

pdk × Portaldot — Development & Operations Plan

Pugar Huda Mantoro

2026-07-02

Contents

pdk × Portaldot — Development & Operations Plan	1
1. Executive Summary	2
2. Product Roadmap	2
v0.2 — TypeScript Companion SDK (Target: 4–5 weeks)	2
v0.3 — Editor Extensions + Community KB (Target: 3 weeks after v0.2)	3
v1.0 — Mainnet-Ready (Target: aligned with POT mainnet launch)	3
3. Marketing & Social Presence	3
X (Twitter) — the official pdk account	3
Cross-promotion with Portaldot official channels	4
4. Website & Documentation	4
Current: portaldot-pdk.vercel.app	4
Planned upgrades (~1 week after X account launches)	4
Custom domain	5
5. Whitepaper	5
6. Community — Discord Channel	5
7. Operations Plan	5
Release cadence	5
Contribution model	6
Uptime & reliability	6
8. Support Requested from Portaldot	6
High-priority	6
Medium-priority	6
Optional (if grant program exists)	6
9. Timeline — 30/60/90 Day	6
First 30 days (starting week of 2026-07-07)	6
Days 30–60	7
Days 60–90	7
10. Success Metrics (12 months)	7
12. Sustainability & Risk	7
Sustainability & monetization stance	7
Bus factor & risk mitigation	7
Risks I've thought about	7
13. References & Benchmarks	8

pdk × Portaldot — Development & Operations Plan

1. Executive Summary

pdk (Portaldot Dev Kit) is a Python CLI that decodes failed Portaldot transactions into named errors with concrete fix steps. It won 1st place in the Builder Tools track of the Portaldot Online Mini Hackathon S1 and is already public on PyPI (`pip install portaldot-pdk`). 14 commands, real POT gas, zero mocks.

Current state: - v0.1.6 shipped, 40 unit + 84 integration/stress tests green. - Auto-publish pipeline (PyPI on every v* tag) already live. - Public site + pitch deck + 3-minute demo video ready for community voting.

What I want to build next (with Portaldot support): A three-phase roadmap that turns pdk into the *default* debugging layer of the Portaldot developer experience — Python CLI → TypeScript companion SDK → editor extensions → mainnet-ready public tooling.

What I need from Portaldot: 1. Recognition/inclusion in official developer onboarding docs. 2. Builder-channel access to stay close to runtime upgrades. 3. Co-marketing on the X / Discord side once the official pdk presence is live. 4. (Optional) Grant support to fund the v0.2 TypeScript SDK work.

2. Product Roadmap

v0.2 — TypeScript Companion SDK (Target: 4–5 weeks)

The current biggest limitation of pdk is that the Python `substrate-interface` library mis-signs custom-pallet calls on Portaldot's V13 metadata (upstream bug I diagnosed mid-hackathon, filed as [polkascan/py-substrate-interface#9]).

That means pdk cannot currently: - Sign Assets pallet transactions - Deploy ink! contracts - Cover the full runtime surface

Solution: a TypeScript layer, shipped as `pdk-ts`, that speaks whatever Python cannot. Same UX, same 14 commands, but Node.js-backed for full runtime coverage.

Why TypeScript, not fix the Python bug? The `substrate-interface` maintainer engagement on issue #9 is slow. Even if fixed upstream, Python remains a second-class citizen on Portaldot metadata V13 — new pallets break signing again on next runtime upgrade. TypeScript via `@polkadot/api` (or **PAPI**, see below) is the ecosystem's first-class path.

Why @polkadot/api or PAPI? The Polkadot ecosystem is actively migrating away from `@polkadot/api` toward PAPI (Polkadot-API): > *PAPI is not an official replacement for polkadot.js, but it is > the recommended default for most new web dapps thanks to its > light client first design, strong TypeScript typing, and small > bundles.* — [Polkadot Ecosystem docs — PAPI]

The v0.2 spike will benchmark both against Portaldot's V13 metadata in week 1 and lock in the winner. Metadata-driven architecture (decoder reads chain types at runtime) means we can migrate the under-the-hood library without changing pdk's user-facing CLI — same design bet PAPI itself made against `@polkadot/api`, which pays off in maintenance.

Why 4–5 weeks realistic? Comparable ecosystem timelines: - Foundry (Paradigm, Rust) launched Dec 2021 as a *reimplementation of dapptools* — the initial usable release came within 4–6 weeks of the internal spike. ([Paradigm — Introducing Foundry]) - `viem` (wevm) went from zero to production-ready TypeScript library in ~3 months by starting from a smaller-scope kernel and layering outward. `pdk-ts` is a smaller ambition (single-chain, 14 commands) so 4–5 weeks is realistic *because we already know the runtime surface* — the same 202-error metadata index that Python-pdk uses is transferable verbatim.

