Le-Space

SPIN-OFF E · PROTOTYP

Relay Button on-chain

Relay-Deployments per Smart Contract — autorisiert, abgerechnet und gedeckelt

> Kein Konto, keine Rechnung, kein Vorstrecken: Der Vertrag hält das Guthaben, gibt den Start frei und rechnet beim Verbrauch ab.

Doku | Quellcode | npm

Das Problem

Der Relay Button startet Infrastruktur auf Knopfdruck. Bezahlt wird sie bisher wie überall sonst — und daran hängt alles Übrige.

- **Das Guthaben liegt beim Anbieter**

  Ein Konto, eine Aufladung, ein Vertrauensverhältnis. Genau die Abhängigkeit, die der restliche Stack vermeidet.

- **Wer für andere deployt, streckt vor**

  Eine App, die ihren Nutzern den Relay-Button anbietet, zahlt erst und rechnet danach ab — manuell, mit Ausfallrisiko.

- **Ein Klick kostet Geld — ohne Deckel**

  Ein eingebetteter Startknopf ohne Guthabenprüfung ist eine offene Rechnung für jeden, der ihn findet.

- **Keine Beteiligung an der Nutzung**

  Der Baukasten ist Open Source und wird eingesetzt — aber es gibt keinen Weg, an einem gestarteten Node etwas zu verdienen.

Die Lösung: ein Prepaid-Vault

Ein Vertrag hält das Guthaben und gibt es schrittweise frei — in vier Aufrufen.

- **Einzahlen** HEUTE

  `approve` und `deposit` mit einem ERC-20-Token. Das Guthaben bleibt beim Nutzer — wir halten es nicht.

- **Reservieren** HEUTE

  `reserve(intentHash, amount, expiresAt)` — gesperrt für genau dieses Deployment, mit Ablaufdatum.

- **Verbrauchen** HEUTE

  `consume(intentHash, amount)` nach dem Start. Abgerechnet wird, was der Node wirklich gekostet hat.

- **Zurückholen** HEUTE

  Läuft das Deployment nicht an, gibt `refund(intentHash)` die Reservierung frei. Niemand muss darum bitten.

> Kein Entwurf: Der Client liegt als `@le-space/browser` im Relay-Button-Monorepo — samt Guthaben-Abfragen, für Base, Avalanche und Ethereum.

RELAY BUTTON ON-CHAIN

Mehr als bezahlen: autorisieren

Der eigentliche Gewinn ist nicht die Zahlung, sondern dass der Vertrag entscheidet, ob ein Node überhaupt starten darf.

- **Geld ist an eine Absicht gebunden**

  Der `intentHash` verknüpft die Reservierung mit genau einem Deployment. Guthaben lässt sich nicht für etwas anderes ausgeben.

- **Der Vertrag ist der Schiedsrichter**

  Keine Reservierung, kein Start. Die Freigabe hängt nicht mehr an einem Backend, das jemand betreibt und abschalten kann.

- **Der Deckel ist eingebaut**

  Ausgegeben werden kann nur, was eingezahlt wurde. Ein eingebetteter Startknopf kann niemanden mehr überraschen.

- **Jeder Start hat einen Beleg**

  Reservierung und Verbrauch sind Transaktionen. Wer wann was gestartet hat, ist prüfbar, ohne dass jemand Buch führt.

RELAY BUTTON ON-CHAIN

Womit wir verdienen

Eine Gebühr pro gestartetem Node — eingezogen im selben Schritt, in dem abgerechnet wird.

- **Gebühr beim Verbrauch**

  Der Vault behält beim `consume` einen Anteil ein. Kein Inkasso, keine Rechnung, kein Zahlungsausfall.

- **Skaliert mit Nutzung, nicht mit Vertrieb**

  Jeder Node, den irgendjemand irgendwo über den Baukasten startet, trägt bei — ohne dass wir ihn kennen müssen.

- **Nicht-verwahrend**

  Das Guthaben gehört bis zum Verbrauch dem Nutzer. Wir halten keine Kundengelder — dieselbe Linie wie bei den Spin-Offs B und C.

- **Öffnet den Baukasten für Dritte**

  Wer den Relay Button einbettet, muss nichts mehr vorstrecken. Das ist erst der Grund, warum ihn jemand einbettet.

RELAY BUTTON ON-CHAIN

Wer damit startet

Überall dort, wo jemand Infrastruktur braucht, aber kein Verhältnis zu einem Anbieter aufbauen will.

App-Hersteller

– Betten den Relay Button in ihre App ein
– Nutzer zahlen selbst, die App streckt nichts vor
– Kein Abrechnungssystem nötig

Communities & DAOs

– Gemeinsame Kasse für ein gemeinsames Relay
– Sichtbar, wer wie lange finanziert hat
– Läuft weiter, solange jemand nachlegt

Agenturen & Wiederverkäufer

– Deployen für Kunden, ohne in Vorleistung zu gehen
– Der Kunde füllt seinen eigenen Vault
– Passt zum Beratungsgeschäft aus Spin-Off D

RELAY BUTTON ON-CHAIN

Stand & nächste Schritte

Die Client-Seite existiert und ist veröffentlicht. Was fehlt, ist der Vertrag dahinter — und die Gebühr darin.

- **Vault-Client veröffentlicht** HEUTE

  Einzahlen, Reservieren, Verbrauchen, Zurückholen plus Guthaben- und Reservierungsabfragen — in `@le-space/browser`, für drei EVM-Ketten.

- **Den Vertrag selbst veröffentlichen** IN ARBEIT

  Quellcode, Deployment-Adressen und ein Audit. Ohne das ist der Client ein Schlüssel ohne Schloss.

- **Gebührenlogik einbauen** IN ARBEIT

  Der Anteil pro gestartetem Node gehört in den Vertrag, nicht in eine Rechnung. Höhe und Empfänger konfigurierbar.

- **An den Lebenszyklus koppeln** GEPLANT

  Laufzeit verlängern, Node stoppen, Reste zurück — und derselbe Vault für weitere Anbieter neben Aleph.

Offene Fragen

Vier Punkte, die vor dem ersten echten Guthaben geklärt sein müssen.

- **Wer hält das Guthaben**

  Nicht-verwahrend zu bleiben ist die ganze Voraussetzung. Der Vertrag muss so gebaut sein, dass wir das Geld eines Nutzers nie bewegen können — auch nicht versehentlich.

- **Kurs und Kosten laufen auseinander**

  Ein Node kostet in Euro, bezahlt wird in Token. Zwischen Reservierung und Verbrauch kann sich das verschieben — Stablecoin oder Puffer?

- **Gaskosten pro Start**

  Reservieren und Verbrauchen sind zwei Transaktionen. Auf einer L2 ist das vernachlässigbar, auf Ethereum nicht — die Kette ist eine Produktentscheidung.

- **Wer haftet für den Node**

  Wenn ein Vertrag statt eines Menschen den Start freigibt: Wer ist Betreiber dessen, was darauf läuft? Das gehört geklärt, bevor es jemand fragt.

RELAY BUTTON ON-CHAIN

Gespräch?

Gesucht: eine App, die den Relay Button einbetten will, ohne in Vorleistung zu gehen — und jemand, der den Vault-Vertrag mit uns fertigstellt und prüft.

Le Space UG (haftungsbeschränkt) · Nico Krause

kontakt@le-space.de · local-first.le-space.de

+49 / 87 21 / 5 06 49 96

Doku **Quellcode** **npm**