`
Trigger: Fires automatically at `mc.js start <id>` (line 2044 of mc.js)
Input: MC title + description + ACs
Output: `/tmp/claim-schema-<mc_id>.json`

Claim Detection (Deterministic Regex)

No LLM inference. Keywords in AC text map to claim_class via `~/system/probes/registry.json`:

AC Keyword	Mapped claim_class	Probe Script
login, auth, sign-in, credentials	`login_works`	`~/system/probes/login-probe.sh`
commit, SHA, git, code change	`commit_verified`	`~/system/probes/git-diff-probe.sh`
a11y, accessibility, WCAG, violations	`a11y_count`	`~/system/probes/playwright-a11y-probe.js`
test, spec, @Test, it(, describe(	`test_count`	`~/system/probes/test-enumeration.sh`
deploy, URL, HTTP 200, curl	`http_200`	(Phase 2 — not yet shipped)

Schema Structure

```
{
  "mc_id": 101065,
  "generated_at": "2026-05-16T14:32:10Z",
  "task_started_at": "2026-05-16T14:32:10Z",
  "git_baseline": {
    "repos": ["/Users/makinja/projects/bilko"],
    "baseline_shas": ["a3f8bc4", "d9e2f01"]
  },
  "claim_stubs": [
    {
      "claim_class": "login_works",
```

```
      "probe": "~/system/probes/login-probe.sh",
      "expected": { "login_count": null },
      "filled_at": null,
      "probe_output_path": null,
      "status": "PENDING"
    },
    {
      "claim_class": "a11y_count",
      "probe": "~/system/probes/playwright-a11y-probe.js",
      "expected": { "violations_critical": 0, "violations_serious": 2 },
      "filled_at": null,
      "probe_output_path": null,
      "status": "PENDING"
    }
  ],
  "block_if_stubs_null": true
}
```

Component 2: Verifier Fills Stubs

Protocol: At `mc.js ready` or `mc.js done` (before gate passes):

1. Read `/tmp/claim-schema-<mc_id>.json`
2. For each `PENDING` stub:
   - Run mapped probe script (e.g., `bash ~/system/probes/login-probe.sh --url ...`)
   - Capture structured JSON output → write to `/tmp/probe-output-<mc_id>-<claim_class>.json`
   - Fill stub fields (`filled_at`, `probe_output_path`)
   - Set `status` to `FILLED` or `FAILED`
3. Any stub remains `PENDING` or `FAILED` → task BLOCKED
4. Write filled schema to `/tmp/claim-schema-<mc_id>-filled.json`

Rule: Verifier may NOT fill stubs from prose or John output. Only probe JSON is accepted.

Component 3: Schema-Stub Gate Hook

File: `~/.claude/hooks/schema-stub-gate.sh`
Trigger: PreToolUse on `mc.js ready` and `mc.js done`

Exit Codes:

- `0` = Allow (all stubs filled or grace period)
- `1` = Block (pending/failed stubs or schema missing after grace period)

Grace Period

Until 2026-06-07: Missing schema → WARN only (allow)
After 2026-06-07: Missing schema → BLOCK

