MVStudioPro Incident Report - Production Instability & Workflow Conflict

Report date: 2026-02-26
Incident window: 2026-02-24 to 2026-02-26 (UTC+08:00)
Classification: Technical incident (production instability + workflow conflict)

1) Timeline (chronological)
- 2026-02-25 17:52:46: EMERGENCY restore of stable health endpoint committed (5741add).
- 2026-02-25 19:16:44 to 20:28:13: Multiple "Agent auto-fix" commits repeatedly modified api/index.ts (b5be298, 9fb3c66, 3501a04).
- 2026-02-26 12:34:37: "Merge branch 'main' into agent/1772021630356" committed (d7dae01).
- 2026-02-26 12:36:49 and 12:38:29: Additional "Agent auto-fix" commits introduced non-runtime prose content into api/index.ts (8ffd933, 605e1b6).
- 2026-02-26 12:53:48: Hotfix restored a valid default handler in api/index.ts (62c2c7c).
- 2026-02-26 13:06:18 to 14:56:39: Supervisor and tier-routing changes merged after endpoint stabilization (03fe5c5, 8289fa8, 896f6a3).

2) Root causes
- Multiple branch conflicts: Concurrent agent branches and merge cycles produced inconsistent states in shared API entrypoint files.
- Rewrites overriding API handlers: Vercel rewrite routing to /api/index combined with repeated handler rewrites increased regression risk.
- Session routing mismatch: Health/diagnostics behavior diverged due mixed path detection from rewritten headers and URL candidates.
- Agent branch upstream failures: Repeated "Agent auto-fix" cycles indicate unsuccessful first-pass integration and repeated rework.
- Anonymous export function syntax crash: Non-executable content entered api/index.ts and created parser/runtime failure risk until hotfix restoration.

3) Technical impact
- Production API entrypoint stability degraded at /api/index.
- Health and diagnostics endpoint behavior became inconsistent during repeated rewrites.
- Deployment confidence and validation throughput declined due repeated rollback and re-apply cycles.

4) Time lost estimation
- Estimated engineering time lost: 8.5 to 11.0 hours during the incident window.
- Estimation basis: repeated restore/hotfix loops on api/index.ts from 2026-02-25 17:52 to 2026-02-26 12:53, plus merge conflict resolution and revalidation.

5) Remediation steps taken
- Restored a valid default Vercel handler in api/index.ts via commit 62c2c7c.
- Re-established deterministic method guard and stable response behavior for health checks.
- Continued routing and provider changes only after API entrypoint syntax and behavior were stabilized.

6) Final stabilization state
- Current api/index.ts is syntactically valid and exports a runtime handler.
- Rewrite policy remains explicit in vercel.json: /api/(.*) -> /api/index.
- Post-hotfix changes proceeded on top of a stable API entrypoint baseline.

7) Lessons learned
- Protect shared API entrypoint files with stricter merge policy and mandatory pre-merge compile checks.
- Block unresolved/generated prose from runtime TypeScript files via CI validation.
- Require targeted smoke checks for rewritten API paths before merge.

- Assign explicit conflict-resolution ownership on shared gateway files.

8) Verbatim error/operational log snippets

[Snippet A: git log branch conflict and stabilization sequence]
```
d7dae01 2026-02-26 12:34:37 +0800 Merge branch 'main' into agent/1772021630356
8ffd933 2026-02-26 12:36:49 +0800 Agent auto-fix (#26)
605e1b6 2026-02-26 12:38:29 +0800 Agent auto-fix (#24)
62c2c7c 2026-02-26 12:53:48 +0800 hotfix(api): restore valid default handler in
api index (#31)
```

[Snippet B: malformed api/index.ts content captured from commit 8ffd933]
```
 agent/1772021727663
The provided code is a simple health check endpoint for a Vercel serverless func
tion that responds with "ok" when the `/api/health` endpoint is called.
 agent/1772021630356
To ensure the code safely handles requests to `/api/health` without breaking, it
's important to implement proper error handling and to ensure that the server re
sponds correctly regardless of input.
 main
 main
```

[Snippet C: restored valid handler captured from commit 62c2c7c]
```
import type { VercelRequest, VercelResponse } from "@vercel/node";

export default function handler(req: VercelRequest, res: VercelResponse) {
  try {
    if (req.method !== "GET") {
      return res.status(405).setHeader("Allow", "GET").send("Method Not Allowed"
);
    }

    res.setHeader("Content-Security-Policy", "default-src 'self'");
    return res.status(200).send("ok");
  } catch (error) {
    console.error(error);
    return res.status(500).send("Internal Server Error");
  }
}
```

End of report.