CoRelayer

COOPERATIVE RELAYER INFRASTRUCTURE FOR AUTONOMOUS TRANSACTIONS

CoRelayer Whitepaper

Relayed transactions on MultiversX,
priced per transaction and paid in USDC

REVISION	DATE
1.0	2 November 2026

ABOUT THIS DOCUMENT

CoRelayer

Relayed transactions on MultiversX, priced per transaction and paid in USDC

DOCUMENT	REVISION	DATE
Whitepaper	1.0	2 November 2026

AUTHOR	CONTACT	WEBSITE
CoRelayer	support@co-relayer.com	co-relayer.com

ABSTRACT

Every MultiversX transaction pays a network fee in EGLD, so a new user must hold EGLD before a first action. CoRelayer pays that fee through relayed transactions, a feature of the MultiversX protocol. The user signs once, in an existing wallet; a relayer in the sender's shard adds a second signature and pays the fee. Customers buy plans in USDC from the CoRelayer contract, priced per transaction and never per user. One price per transaction, set on chain, determines every price; a rise takes effect 48 hours after it is announced. A relayer cannot alter a signed transaction, cannot send it twice and cannot move any balance, because MultiversX's own rules prevent each. A relayer can delay or decline, so every relayer is listed on chain and can be checked before signing.

NOTICE

This paper describes CoRelayer as it operates on MultiversX. Prices and limits are set on chain and published at co-relayer.com; where they differ from this paper, the chain is authoritative. Built on MultiversX. CoRelayer is an independent project and is not affiliated with, endorsed by, or a product of MultiversX. "Finance Computer" is a name used by its owner.

WHITEPAPER

Contents

01 Introduction 04

02 Relayed transactions on MultiversX 06

03 Delivery 08

04 Sponsorship 11

05 Security model 12

06 Plans and prices 14

07 AI agents 18

08 Payments 20

09 Operations 22

10 Growth 23

11 Summary 26

12 Glossary 27

13 References 29

01

Introduction

Every MultiversX transaction pays a network fee in EGLD, so a new user must hold EGLD before a first action. CoRelayer pays that fee. The user signs once, in an existing wallet; the customer pays for a plan in USDC.

1.1 The problem

The **network fee** is the fee MultiversX charges for every transaction, paid in EGLD, the network's native token. A user who holds only USDC cannot send USDC without first acquiring EGLD. For a new user of an application, that means buying EGLD, moving it and waiting before the first action. Each of those steps loses users.

Software that acts on its own faces the same requirement in a harder form. A gas balance is an operational account: it must be monitored and refilled, it is exposed to key compromise, its required size moves with the EGLD price, and an empty balance stops an agent without a visible error.

1.2 The service

A **relayer**, a wallet CoRelayer operates, pays the network fee of each transaction. The user still signs the transaction, it is still sent from the user's address, and it moves only what the user approved. Only the payer of the network fee changes. MultiversX supports this natively (chapter 2).

Customers pay with a **plan**: a number of transactions every 30 days, for any number of users, paid in USDC. The rules are held on chain in the **CoRelayer contract**, a smart contract that stores the prices, each account's purchases, the relayers permitted to sign and the split of each payment.

The name abbreviates the full description: Cooperative Relayer Infrastructure for Autonomous Transactions. "Cooperative" refers to many relayers across every MultiversX shard working under one set of on-chain rules; opening that set to independent operators is phase 3 of the roadmap (chapter 10).

CoRelayer holds no customer or user funds and no private keys of either. It is not a wallet, a bridge, an exchange or a chain, and it cannot send anything MultiversX would refuse.

1.3 Customers

CoRelayer serves application and product teams, and AI agents. Table 1 lists typical uses.

USE	ROLE OF CORELAYER
Games and mints	Players mint, trade and claim from their first session.
Payments and payouts	Stablecoin payments and payouts from named wallets, with no gas balance.
Onboarding	New users act at once, within a daily limit per user.
Wallets	Network fees covered for the wallet's accounts, on selected contracts.
AI agents	Agents pay in USDC; the plan covers the network fee.
Fleets and devices	One account and one bill for many addresses.

