

PDK — Portaldot Dev Kit · Whitepaper (Draft v0.1)

Pugar Huda Mantoro

2026-07-08

Contents

PDK — Portaldot Dev Kit	1
Whitepaper (Draft v0.1)	1
Abstract	1
1. Problem Statement — the developer experience gap on Portaldot	1
2. Architecture — metadata-driven decoding, no hardcoded tables	2
3. Coverage matrix — commands × runtime surface	3
4. Community knowledge base — governance & contribution model	3
5. Roadmap alignment with Portaldot mainnet	3
6. Comparison with existing Substrate tooling	4
7. Security model — what PDK trusts, what it verifies	4
8. Case studies	4
9. References + benchmarks	5
Appendix A — Command reference (pdk 0.1.6 + pdk-ts α.1)	5
Appendix B — KB schema	5

PDK — Portaldot Dev Kit

Whitepaper (Draft v0.1)

Author: Pugar Huda Mantoro (solo maintainer) **Date:** 2026-07-08 **Status:** DRAFT — first pass, iteration expected with Portaldot review.

Abstract

Portaldot's rejection path — the `DispatchError { Module: { index, error } }` returned when a transaction fails — is the single biggest friction point for new builders on the chain today. This paper describes **PDK (Portaldot Dev Kit)**, an open-source, metadata-driven developer toolkit that turns raw dispatch codes into named errors and actionable fix steps, and its two-language architecture (pdk Python + pdk-ts TypeScript) that maintains full runtime coverage as Portaldot evolves. PDK was 1st place in the Builder Tools track of the Portaldot Online Mini Hackathon S1 and is intended as ecosystem infrastructure, not a one-off hackathon entry.

1. Problem Statement — the developer experience gap on Portaldot

Portaldot is Rust-first, Substrate-based, and in Season 1 every builder runs a local node. The organizer-intended developer environment ships with:

- A local Portaldot node binary (Linux + macOS)
- An official Python SDK (portaldot-py)
- Runtime metadata that includes 202 named errors across 31 pallets
- **No CLI, no error decoder, no scaffolding, no faucet, no knowledge base**

When a transaction fails, the node prints exactly one line:

```
ExtrinsicFailed: DispatchError { Module: { index: 6, error: 2 } }
```

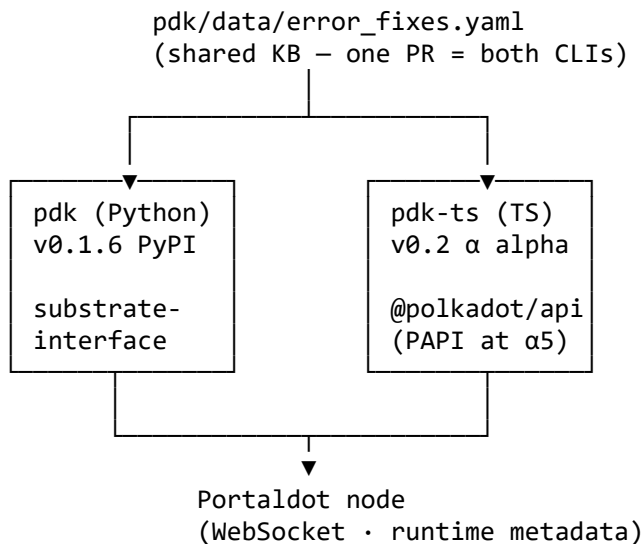
There is no `--verbose` flag that names it, no error catalog in the docs to look it up in, and no ecosystem tool that decodes it. The Portaldot Discord's most-asked question during S1 was *"how do I get POT?"* — a question with an answer already in the runtime (pre-funded dev accounts), just not a discoverable one.

The gap is not knowledge; it is discoverability. Every fact the decoder needs is already in the chain's own metadata. What was missing was the layer that reads it and speaks it back to the developer in readable English.

