trustware

Client-Owned Gas Sponsorship

Trustware’s Non-Custodial Paymaster System

August 2026

Harrison Lassige
trustware.io

Client-Owned Gas Sponsorship: Trustware's Non-Custodial Paymaster System

Trustware · August 2026

Every sponsored transaction is backed by someone's money. Who owns that money, and who can move it, is the part most products give up by default.

At a glance:

- **Gas sponsorship is ultimately an ownership question.** Every sponsored transaction is backed by real money, and the market default is that the provider controls the paymaster and custodies the funds behind it.
- **Trustware makes the paymaster client-owned.** Each client deploys and owns its own onchain paymaster. Sponsorship funds sit in a contract only the client controls, and Trustware cannot move, withdraw, or redirect them.
- **No other productized model offers this combination.** A client-owned, non-custodial, app-sponsorship paymaster, deployed per client and wired into a routing and settlement layer, is not something the market has productized before.
- **Better UX and non-custody compound**. Touchless on the surface, trustless underneath.

Gas sponsorship is often framed as a UX problem: hide the gas, clean up the flow, let the user transact without holding the native token. That framing is useful, but incomplete. Sponsoring gas means spending real money on someone else's transaction and that money has to be held, funded, and governed by someone.

When an application sponsors gas, something has to hold the funds and enforce the rules. Someone funds the paymaster, someone decides which transactions qualify, how much can be spent, what happens when usage spikes, and who can pull unused funds out. In effectively every productized model today, the provider runs the paymaster, owns the funding or billing relationship, and hands the application a dashboard and an API on top of infrastructure it does not own.

Trustware Paymasters give each client its own account-abstraction-compatible paymaster, deployed and owned by the client. The sponsorship funds sit in a client-controlled onchain contract tied to the wallet that deployed it. Trustware provides the deployment flow, policy and dashboard layer, sponsorship authorization, and the routing and settlement integration. The client keeps the onchain sponsorship layer and the funds inside. The client owns the paymaster and the client controls the funds. Trustware cannot move, withdraw, or redirect them.

This resets the trust model for gas sponsorship, and it is the same model the rest of Trustware runs on. Trustware is the infrastructure applications use to complete onchain transactions end to end: it accepts any asset from any chain and any wallet, routes through the best available liquidity, and settles into the asset, chain, and destination the application needs, without ever taking custody. The paymaster brings that logic to gas by funding the step that gets a transaction confirmed, and the application owns it.

The problem with provider-managed gas sponsorship

Most new users come to digital assets with the mental model of a card, where fees are invisible and no one holds a separate asset just to pay them. Gas breaks that expectation, for the user and for the business accepting the payment.

Gas is a property of how public blockchains are built. Every computation and state change a network performs has to be paid for in that chain's native token, which is what compensates the parties that secure it. So a user holding only stablecoins on a given chain still cannot transact until they acquire its gas token, and that structural fact is the root of the friction gas sponsorship exists to remove. Application teams want gas sponsorship to take that friction off the user. A user should not have to leave the product, go acquire a native token, come back, and try again before completing what they came to do. Sponsoring that gas can lift onboarding, deposits, in-product flows, and completion rates.

The demand is already proven: centralized exchanges hide gas entirely, and that invisibility onboarded most of crypto. The open problem is delivering it onchain without a custodian. Provider-managed gas sponsorship gives the application a clean API and dashboard controls, but the ownership sits somewhere else. The application funds a provider-managed system and configures rules offchain, while depending on the provider's paymaster to actually service the transactions. It also tends to sit beside the application's stack as a separate product, not something that plugs into the routing, settlement, and keys already in place. This works for simple use cases, but is a real problem for any application or institution that cares about custody, controls, reconciliation, and operational accountability.

For those teams, the specifics matter: Who owns the sponsorship funds, and where do they actually sit? Who can move or withdraw them? Can the application verify the balance onchain, keep separate controls per project and per chain, and tie sponsorship rules to its own API keys and transaction flows? And is any of that connected to routing and settlement, or is sponsorship just bolted on?

Trustware’s paymaster model

Trustware Paymasters let a client deploy and manage its own paymaster contracts through Trustware infrastructure. The current system is live on Avalanche and Base, with additional chains rolling out. Under the hood it uses standard ERC-4337 paymaster contracts on EntryPoint v0.7, with an offchain signer that authorizes which operations get sponsored.