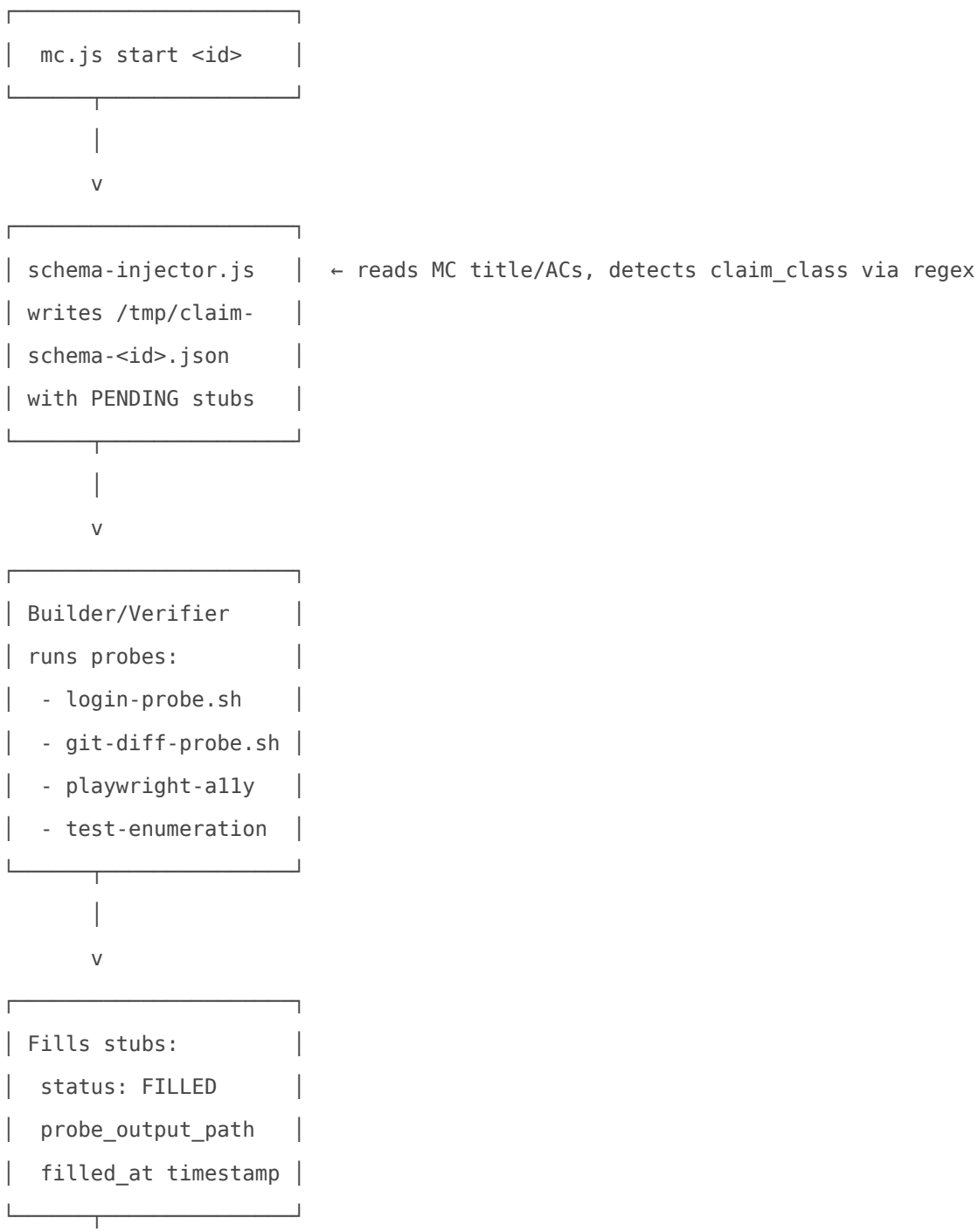
This gives 3 weeks for backfill of older MCs that started before the schema injector shipped.

Blocking Logic

```
# Extract MC ID from stdin
MC_ID=$(echo "$INPUT" | jq -r '.args[0] // empty')

SCHEMA_PATH="/tmp/claim-schema-${MC_ID}.json"

# Check if schema exists
if [ ! -f "$SCHEMA_PATH" ]; then
  if [ "$NOW" -lt "$GRACE_CUTOFF" ]; then
    # Grace period — warn and allow
    echo "WARN: No claim schema for MC #${MC_ID}" >&2
    exit 0
  else
    # Past grace period — block
    echo "BLOCKED: No claim schema for MC #${MC_ID}" >&2
    exit 1
  fi
fi

# Check for pending/failed stubs
PENDING_COUNT=$(jq '[.claim_stubs[]? | select(.status == "PENDING" or .status == "FAILED")] |
length' "$SCHEMA_PATH")

if [ "$PENDING_COUNT" -gt 0 ]; then
  echo "BLOCKED: MC #${MC_ID} has ${PENDING_COUNT} claim stub(s) not filled." >&2
  jq -r '.claim_stubs[]? | select(.status == "PENDING" or .status == "FAILED") | "  -
\(.claim_class): \(.status)"' "$SCHEMA_PATH" >&2
```

```
  exit 1
fi

# All stubs filled — allow
exit 0
```

Workflow Diagram



```
        |
        v
┌──────────────────────┐
| mc.js ready/done     |
└──────────┬───────────┘
           |
           v
┌──────────────────────┐
| schema-stub-gate.sh  |  ← hook checks stubs
|  - All FILLED? ALLOW |
|  - Any PENDING? BLOCK|
└──────────────────────┘
```

Test Invocation

```
# Simulate mc.js ready call with MC ID
echo '{"args":["101065"]}' | bash ~/.claude/hooks/schema-stub-gate.sh

# Expected: exits 1 if any stubs PENDING, exits 0 if all FILLED
```

Related

- **Parent MC:** #101065 (Deterministic Session Compiler)
- **Probe Registry:** 4 Deterministic Probes
- **Reality Anchor Doctrine:** v1 Final
- **Child MCs:** #101133 (login-probe), #101134 (git-diff-probe), #101135 (playwright-a11y), #101136 (test-enumeration)

Revision #2
Created 2026-05-17 12:05:05 UTC by John
Updated 2026-06-21 20:03:39 UTC by John