Deliverables - `pdk-ts` npm package - Feature parity with pdk for all 14 commands - Assets pallet operations working end-to-end - ink! contract deploy + interact - Documented migration guide (Python → TS or hybrid)

v0.3 — Editor Extensions + Community KB (Target: 3 weeks after v0.2)

Move FailLens into the IDE where developers actually live, and formalize the community knowledge-base contribution flow.

Deliverables - VS Code extension: hover any raw `Module { index, error }` code in the terminal, get inline decoded error + fix. - Cursor extension (same codebase, different marketplace). - Community `error_fixes.yaml` contribution guide + first 10 community-authored fixes onboarded. - Public issue tracker for the KB.

v1.0 — Mainnet-Ready (Target: aligned with POT mainnet launch)

Deliverables - One-flag testnet support: `pdk --node wss://testnet.portaldot.io` (already coded, just needs testnet endpoint). - Mainnet metadata index regenerated + verified. - Long-tail QA harness against public RPC. - Security audit of any code paths that touch signing. - Stable API contract for the KB schema (community can rely on it).

3. Marketing & Social Presence

X (Twitter) — the official pdk account

Account identity

Field	Value
Display name	pdk — Portaldot Dev Kit
Handle (preferred)	@portaldotdevkit
Handle fallbacks	@pdk_portaldot · @portaldot_pdk · @use_pdk · @pdk_dev
Location	Global · open-source
Website	portaldot-pdk.vercel.app
Category	Software / Developer Tools
Registration Gmail	portaldotdevkit@gmail.com

Bio (280 chars) Three drafts — recommend Version A (informative, matches the `pip install-first-thing` convention that @wagmi_sh and @ethers_io use):

Version A — technical/direct (recommended):

The debugger for Portaldot. Turn cryptic transaction errors into named fixes, live in your terminal. 14 commands, one CLI, real POT gas.

Version B — community-first:

pdk — the Portaldot Dev Kit.

Every failed Portaldot transaction, decoded. Community-owned error knowledge base. Built for builders.

Version C — punchy:

Portaldot errors, decoded. 🧐

The dev kit for Portaldot builders — 14 commands in one CLI, real POT gas, zero mocks. Free & open-source.

```
pip install portaldot-pdk
```

Launch content plan (first 2 weeks)

Day	Post	Rationale
1	Launch tweet — 3-min demo video + install command	Anchors the account. Video-first launch mirrors how Vercel/Turborepo/Bun launch products on X.
2	Thread: “the story behind FailLens” (Day-1 metadata spike)	Story-driven, humanises the tool. Same tactic Paradigm used for Foundry v1.0 announcement.
3	Short vertical clip: <code>pdk debug --demo</code> in action (15s)	Vertical clips retain 3–4× longer than horizontal on X.
5	Thread: 3 most common Portaldot errors, decoded	Adds SEO gravity — Portaldot devs searching for error text will find <code>pdk</code> .
7	Community CTA: “hit a Portaldot error we haven’t decoded yet? 5-line YAML PR.”	Sets contribution norm early. Explicit, low-friction ask.
10	v0.2 announcement + TypeScript SDK preview	Shows roadmap velocity while the launch attention is still warm.
14	First community KB contribution highlight	Rewards the first contributor publicly — creates a template others follow.

Cadence, long-term: 3–4 posts/week. Marketing research on technical founder accounts finds *consistency* is a stronger predictor of audience growth than volume — “founders who post consistently for 90 days build enough of an audience that a single announcement post reliably reaches thousands of qualified people” ([Okara — X Marketing for Founders]). Posting more than daily without genuine substance is a well-documented net-negative for dev tools.

Cross-promotion with Portaldot official channels

- Retweet-worthy launches: v0.2 SDK ship, editor extension ship, public testnet integration, mainnet-ready milestone.
- Guest-post on Portaldot’s own blog / newsletter if that channel exists.
- Portaldot team members occasionally quoting `pdk` output when helping developers in Discord (organic, not forced).

4. Website & Documentation

Current: `portaldot-pdk.vercel.app`

Already live. Landing hero + 3-minute video + pitch deck + install instructions. Auto-deploys on every commit.

Planned upgrades (~1 week after X account launches)

- **Docs section:** full command reference (auto-generated from Typer CLI), tutorial (“your first `pdk` transaction”), FailLens error index browser.