Table 1 Typical uses of CoRelayer.
co-relayer.com

In a machine-to-machine financial stack, CoRelayer is the delivery layer: above the ledger and below the applications, agents and wallets that use it (Figure 1). It is built for the Finance Computer.[1]

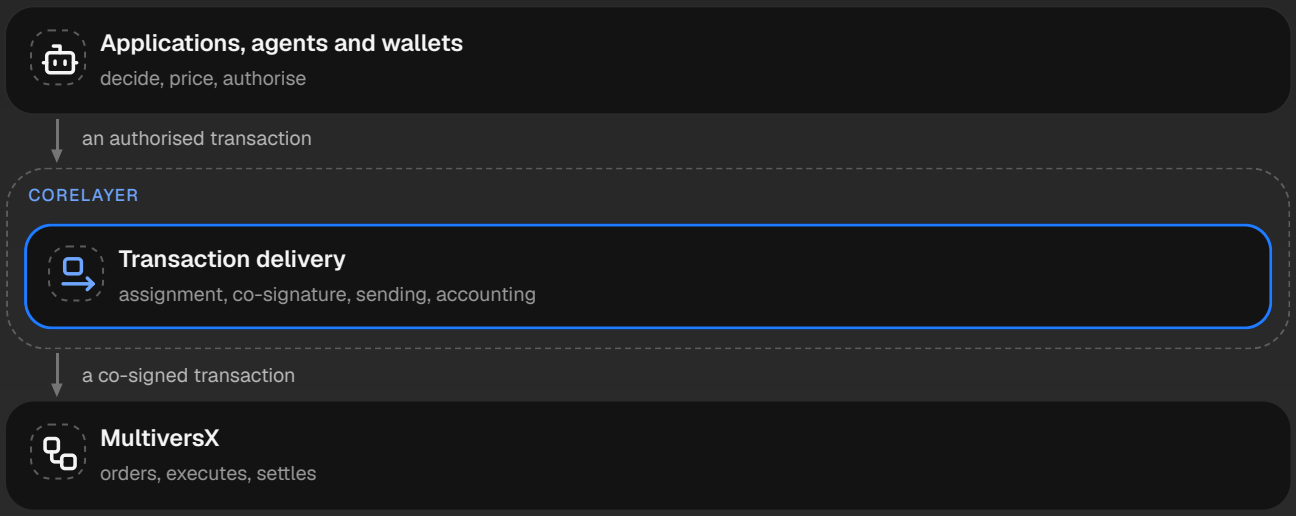


Figure 1 Transaction delivery sits above the ledger and below everything that uses it.
CoRelayer

Chapters 2 to 4 describe the mechanism, chapters 5 to 9 the security model, prices, agents, payments and operations, and chapter 10 the growth path. No price is typed into this paper: chapter 6 prints the plans from the published pricing data. Terms in bold are defined at first use and collected in the glossary.

1 "Finance Computer" is a name used by its owner.

02

Relayed transactions on MultiversX

CoRelayer uses a feature of the MultiversX protocol. This chapter states the protocol facts the rest of the paper relies on; MultiversX's documentation is their source [1].

2.1 Two additional fields

Relayed transactions v3 (relayed v3) add two fields to an ordinary transaction: the relayer's address and the relayer's signature (Table 2). A transaction that carries them is a **relayed transaction**; the account that signs and pays the network fee is its relayer.

FIELD	CONTENT
`relayer`	The address of the account that pays the network fee
`relayerSignature`	That account's signature over the bytes the sender signed

Table 2 The two fields of a relayed transaction, as MultiversX names them.
MultiversX documentation: relayed transactions [1]

Both signatures cover the same bytes, and those bytes include the relayer's address. The relayer therefore cannot be replaced after the sender signs: a different address invalidates the sender's signature and the network refuses the transaction. The sender knows which account pays before approving.

2.2 Shards, nonces and failures

MultiversX divides its accounts among three **shards** that process transactions in parallel, coordinated by a metachain. A relayer must be in its sender's shard [1]. CoRelayer keeps relayers in every shard and assigns each transaction a relayer in its sender's shard.

Every transaction carries a **nonce**, the sender's transaction counter; the network accepts each value once. A relayed transaction consumes only the sender's nonce, which lets one relayer serve many senders at once. A signed transaction cannot execute twice: once its nonce is used, any copy is refused.

A relayed transaction has no expiry of its own. Once co-signed, it stays valid until the sender's nonce moves past it. Without the relayer's signature it cannot run.

A relayed transaction that runs and fails still costs a fee: MultiversX charges the relayer the fee of the full gas limit. A transaction that never runs, for example because its nonce was already used, costs nothing.

