

"WE ARE STILL IN STEALTH" IS NOT AN ARCHITECTURE DESCRIPTION

Not launched is a product-status sentence. It does not tell an assessor what exists, what is reachable, what can be activated, what already influences payment security, or what changes when the switch is thrown.

PRACTITIONER POSITION

A stealth-stage label can protect commercial timing, but it cannot describe technical state. My practitioner position is that every payment-relevant capability needs two separately evidenced propositions: what is effective now and what is proposed next. A third fact often matters just as much: what is already deployed, reachable, credentialed, or administratively activatable even if customers cannot use it yet. The transition between those states is not a storytelling edit. It is an accountable change that may reopen payment journeys, account-data and security-influence paths, component and identity inventories, provider responsibilities, public-presence records, vulnerability and penetration-testing populations, control evidence, reporting assumptions, and validation direction. The working artifact is therefore a capability-and-transition ledger-not a slide labeled current/future and not a founder saying, 'we are still in stealth.'

THE FIELD PATTERN

WHAT THIS LOOKS LIKE IN PRACTICE

A startup says its payment product is not live. The sales page is private and the ordinary customer route is disabled. Yet the production domain resolves, the edge accepts traffic, the callback receiver is active, live provider credentials exist in a secret store, an administrator can enable a feature flag without a deployment, a support console can replay transactions, and the proposed architecture diagram shows only the intended hosted payment flow. Engineering calls these pre-launch plumbing. Product calls them dark features. The assessment file calls the whole environment future state. Nobody has recorded which capabilities work now, which are merely designed, which are dormant but production-capable, who can activate them, what evidence proves each state, or which PCI claims must be reopened when a transition occurs. The launch meeting is being asked to approve a label rather than a known architecture.

SIGNALS DETECTED

WHAT SHOULD MAKE YOU STOP

- Current-state diagrams contain proposed components, while proposed-state diagrams omit production-capable infrastructure that already exists.
- Not launched, disabled, dark, beta, sandbox, internal only, and no customers are used as if they were technical control states.

- A feature flag, provider-console setting, DNS change, secret promotion, routing rule, or administrator action can expose a payment path without a conventional release.
- Live credentials, production tenants, public endpoints, callback receivers, queues, logs, support tools, or administrative interfaces exist before the declared launch date.
- A component is called future state even though it can already reach, alter, redirect, observe, authenticate to, or recover a payment-relevant service.
- The only transition record is a product launch checklist; it does not enumerate affected identities, providers, boundaries, tests, evidence, or claims.
- The team can show what the happy path is supposed to become but cannot demonstrate the negative state of alternate, legacy, support, failure, or rollback paths today.
- Development, staging, preview, and production names are accepted without evidence of actual data, credentials, reachability, administrative control, and deployment relationships.
- The scope statement changes when the same architecture is described to engineering, leadership, the assessor, and the compliance-accepting entity.
- Nobody owns the moment when a proposed capability becomes an effective fact or the obligation to reopen the affected assessment record.

WHY IT MATTERS

THE COST OF LEAVING IT UNEXPLAINED

Start by separating four states that product vocabulary routinely collapses. Proposed means designed or planned but not yet effective. Present means the component, identity, provider service, route, or control exists in the observed environment. Production-capable means an existing capability can become effective through an authorized or unauthorized action such as a feature flag, credential use, routing change, provider-console setting, deployment promotion, or administrative invocation. Effective means the capability is operating for the stated journey, population, and period. These are practitioner state labels; they are not new PCI SSC terms or automatic scope categories.

Treat each state as a proposition with evidence. A roadmap proves intent, not absence. A disabled user-interface control may prove one presentation state, not that an API, webhook, support function, direct URL, old client, provider console, or administrator cannot invoke the capability. A production-capable conclusion should identify the enabling action, actor, credential, dependency, time, observation method, and evidence. An absence claim needs defined discovery sources and negative-path testing, not confidence in naming conventions.

