PROOFLINE / PRODUCT VALIDATION CASE STUDY

From blocked Windows onboarding to a verified Windows compatibility fix

A public developer-product validation showing how functional QA, user friction, root-cause analysis, and regression proof become one concise evidence package.

4	107/107	10 files	0
confirmed product defects	tests passing after fixes	minimal compatibility patch	live calls or private data used

EXECUTIVE SUMMARY

The public CALL-E integration toolkit advertised a stable path for Codex and other agents. On a clean Windows environment, that path stopped before authentication and later failed validation and tests. The investigation separated documentation, dependency-policy, line-ending, and path-separator failures; produced a minimal patch; and proved the result with the complete local test suite.

The work did not require a production account, live telephone call, private repository, customer data, or bypass of identity controls.

What this demonstrates

+ Real-user onboarding validation, not only unit-level correctness.
+ Clear separation between confirmed defects, product friction, and untested assumptions.
+ Root-cause isolation with the smallest safe remediation.
+ Acceptance criteria and full regression evidence before calling a finding resolved.

Environment	Windows / PowerShell / Node 24.14.0 / pnpm 11.9.0 / public commit 46f0747
Scope	Stable onboarding guide, clean install, five integrations, repository checks, unit and end-to-end tests

CONFIRMED FINDINGS

Evidence translated into product impact

Severity reflects the point where a reasonable Windows user is blocked or misled. Every item below was reproduced against the same public commit.

1. Canonical quick start cannot run in PowerShell

HIGH

Customer impact: A Windows user following the advertised stable guide stops before authentication because the POSIX `env VAR=value command` form is not a PowerShell command.

Evidence: The exact guide command produced `CommandNotFoundException`. The same guide omitted the repository's Node >=22 prerequisite and its two recovery links returned HTTP 200 pages whose body said 404.

Resolution and proof: Render shell-specific commands, preflight Node and CLI dependencies, use canonical recovery URLs, and return a real 404 for missing documents. Acceptance: a clean PowerShell session reaches authenticated tool discovery without manual command translation.

2. Clean install fails under a documented pnpm range

HIGH

Customer impact: A new contributor cannot install dependencies with a pnpm version that satisfies the published `>=10.18.3` range.

Evidence: `pnpm install --frozen-lockfile` exited 1 because the esbuild postinstall was ignored. pnpm also inserted an unresolved `allowBuilds` placeholder into the versioned workspace file.

Resolution and proof: Define the approved-build policy explicitly or pin and activate the exact supported pnpm release. Acceptance: a clean install exits 0 and leaves tracked files unchanged.

3. Valid YAML frontmatter is rejected on normal Windows checkouts

HIGH

Customer impact: Five integrations fail validation even though their metadata is valid, blocking contributors and Windows CI.

Evidence: The files began with the valid bytes `2D 2D 2D 0D 0A` (`---\r\n`). Six extractors only accepted LF line endings.

Resolution and proof: Change delimiter expressions to accept `\r?\n`. Acceptance: all integration checks pass on both LF and CRLF checkouts.

4. Regression assertions assume POSIX path separators

MEDIUM

Customer impact: Six assertions in four test files fail falsely on Windows after the integrations are valid.

Evidence: Assertions searched for fragments such as `skills/calle`, while production code correctly emitted `skills\calle` through `path.join()`.

Resolution and proof: Build expected fragments with `path.join(...)`. Acceptance: the complete suite passes on Windows without weakening any assertion.

DELIVERY METHOD

A repeatable validation loop

1	**Model the journey** Define the user goal, prerequisites, supported environments, state transitions, and success evidence.
2	**Reproduce cleanly** Start from a clean environment and record the first point where actual behavior diverges from the promise.
3	**Reduce the cause** Change one variable at a time until the smallest failing condition and affected scope are known.
4	**Patch minimally** Fix the cause, preserve unrelated behavior, and state what the change deliberately does not address.
5	**Prove the outcome** Run focused checks plus the full available regression suite; keep the pre-fix failure and post-fix result.

Example Linear-ready ticket

TITLE	[Windows] Stable onboarding guide stops before authentication in PowerShell
ENVIRONMENT	Windows 11, PowerShell, Node 24.14.0, public stable guide retrieved 31 Jul 2026
STEPS	1. Open a clean PowerShell session. 2. Follow the published stable guide. 3. Run the first attributed command using env VAR=value command.
EXPECTED	The command sets attribution variables, runs the CLI, and proceeds to authentication or tool discovery.
ACTUAL	PowerShell raises CommandNotFoundException because env is not a PowerShell command.
IMPACT	High - the advertised Windows onboarding path is blocked before the product can be evaluated.
ACCEPTANCE	A shell-aware PowerShell snippet reaches authenticated tool discovery from a clean environment; missing recovery docs return HTTP 404.

REGRESSION RESULT

107 tests passed / 0 failed

The result included 35 unit tests, 13 CLI end-to-end tests, the five integration suites, and core checks. The patch was also verified to apply cleanly to a second fresh clone of the same commit.

ENGAGEMENT MODEL

How the same approach fits a customer-facing web product

For booking, payment, search, and support journeys, the technical evidence loop expands to include customer state, trust, recovery, accessibility, and commercial impact.

Journey	Core states and invariants	Evidence delivered
Booking	Availability, dates, party size, price, fees, confirmation, change and cancellation	State matrix, reproducible defects, recovery friction, confirmation consistency
Payment	Totals, discount, currency, decline, timeout, duplicate action, refresh and return	Test-method log, expected/actual, network or UI evidence, no unauthorized real charge
Search	Query intent, filters, empty states, sort, pagination, stale state and responsive behavior	Coverage grid, relevance/friction observations, exact examples and severity
Support	Entry point, issue classification, context retention, escalation, status and closure	End-to-end trace, missing context, broken handoffs and recovery recommendations

OPERATING GUARDRAILS

+ No real payment, destructive action, customer contact, or production change without explicit authorization.
+ No invented severity, reproduction, credential, device, or user testimony.
+ AI assistance is disclosed when relevant; every reported issue remains independently reproducible.
+ Unconfirmed observations are labeled as hypotheses, not defects.

Available for a focused first flow.
Start with one high-risk customer journey, receive a prioritized evidence pack, and evaluate the reporting quality before expanding coverage.

proofline-audit.chrismellado.chatgpt.site