FACT	VALUE
Shards	Three, coordinated by a metachain
The relayer's shard	The sender's shard
Relayed transactions v3	Enabled at epoch 1671
Relayed transactions v1 and v2	Disabled at epoch 1918, on 30 October 2025; v3 is the only relayed format
Extra gas for a relayed transaction	50,000
Nonces used	The sender's only
A failed execution	The relayer pays the fee of the whole gas limit
A transaction that never runs	No fee; the nonce does not move
A copy of a used nonce	Refused

Table 3 The MultiversX facts this paper relies on.
MultiversX documentation: relayed transactions [1]

The CoRelayer contract reads time from MultiversX blocks, in milliseconds.

03

Delivery

A transaction passes through five steps, from the plan that pays for it to the block that records it.

01 **Purchase.** The customer pays USDC to the CoRelayer contract, which converts the payment into **credits**, a balance it keeps for the account, and records the purchase.

02 **Assignment.** The customer's server requests a relayer. CoRelayer assigns a healthy, funded relayer in the sender's shard, for a short period.

03 **Signature.** The user signs once. The relayer's address is among the signed fields and cannot be replaced.

04 **Co-signature.** The relayer signs the same bytes, and the transaction is sent to the network.

05 **Accounting.** The transaction's worst-case fee is converted into transactions of the plan and billed when it executes.