- **Community KB browser:** searchable UI over `error_fixes.yaml`, showing contributor attribution.
- **Roadmap page:** public v0.2 / v0.3 / v1.0 tracker.
- **Changelog:** auto-populated from CI release pipeline.

Custom domain

Once Portaldot okays the naming, migrate to `pdk.portaldot.io` or similar sub-domain (needs DNS delegation from your side). Fallback: `pdk.dev` or `portaldot-pdk.com` (I'll register whichever the team prefers).

5. Whitepaper

Scope: 8–12 pages, technical + narrative.

Sections planned: 1. Problem statement — the developer experience gap on Portaldot 2. Architecture — metadata-driven decoding, no hardcoded error tables 3. Coverage matrix — commands × runtime surface 4. Community knowledge base — governance & contribution model 5. Roadmap alignment with Portaldot mainnet 6. Comparison with existing Substrate tooling 7. Security model — what pdk trusts, what it verifies 8. Case studies — Day-1 spike, substrate-interface#9 diagnosis, repo-rename recovery 9. References + benchmarks

Timeline: first draft in 3 weeks, iteration + Portaldot review in weeks 4–5.

6. Community — Discord Channel

Two options here, and I'd like your input:

Option A (preferred): dedicated #pdk channel inside Portaldot's official Discord. Lower friction for existing community members, tighter loop with runtime updates from the core team, and Portaldot's brand stays central. I'd moderate the channel and act as first responder for pdk-related questions.

Option B: standalone pdk Discord server. Fallback if Option A isn't feasible. Would still cross-link with Portaldot's official Discord.

Why Option A? Ecosystem-tool channels living inside the parent chain's Discord is the industry norm — foundry-rs uses #foundry-help inside various chain Discords rather than balkanising the community. Standalone servers work for full-stack products (wallets, exchanges) but for a debugging CLI, the audience is already in Portaldot's Discord asking about the errors pdk decodes.

Community activities planned: - Weekly "what did we learn this week" post — new KB entries, runtime observations, ecosystem quirks. - Monthly community call once contributor base grows (~10+ people). - Fast-track review for KB PRs (24–48h SLA).

7. Operations Plan

Release cadence

- **Patch releases** (`v0.x.y`) — as needed, auto-published via CI.
- **Minor releases** (`v0.x.0`) — every 4–6 weeks aligned with milestone deliverables.
- **Major releases** (`v1.0.0`, `v2.0.0`) — aligned with Portaldot runtime milestones (mainnet, major upgrades).

Contribution model

- All contributions via GitHub PR (existing flow).
- KB additions: 5-line YAML template, no code review overhead.
- Feature contributions: 1–2 maintainer review, CI must pass.
- CONTRIBUTING.md already committed with the standards.

Uptime & reliability

- CI runs on every push, blocks merges on failure.
 - 40 unit + 84 integration tests, coverage tracked.
 - Nightly runs against latest Portaldot node build (once testnet is public).
-

8. Support Requested from Portaldot

Three concrete asks, in priority order:

High-priority

1. **Inclusion in official developer onboarding docs.** Adding pdk as the recommended debugging tool in “your first Portaldot transaction” tutorials means every new builder finds pdk in their first hour, not after they’ve hit five undecoded errors.
2. **Builder-channel access.** Direct line to core team announcements — runtime upgrades, new pallets, breaking changes. Lets pdk ship decoder updates *before* the upgrade hits mainnet.

Medium-priority

3. **Co-marketing on official X / Discord.** Once the official pdk X account is live, occasional retweets or Discord announcements amplify community reach. I won’t spam — only for milestone launches.

Optional (if grant program exists)

4. **Grant support for v0.2 TypeScript SDK.** ~4 weeks of focused engineering. This direction would be highest-leverage because it directly unblocks Assets + ink! contract deploy support that the whole Python builder base is currently locked out of.
 5. **Testnet endpoint access early.** Even a private testnet URL would let me pre-integrate pdk before public launch. pdk is already one-flag-ready (`--node wss://<endpoint>`) so no code work is required on our side — this ask is purely about early access.
-

9. Timeline — 30/60/90 Day

First 30 days (starting week of 2026-07-07)

- Week 1: X account live, first launch tweet, dedicated #pdk discussion inside Portaldot Discord (Option A above).
- Week 1: Whitepaper draft v1.
- Weeks 2–4: v0.2 TypeScript SDK scaffolding + Assets pallet operations.
- Week 4: v0.2 alpha release.

Days 30–60

- v0.2 GA release.
- Website docs section + KB browser.
- First 10 community KB contributions onboarded.
- Whitepaper published.