2. Architecture — metadata-driven decoding, no hardcoded tables

PDK's core design principle is that the chain is the source of truth. No pallet lists, no dispatch tables, no error names are hardcoded in either the Python or TypeScript CLI. Every invocation:

1. Opens a WebSocket to a Substrate/Portaldot node
2. Fetches the current runtime metadata
3. Walks the pallets → errors index to resolve raw codes
4. Enriches the named error with human-authored fix steps from a shared YAML knowledge base



This design means:

- **Runtime upgrades never rot the decoder.** New pallets, renamed errors, or reindexed dispatch tables update automatically because the decoder reads them fresh every call.
- **A community fix contributed once benefits both CLIs.** The Python and TypeScript sides both read `pdk/data/error_fixes.yaml`. Cross- porting is unnecessary.
- **Adding a new command is orthogonal to adding a new chain.** PDK's smoke tests run against public Polkadot RPC (`wss://rpc.polkadot.io`) in CI to prove chain-agnosticism — the tool works on any Substrate chain, but ships polished specifically for Portaldot.

3. Coverage matrix — commands × runtime surface

PDK exposes 14 commands split into read-only, write, and diagnostic tiers. Support state as of alpha.1:

Command	Python (v0.1.6)	TypeScript (α.1)	Runtime surface
doctor	☐ shipped	☐ shipped	System, Contracts pallet presence, runtime
accounts	☐ shipped	☐ shipped	System.account query for dev keypairs
pallets	☐ shipped	☐ α.2	Runtime metadata pallet enumeration
storage	☐ shipped	☐ α.2	Any storage entry, generic API
keys	☐ shipped	☐ α.2	SS58 encode / decode / keypair inspect
simulate	☐ shipped	☐ α.3	Fee estimation, balance feasibility
send	☐ shipped	☐ α.3	Balances.transfer_keep_alive (signed)
seed	☐ shipped	☐ α.3	Bulk fund from YAML fixture
debug	☐ shipped	☐ α.4	FailLens — decode failed tx (hero feature)
explain	☐ shipped	☐ α.4	Raw code → named error (unique to PDK)
report	☐ shipped	☐ α.4	Recent-block failure aggregation
watch	☐ shipped	☐ α.4	Live event stream, pallet-filtered
ai-setup	☐ shipped	☐ α.4	Optional AI decoder configuration
debug --fix	☐ shipped	☐ α.4	Diagnose + submit corrected transaction

pdk-ts reaches parity at beta.1, then ships as portaldot-pdk-ts on npm at 0.2.0.

4. Community knowledge base — governance & contribution model

The `error_fixes.yaml` file is deliberately the simplest possible contribution artifact — a five-line YAML block per new fix. This is a governance design choice, not an implementation detail: the goal is that any Portaldot builder who hits an undocumented error can spend 90 seconds writing a fix and the entire ecosystem benefits.

Contribution flow:

```
- pallet_index: 6
  error_index: 2
  name: Balances.InsufficientBalance
  what_happened: You tried to transfer more POT than the account holds.
  how_to_fix:
    - Check the sender balance.
    - Lower the amount, or fund the account first.
```

PR review criteria are three checks:

1. Runtime index matches (portaldot-1002 metadata is the arbiter)
2. `what_happened` reads clean at 6th-grade English level
3. `how_to_fix` contains at least one actionable step

Governance progression: BDFL (Benevolent Dictator for Life, i.e. solo maintainer) today → steering-council model when the contributor base crosses ~20 active people. This follows the Python community's canonical transition path (PEP 8016). A `GOVERNANCE.md` will land before v0.2.0 to document succession triggers.

5. Roadmap alignment with Portaldot mainnet

PDK v1.0 is designed to align with Portaldot's own mainnet milestone. Every version bumps is scoped so testnet integrations, mainnet freeze windows, and public RPC endpoints slot in naturally:

- **v0.1.x** (shipped) — Python CLI, local-node dev loop, 202-entry error index against portaldot-1002
- **v0.2** (alpha.1 shipped, in progress) — TypeScript companion, full runtime coverage including Assets + ink! contract deploy
- **v0.3** (post-alpha.4) — VS Code + Cursor editor extensions
- **v1.0** (aligned with POT mainnet launch) — public RPC support, mainnet metadata index regenerated + verified, external security audit of signing paths, stable KB schema

Because pdk-ts already passes integration tests against public Polkadot RPC today, `--node wss://<official-portaldot-rpc>` is a one-flag substitution when the endpoint becomes available.