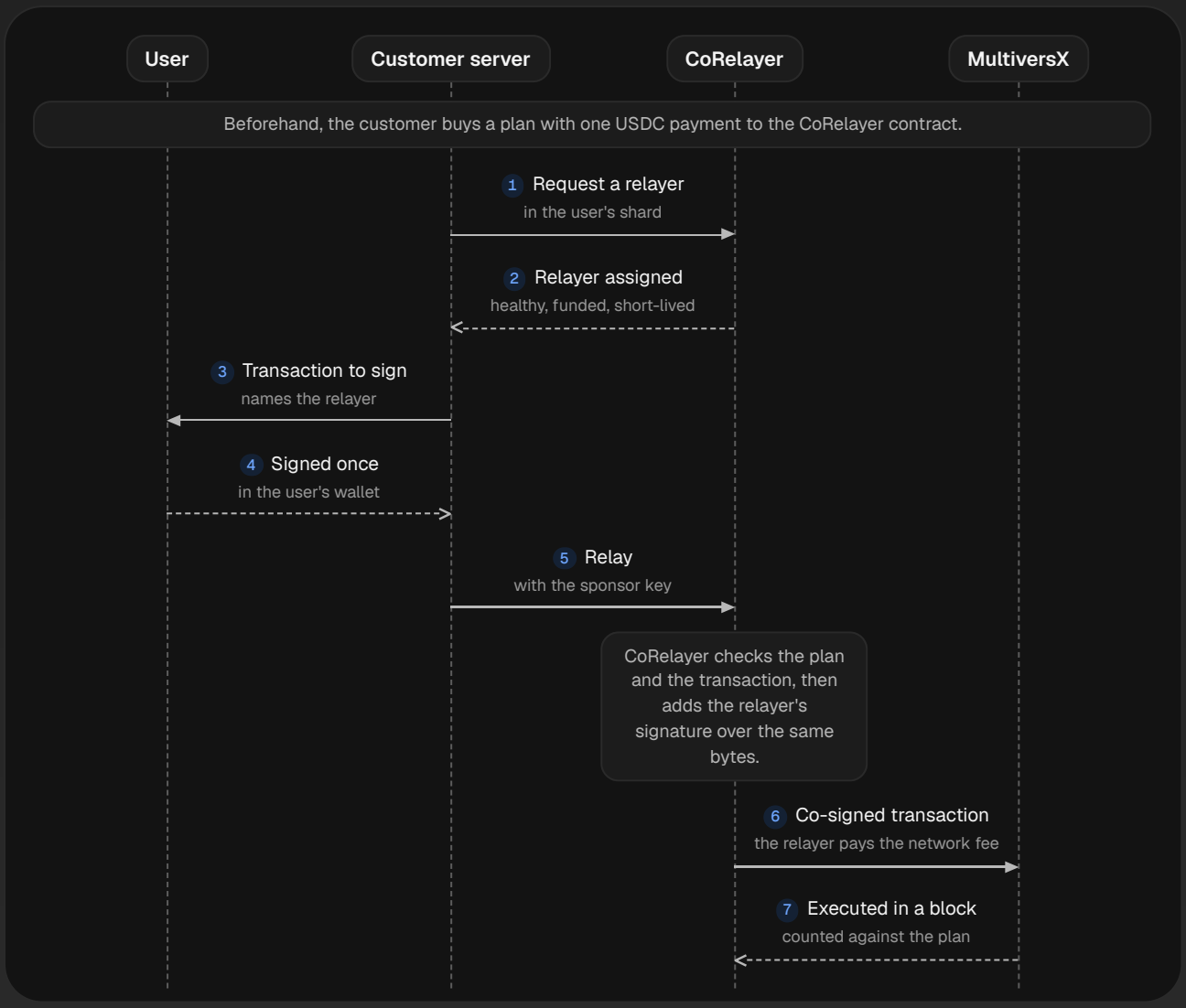


Figure 2 One transaction, from the request for a relayer to the block that records it. The customer's server holds the sponsor key; the user holds the signing key.

CoRelayer documentation: delivery [2]

Buying a plan costs no network fee: calls to the CoRelayer contract, such as purchases and plan settings, are relayed by CoRelayer itself, so a wallet holding only USDC can buy and manage a plan. While an account has no plan to pay with, such a call costs nothing; once the account can pay, it counts against the plan like any other transaction. Each named wallet beyond the ones a plan includes carries a small one-time fee.

3.1 Failures

The relayer's signature is the boundary

Before the relayer's signature exists, nothing can run, and an error means nothing was sent. After it, the transaction remains valid until its nonce is used, so its outcome is reported as a state, not as a failure.

Sending identical bytes again is the same transaction and is safe. An optional key per user action prevents a client from signing a second transaction for one action. Table 4 lists the three failures that can occur after the user signs.

FAILURE	HANDLING
An access point to the network does not answer	The identical bytes are sent through another. Nothing is rebuilt or re-signed.
The relayer's EGLD runs low	Relayers are kept above a floor and refilled, so the signed transaction proceeds.
The relayer is lost before it signs	The user signs again on the same nonce, so only one of the two transactions can run.

Table 4 Three failures, none of which sends a transaction twice.

CoRelayer documentation: delivery [2]

A transaction that runs and fails is still charged by the network; CoRelayer pays the fee and the transaction counts against the plan. A transaction that never runs is not counted. The dashboard shows each transaction's timeline, where it stopped and why.

04

Sponsorship

A plan pays either for every user of an application or for a set of named wallets. It limits the number of transactions, not the number of users.

A **sponsor key**, held on the customer's server, makes the plan pay for transactions signed by any wallet. Each key carries a required list of the contracts it covers, optional limits per user and per day, and can be revoked at any time; no key can create or revoke other keys. The key is shown once and never belongs in a web page. Sponsorship is available on Builder, Growth, Scale, Enterprise, Agent Pro and Agent Fleet.

A sponsor key only chooses who pays; it never proves that a user is present. The user does that when a relayer is requested, by signing a short message in their wallet that names the application's web origin; signing it moves no funds.

Named wallets are addresses recorded on chain against an account, so anyone can verify which addresses a plan pays for. Each plan includes a number of them: 3, 10, 50, 250 and 1,000 on the five plans for applications and teams, and 1, 25 and 1,000 on the three agent plans. Adding or replacing one within that number has no fee, nor has removing one. Each one beyond it carries a small one-time fee, not refunded, up to 10,000 per account.

	SPONSOR KEY	NAMED WALLETS
Pays for	Any wallet using the application	The listed wallets
Number	Unlimited; the plan is the limit	1 to 1,000 included, by plan; up to 10,000 in all
Requires	A customer server	Nothing further
Plans	Builder and above, Agent Pro, Agent Fleet	Every plan
To add	No fee	A small one-time fee beyond the plan's number

Table 5 The two forms of sponsorship.
co-relayer.com/product

Four limits bound what a plan can spend:

- **Price ceilings.** Every purchase and renewal carries a maximum price signed by the buyer. A renewal above it is skipped, not charged.
- **Contract lists.** A sponsor key pays only for calls to the contracts it lists.
- **Per-user limits.** A daily cap per user, and optionally per key.
- **Plan limit.** When the plan's transactions are used, **sending pauses**. Pay-as-you-go spends only from a budget set aside for it.