Model the capability, not just the component. One service can expose multiple capabilities with different data and influence consequences: accept a payment request, redirect a browser, receive a callback, retry a transaction, reveal status, export records, change provider configuration, rotate a credential, deploy code, or bypass the ordinary journey. Recording only that a lambda, container, SaaS platform, or application exists hides the behavior that matters to architecture and assessment claims.

Give current and proposed states their own relationship rows. For each capability, join the payment journey, account-data state, security-influence path, component, identity, provider service, boundary, public presence, owner, configuration source, observed evidence, control, test population, and claim. Do not

overwrite the current row with the target design. Preserve what was true, what is intended, and what changed so a reviewer can reconstruct the period without founder oral history.

Interrogate dormant capability. A feature can be unused by customers yet already matter because its code is deployed, its endpoint is reachable, its credential is live, its administrator can activate it, or its failure path affects a protected journey. None of those facts alone determines PCI DSS scope. Together they are inputs to the entity's scope determination and the assessor's evaluation of whether the documented boundary and applicable controls are accurate.

Make the transition a governed object. Record the exact from-state and to-state, triggering action, change record, approver, planned and observed time, rollback or deactivation path, affected relationships, evidence before and after, and the person authorized to stop the transition. A launch date is not enough. The transition may occur earlier through infrastructure, provider configuration, DNS, identity, or support operations-and may partially revert while leaving residual capability behind.

Build a claim-impact fan-out. When a capability becomes present, reachable, production-capable, or effective, identify which scope, segmentation, account-data, provider-responsibility, inventory, secure-development, change-control, vulnerability-scanning, penetration-testing, logging, access, incident-response, evidence, reporting, and validation propositions need reconsideration. Reopening a proposition does not pre-decide its result. It prevents a stale result from surviving a changed fact.

Use significant-change guidance as a floor for disciplined evaluation, not as a magic label generator. PCI SSC FAQ 1317 identifies changes to CDE hardware, software, networking, account-data flow or storage, boundaries, supporting infrastructure, and third-party providers or services as activities that can have security impact and must be evaluated in the entity's context. The capability ledger makes that evaluation traceable before a launch event scatters the evidence across tickets and chat.

Reconcile discovery sensors instead of appointing one source as truth. Source control shows code and configuration that may never be deployed. Infrastructure-as-code misses manual and provider-side changes. Cloud inventory can show resources without explaining payment purpose. Feature systems show declared flags without proving all invocation paths. DNS, certificates, secrets, identity, observability, vulnerability data, provider consoles, and tickets each reveal a partial state. The governed record states what each source observed, where they disagree, and who disposed of the mismatch.

Keep commercial disclosure separate from assessment completeness. A public roadmap can stay quiet. A pitch deck can remain narrow. The internal architecture, controlled assessor evidence, and accountable entity representation still need the facts necessary to evaluate the claim. The second installment in this series governs where those facts go; this installment governs whether the facts have been correctly separated by technical state before anyone decides how broadly to disclose them.

Run the Tuesday test before launch. Ask a reviewer who did not attend the design meetings to select one capability and determine what it can do today, what it cannot do, what evidence supports both propositions, who can change the state, which action would make it effective, and which claims would reopen. Then perform one authorized transition in a fictional or safely controlled exercise and verify that the diagrams, inventories, provider records, public-presence list, test populations, evidence contracts, and claims ledger all move together.

Close with a bounded disposition, not a green launch badge. The accountable decision should state the capability, current state, proposed state, production-capable path, known evidence, untested remainder, affected claims, required actions, owner, review date, and reopening triggers. Applicability and validation direction remain with the relevant compliance program and compliance-accepting entity; assessor conclusions remain assessor judgment. The practitioner method makes the technical facts ready for those decisions without pretending to make them automatically.