6. Comparison with existing Substrate tooling

Tool	Language	Scope	Portaldot-aware?
substrate-interface (Python)	Python	Low-level SDK	Partial (V13 signing bug)
polkadot.js / @polkadot/api	TS/JS	Low-level SDK	Yes
PAPI (polkadot-api)	TS	Modern light-client SDK	Yes (metadata-driven)
Subxt (Rust)	Rust	Low-level SDK	Yes
PDK	Py + TS	CLI + FailLens + KB	Yes, primary focus

PDK does not compete with any of these — it sits on top. Python pdk uses substrate-interface as its RPC layer; pdk-ts uses @polkadot/api (migrating to PAPI at alpha.5 after benchmark). What PDK adds is:

1. The CLI layer with 14 commands scoped to the dev loop
2. The FailLens decoder that speaks named errors + fix steps
3. The community-owned knowledge base

None of the underlying SDKs ship these, and none is Portaldot-specific.

7. Security model — what PDK trusts, what it verifies

- **Trusted:** the chain's runtime metadata. Every error name, pallet index, and dispatch type comes from a direct WebSocket call.
- **Trusted:** the human-authored KB in error_fixes.yaml1. Every fix entry is code-reviewed before merge.
- **Not trusted:** AI-suggested diagnoses. The optional AI layer (pdk-ts ai-setup at alpha.4) always renders alongside a yellow "UNVERIFIED" banner. If the KB has a verified fix for the error, the AI panel is suppressed.
- **Not trusted:** third-party mirror RPCs. All node endpoints are explicit — the CLI never falls back silently to a public RPC.
- **Signing paths:** all local. Keys never leave the CLI process. At v1.0 an external security audit will verify the signing paths after they land in pdk-ts α.3.

8. Case studies

Day-1 metadata spike (2026-05-19). A two-hour walk through the runtime metadata to test one assumption: every error has a name on-chain. The instant `Module 6, error 2 → Bal-`

ances.InsufficientBalance resolved programmatically, the whole product clicked. Every command that followed was derived from that single proof.

Upstream signing bug (polkascan/py-substrate-interface#9). Mid- hackathon, extending PDK to the Assets pallet revealed that Python substrate-interface mis-signs custom-pallet calls on Portaldot's V13 metadata. Half a day debugging, then filed upstream. This is the proximate reason pdk-ts exists as a TypeScript companion: the Python SDK's coverage of V13 custom-pallet signing is upstream-blocked.

Repo rename recovery. Two hours mid-event when the naming convention was announced. Three integrations (PyPI Trusted Publishing OIDC, Vercel webhook, GitHub release flow) broke. Switching `release.yml` to token auth made the pipeline rename-proof going forward.

9. References + benchmarks

- Repository: <https://github.com/PugarHuda/portaldot-hackathon-2026-pdk-AmpunBang>
 - PyPI: <https://pypi.org/project/portaldot-pdk/>
 - pdk-ts (TypeScript): <https://github.com/PugarHuda/portaldot-hackathon-2026-pdk-AmpunBang/tree/master/pdk-ts>
 - Upstream issue filed: <https://github.com/polkascan/py-substrate-interface/issues/9>
 - Portaldot developer docs: <https://portaldot-dev.readthedocs.io/>
 - PAPI (recommended next-gen SDK): <https://polkadotecosystem.com/tools/dev/papi/>
 - OSS governance model (BDFL → steering council, PEP 8016): <https://peps.python.org/pep-8016/>
 - Foundry brand + governance reference: <https://github.com/foundry-rs/foundry>
-

Appendix A — Command reference (pdk 0.1.6 + pdk-ts α.1)

Full CLI reference auto-generated from Typer (Python) and commander (TypeScript) will live at `portaldot-pdk.vercel.app/docs` after v0.2.0.

Appendix B — KB schema

See `pdk/data/error_fixes.yaml` for the current 29-entry knowledge base and `pdk/data/error_index.json` for the 202-entry verified runtime index against `portaldot-1002`.

This is a draft. Corrections, disagreements, and expansions welcome — open an issue on the repo or PR against `docs/pdk-whitepaper-draft.md`.