At the product level the model is straightforward:

1. Clients deploy their own paymaster through Trustware’s custom factory and client-owned paymaster contracts.
2. The deploying wallet or address becomes the sole owner/controller of the paymaster.
3. Sponsorship funds stay in a client-controlled onchain contract, held in the paymaster's own EntryPoint deposit. That deposit is the gas-paying balance and is withdrawable at any time, distinct from the anti-DoS stake, which carries an unstake delay.
4. The owner controls funding and withdrawals (can withdraw balance at any time).
5. Upgrades are owner initiated and move the deployment to the new Trustware-published contract versions.

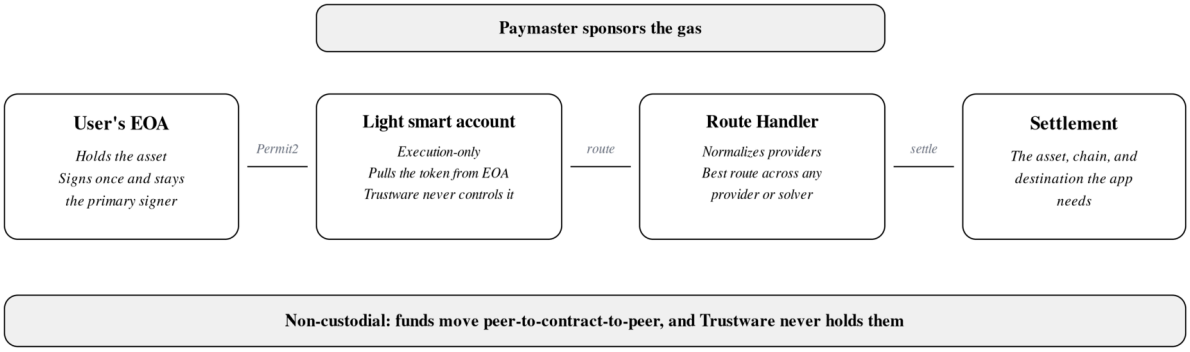
6. Trustware provides the factory, dashboard, API-key scope rules, sponsorship authorization flow, plus routing and settlement integration.
7. Trustware cannot move, withdraw, or redirect client sponsorship funds.

Trustware clients do not build their own paymaster contract systems, policy engine, dashboard, activity tracking, or gas accounting from scratch. Trustware authorizes which sponsored UserOperations are eligible under the client's rules. Ultimately, the control model is the product.

Under the ERC-4337 model, gas sponsorship applies only to UserOps, and those originate from smart accounts, not EOAs, whose transactions never touch the EntryPoint. EIP-7702 offers one way around this by letting an EOA delegate to account code and act like a smart account. Trustware took a different path: it deploys deterministic light smart accounts for the wallets that interact with the paymaster. That keeps the EOA as a pure signer and the smart account as an isolated, execution-only vehicle, and it gives every user a predictable account address without altering the EOA itself. Without this account layer, the paymaster could not sponsor a transaction through the Route Handler at all.

These accounts are execution-only: the user's EOA stays primary and remains the signer, and Trustware never controls the smart account. As shown in Exhibit 1, the user signs a Permit2 authorization with their EOA key, which lets the smart account pull the token from the EOA, and the account then executes the route with the paymaster covering the gas. The user holds their funds the whole time and, after the one-time Permit2 authorization, signs once per route.