QUESTIONS

QUESTIONS THAT EXPOSE THE WACK

- 01 What can this capability do today, regardless of what the product calls its lifecycle stage?
- 02 Which evidence distinguishes proposed, present, production-capable, and effective states?
- 03 What actor, credential, flag, route, provider setting, deployment, DNS record, or support action can change the state?
- 04 Can the capability be invoked through an alternate, direct, legacy, administrative, retry, failure, or rollback path?
- 05 Which payment journey, account-data state, security-influence path, component, identity, provider service, boundary, or public presence changes with it?
- 06 Which inventory, diagram, scan population, penetration-test proposition, control evidence, responsibility statement, or assessment claim must be reopened?
- 07 What source proves absence rather than merely failing to show presence?
- 08 Who is authorized to approve, perform, stop, reverse, and attest to the transition?
- 09 What remains deployed, credentialed, reachable, or influential after rollback or deactivation?
- 10 Could a cold reviewer reconstruct both states and the transition without the founder, architect, or product lead in the room?

THE CLUSTER OF WACK

FIND / MAP / EXPLAIN / REDUCE / PROVE

ESCAPE THE WACK

01 / FIND

Enumerate payment-relevant capabilities across code, infrastructure, cloud, identity, DNS, secrets, provider consoles, flags, observability, vulnerabilities, tickets, and support paths. Record proposed, present, reachable, production-capable, and effective facts separately.

02 / MAP

Join each capability and transition to journeys, data and influence paths, components, identities, providers, boundaries, public presences, configurations, owners, evidence, controls, tests, claims, and reopening triggers using stable identifiers.

03 / EXPLAIN

Write one bounded current-state proposition and one bounded proposed-state proposition. Identify what could activate the capability, what proves the negative paths, where sources disagree, and which authority owns each resulting decision.

04 / REDUCE

Remove unused deployed code, dormant routes, live credentials, unmanaged flags, orphaned endpoints, alternate invocation paths, stale provider settings, and ambiguous environment labels that create capability without an owned purpose or exit.

05 / PROVE

Have a cold reviewer reproduce the state from evidence, then exercise one safely authorized transition and show that every affected relationship, test population, evidence contract, and claim reopens and closes through accountable disposition.

THE FIELD LAW

THE LINE WORTH REMEMBERING

NOT LAUNCHED IS NOT THE SAME AS NOT PRESENT. NOT ENABLED IS NOT THE SAME AS NOT CALLABLE. PROPOSED IS NOT CURRENT. A TRANSITION IS NOT COMPLETE UNTIL THE AFFECTED CLAIMS KNOW IT HAPPENED.

Download the Current-versus-Proposed Capability Ledger. Select one dark, beta, internal, disabled, sandbox, or pre-launch payment capability and prove what exists now, what can become effective, who can change it, and which PCI propositions the transition reopens.

TECHNICAL GROUNDING

START WITH THE SOURCE MATERIAL

These references ground the practitioner interpretation. They do not turn the metaphor or method into official PCI DSS terminology.

- WACK-SOURCE-0001 / PCI DSS v4.0.1 / PCI Security Standards Council / [Open source](#)

- WACK-SOURCE-0005 / PCI DSS Scoping and Segmentation Guidance for Modern Network Architectures / PCI Security Standards Council / [Open source](#)

- WACK-SOURCE-0008 / FAQ 1473: Role of compliance-accepting entities and assessors in determining applicability / PCI Security Standards Council / [Open source](#)

- WACK-SOURCE-0013 / PCI SSC Document Library / PCI Security Standards Council / [Open source](#)

- WACK-SOURCE-0088 / FAQ 1317: What is meant by significant change in PCI DSS? / PCI Security Standards Council / [Open source](#)

AUTHORITY CLASSIFICATION / PCI_SSC_SOURCE / ASSESSOR_JUDGMENT / PRACTITIONER_POSITION / PRACTITIONER_METHOD / AI_ASSISTED

Practitioner reviewed by Scott Norton on 2026-09-13 against canonical digest

c548a19b349120d842de01f84bc089d16b809dfdb80adbbc35050751f920b5b1. Educational content, not a compliance determination.

Independent practitioner resource; not affiliated with or endorsed by PCI SSC. Authorship and authority classification do not determine legal intellectual-property ownership.