Days 60–90

- v0.3 — VS Code + Cursor extensions.
 - Testnet integration (assuming public endpoint by then).
 - First community call.
 - v1.0 release candidate for mainnet alignment.
-

10. Success Metrics (12 months)

- **Adoption:** 500+ PyPI installs/month, 100+ GitHub stars.
 - **Coverage:** 400+ error indices in KB (up from 202), including all Portaldot mainnet pallets.
 - **Community:** 20+ external KB contributors, 3+ TypeScript SDK contributors.
 - **Ecosystem integration:** pdk referenced in official Portaldot docs, mentioned in 5+ builder-facing blog posts.
 - **Reliability:** 99%+ CI green over the year, zero unplanned breaking changes.
-

12. Sustainability & Risk

Sustainability & monetization stance

- **Free & open-source, forever.** MIT license. No paid tiers on the CLI itself. Same stance as viem, wagmi, foundry.
- **Ecosystem grants are the sustainability model.** Portaldot ecosystem funding (\$8) covers v0.2. After that, either continued ecosystem support or a Retro-PGF-style bounty-per-PR model for KB contributors funded by the chain foundation.
- **What we won't do:** paid analytics, telemetry, "pro plans", hosted paid services. pdk exists to help Portaldot devs; the moment it monetizes them, it stops being neutral tooling.

Bus factor & risk mitigation

Mitigations in place already: - **Fully automated release pipeline.** CI green + v* tag → auto-publish to PyPI. Any repo collaborator can cut a release — no personal credential dependency after the initial setup. - **CONTRIBUTING.md + comprehensive tests** committed already — 40 unit + 84 integration/stress. New contributors can onboard without me. - **Metadata-driven design.** Even if maintenance paused, the decoder self-regenerates against the chain's runtime metadata — it won't rot silently. - **KB is community-owned.** error_fixes.yaml is intentionally the *simplest possible* contribution format (5-line YAML). Fully survives maintainer change.

Mitigations to add in the first 90 days: - Onboard 2 co-maintainers with merge rights (target: week 8, after v0.2 ships). - Publish GOVERNANCE.md (see §7) with explicit succession clause. - Move the release-key custody to a shared vault (e.g., 1Password team account owned by the pdk Gmail from §3).

Risks I've thought about

Risk	Mitigation
Runtime upgrade breaks decoder	Metadata regen script runs in CI; alerts on schema diff
Upstream @polkadot/api deprecates in favour of PAPI	v0.2 spike decides which library to build on (\$2)
Community KB stays empty (nobody contributes)	Seed with 20+ curated entries; publicly credit early contributors
PyPI account compromise	2FA + trusted-publishing via GitHub OIDC (already partially set up; hardening in week 1)
Portaldot mainnet delays	pdk works against any Substrate node — value doesn't gate on mainnet timing
I get hit by a bus	See "Bus factor" above; co-maintainer onboarding is the primary hedge

13. References & Benchmarks

Every non-obvious choice above traces back to a precedent. Full list:

Ecosystem tooling references - Paradigm — Introducing the Foundry Ethereum development toolbox (2021) - Paradigm — Announcing Foundry v1.0 (2025) — 8.6k stars, 466 contributors, 4939 PRs benchmark - foundry-rs/foundry on GitHub - viem.sh docs — Why viem — reliability, DX, stability positioning - wevm/viem GitHub - Dynamic.xyz — The Promise of viem

Polkadot / Substrate ecosystem - Polkadot Ecosystem — PAPI (Polkadot-API) intro — PAPI as default recommendation for new TS dapps - Polkadot Forum — Why Polkadot-API? - Polkadot Developer Docs — API libraries - Polkadot Ecosystem — Development Tools directory

Portaldot-specific - Portaldot website - Portaldot ecosystem page - Portaldot developer docs (readthedocs)

Governance & sustainability - Open Source Guide — Leadership & Governance — governance model taxonomy - BDFL — Wikipedia — model context - PEP 8016 — The Steering Council Model (Python) — canonical BDFL → council transition - ArXiv 2509.16295 — Patterns in the Transition From Founder-Leadership to Community Governance of Open Source — data on OSS governance transitions

Marketing & X/Twitter cadence - Okara — X Marketing for Founders — "consistency > volume" data for technical founders - wevm_dev on X — wagmi vs viem positioning — dev-tool account tone reference

Own artifacts (this repo) - Pitch deck: portaldot-pdk.vercel.app/slide - Community-voting submission: docs/community-voting-submission.md - Showcase Q&A: docs/showcase-qa.md - Live-demo narration: docs/live-demo-narration.md - Repository: github.com/PugarHuda/portaldot-hackathon-2026-pdk-AmpunBang