Exhibit 1: Flow of funds, sponsored and non-custodial



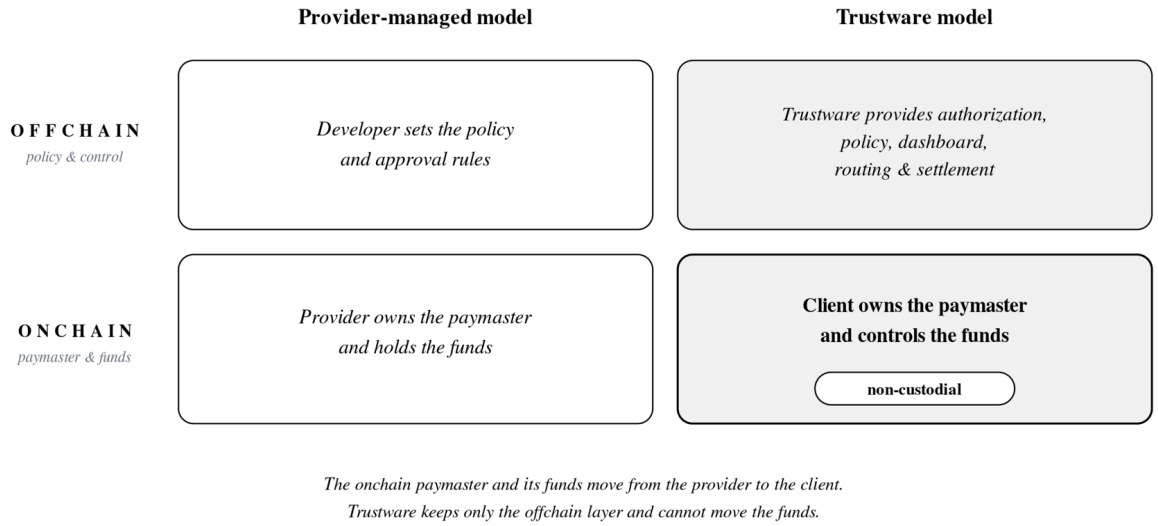
The offchain/onchain inversion

The split that matters is offchain policy versus onchain control. In most gas-sponsorship products, the developer controls the policy or approval logic offchain while the provider keeps the onchain paymaster and the funding model. As shown in Exhibit 2, Trustware took the opposite approach.

The client owns the actual onchain contract and the sponsorship funds. Trustware provides the offchain authorization, the policy controls, the API integration, dashboard, and the routing and settlement context. The client configures how gas gets sponsored without handing the sponsorship layer to a provider-managed balance.

The deploying address is the contract's owner, with funding and withdrawal running through that owner. Upgrades are owner-initiated, and whatever is not spent stays in the client's contract. For teams that actually care about custody and operational control, that is a different trust model.

Exhibit 2: The offchain/onchain inversion



The onchain paymaster and its funds move from the provider to the client. Trustware keeps only the offchain layer and cannot move the funds.

What clients control

Trustware Paymasters are built for applications that need sponsorship to be programmable, accountable, and tied to real product flows.

From the Trustware dashboard, a client manages:

- Paymaster deployment
- Top-ups and withdrawals
- Contract upgrades
- Monthly budgets
- API-key scope rules
- Maximum uses per sender
- Maximum cost per transaction
- Rules priority
- Paymaster activity and usage tracking

Sponsorship gets expensive or abusable fast if it is not scoped well. A serious system has to answer which actions get sponsored, which users or API keys qualify, how much spend is allowed, and how usage is tracked across projects and chains. Trustware gives clients those controls without taking ownership of the onchain sponsorship layer underneath them.

What this gives applications

Applications want users to complete actions, the right asset to land in the right place, and enough visibility, reconciliation, and guardrails to run it in production. They want better UX without taking on a new custody problem, and they do not want to build paymasters.

Using Trustware Paymasters, users complete supported actions without a separate native token: gas is sponsored, or drawn from the asset they are already moving. The application controls its own sponsorship budgets and rules: sponsorship tied to API keys, projects, senders, costs, and usage limits. As of today, paymasters are deployed and run per chain and per project; applications can have multiple projects in production.

Trustware's non-custodial settlement and routing infrastructure already enables any asset/chain transaction flows. The paymaster was built to plug directly into Trustware's Route Handler. Route providers share no common standard, so the Route Handler normalizes their endpoints. That lets Trustware run one user intent across several providers at once and take the best route returned. Because the paymaster sits behind that layer, it works with any route the Route Handler produces, from any provider or solver network. The client defines the destination state, the user brings what they have, and Trustware handles the routing, settlement, and sponsored completion between those two points without ever taking custody.

From UX to infrastructure layer

Gas sponsorship started as a way to hide gas from users, and that job still matters; but once an application takes sponsorship seriously, the paymaster becomes a control surface.

When do we sponsor? What do we sponsor? How much? How do we track it? Who holds the funds behind the experience, and who can move them?

Answer those questions with a provider-managed balance and the application has handed a critical part of its transaction path to someone else. Answer them with an owned contract and the application keeps it: that is the move Trustware is building.

A paymaster becomes infrastructure when it sits inside the application's operating layer:

- it shapes conversion and onboarding
- it governs sponsored-transaction economics
- it produces usage and policy data the application owns
- it connects to routing, settlement, and reconciliation
- it decides how much control the application keeps over the part of the flow that determines whether a transaction completes

This is why the paymaster matters beyond gas UX and why Trustware is building at the application completion boundary. Deposit, routing, settlement, gas sponsorship, policy, status tracking, and reconciliation all point at the same problem: applications need infrastructure that turns user-controlled value and intent into application-ready outcomes without adding a new custodial counterparty.

Trustware’s approach

It is simpler to run one provider-managed paymaster for every customer and bill against it than to give every client its own deployed, owned, upgradeable paymaster contract, with funds onchain under the client's sole control, then wrap that in a dashboard, a policy engine, and routing and settlement integration. It is also the version application teams actually want once sponsorship becomes material to their economics.

Across the paymaster-as-a-service providers there is not another productized app-sponsorship model that gives every client a client-owned paymaster contract with sponsorship funds held onchain under the client's sole control. That claim is intentionally precise. It does not claim that no one can deploy a paymaster contract. Client-deployable paymaster contracts exist in certain user-pays ERC-20 setups and custom infrastructure builds. What has not been offered as a product is the combination:

Paymasters deployed per client through a client-owned onchain factory contract with:

- Sponsorship funds under the client's sole control
- Full owner-gated funding, withdrawal, and upgrades
- Policy and dashboard controls built for application teams
- All of it integrated into a broader routing and settlement layer

The market has seen pieces of this before: paymaster contracts, gas sponsorship, account abstraction, and dashboard-managed policies. What has not been productized is the full system: a client-owned, non-custodial paymaster that applications can deploy, fund, control, and operate directly. With Trustware, the infrastructure that sponsors gas is no longer controlled by the provider: it is owned by the application.

Security and control posture

Paymasters sit on a sensitive boundary as they authorize gas spend, hold sponsorship balances and decide which transactions an application is willing to subsidize. An application paying for gas does not want to underwrite a transaction tied to sanctioned or tainted funds. Trustware screens for sanctions and AML at the routing layer, before a route is quoted or executed, so funds that fail screening never get a route in the first place. Sponsorship rides on Trustware's routing, so a client is never put in the position of paying gas to advance flagged funds.

The system is built around client-controlled contracts, scoped policies, usage limits, activity visibility, and extensive internal reviews across contract safety, usage behavior, and upgrade controls. The control model is the core security property. Trustware’s role is to make the client-owned sponsorship usable: deployment, authorization, policy, dashboard management and the connection into routing and settlement flows.

The broader Trustware thesis

Trustware is building the Universal Deposit Layer for onchain and offchain applications. Any application should be able to accept any value from users across assets, chains, and wallets, then receive the destination state they need without rebuilding routing, settlement, gas sponsorship, and reconciliation themselves.

The status quo for accepting value onchain is the deposit address: the application hands the user an address and waits for funds. That assumes the user already holds the right asset on the right chain and the native gas to send it, so when they have the asset but not the gas, the deposit stalls or the funds sit at an address they cannot act from. Without sponsorship, the flow breaks at the exact moment an application is trying to onboard someone, and a route is not enough if the user stalls before they can submit it. But a settlement layer is worth more when it can decide, per operation, whether sponsoring gas improves completion, and act on that without ever holding the client's funds. Trustware makes gas sponsorship client-owned, policy-driven, and connected to the rest of the completion layer. Sponsorship stops being a provider-managed black box or a UX patch, and becomes part of an infrastructure stack the application actually controls.

Trustware's thesis is that application-owned infrastructure wins, because the teams closest to the user should control the systems that move, settle, and sponsor value on their behalf. That is the stack Trustware is building: deposits, settlement, routing, and now sponsorship, all designed around the same principle. Trustware believes applications should own their infrastructure and a user should keep control of their funds. Providers should enable the system, and not become the system.

Crypto has long been split between trustless and touchless, so most products pick one: exchanges are touchless but custodial, self-custody is trustless but makes the user touch everything. Trustware is built so applications do not have to choose. The paymaster is the proof: sponsored, abstracted gas on a contract the application owns, with funds Trustware cannot move. Touchless on the surface, trustless underneath.

Please visit docs.trustware.io for more information on the Trustware API and SDK.