05

Security model

A service that pays network fees for users sits between them and the chain. The relevant question is what that service could do if it were not trustworthy [3].

PROPERTY	BASIS	VERIFICATION
A relayer cannot alter a signed transaction	The sender's signature covers every field, the relayer's address included; any change invalidates it	Both signatures over the same bytes, on the MultiversX Explorer
A transaction cannot execute twice	The network accepts each nonce once	The nonce in the signed bytes, on chain
A relayer cannot move any balance	Relayed transactions have no approval, allowance or custody mechanism; plan payments go to the contract	Each payment's split, an event of its own transaction
A relayer can delay or decline	It may sign late or refuse	The relayer is named before signing and can be checked against the on-chain relayer list

Table 6 What a relayer can and cannot do, and what each property rests on.
CoRelayer documentation: the trust model [3]

The relayer's signature authorises the fee payment and nothing else. A refusal occurs before anything is co-signed, so nothing was sent. The service reports which case applies, so a client never has to guess whether to wait, resend the same bytes or request a new signature.

Every CoRelayer relayer is listed on chain in the **relayer list**. A client can confirm, through MultiversX's public API, that the assigned relayer is on the list before the user signs [4]; the SDKs include this check. Clients should pin the contract address and the **chain id**, the network identifier in every signed transaction, and refuse a discovery file that contradicts them.

5.1 Keys

Three keys act on the contract:

- **The owner key**, held offline, sets prices and plans, registers relayers, resumes paused paths and upgrades the contract.
- **The operations key** can pause and can manage registered relayers. It cannot change prices, resume a path or register a relayer.
- **The reporting key** can only report usage, within the bounds of chapter 8.

Each money path has a **pause switch**: the operations key can pause it, and only the owner key resumes it. Purchases and settings, deposits, **settlement** (the periodic call that debits pay-as-you-go budgets), payments to relayers and the charging part of renewals can each be paused separately. The release of an unused pay-as-you-go budget can never be blocked.

An upgrade is announced at least 7 days before it is sent, with its source commit and the hash of its reproducible build, and the contract refuses an upgrade unless every money path is paused. Before an upgrade completes, the new code checks the contract's EGLD and WEGLD against its own ledgers, and nothing sent to the contract from outside can make that check fail. The deployed code is a **verified build** that anyone can rebuild and compare byte for byte. The contract's address and chain id are published at `co-relayer.com/.well-known/corelayer.json` [5].

The delivery rules are numbered invariants, each with a test. The counting rule of chapter 6 has published test vectors. The API types, references and error catalogue are generated from their sources, and a drift check fails the build.

RISK	BOUND
A compromised **signing server**, a server holding relayer keys	Exposes only the relayers' working funds, CoRelayer's own capital. No plan, credit or user token is reachable; spare relayers replace those affected.
A single owner key	Held offline. Price rises wait 48 hours; upgrades are announced and require every money path to be paused.
A leaked sponsor key	Its contract list, its daily limits and revocation.
A signed transaction left in a log	Inert without CoRelayer's co-signature; one first seen long ago is refused.

Table 7 Residual risks, and what bounds each.
CoRelayer documentation: the trust model [3]

Security reports go to support@co-relayer.com, under the disclosure policy at docs.co-relayer.com/security/disclosure and co-relayer.com/.well-known/security.txt.

06

Plans and prices

CoRelayer is priced per transaction, never per user. Every price follows from one USDC price per transaction, set on chain.

6.1 The price per transaction

The **price per transaction**, which the contract calls the tariff, converts each plan's price units into USDC. No price feed moves it. Only the owner key changes it, within limits compiled into the contract:

- It stays between $0.001 and $1 per transaction.
- One change at most doubles or halves it.
- A rise takes effect 48 hours after it is announced; a cut takes effect at once.

6.2 Counting

One transaction is counted for every 0.001 EGLD of network fee a transaction can reach at most, rounded up, and never fewer than one. The documentation and the API call this counting unit the **Relay Unit**.

$$n = \max(1, \operatorname{ceil}(F \;/\; 0.001\ \text{EGLD})) \qquad (1)$$

n	the transactions counted for one transaction sent
F	the highest network fee the transaction can reach, computed from its own fields before signing

The rule, its rounding and its test vectors are published [6]; a fingerprint of that document is stored in the contract and cannot change without 48 hours' notice. In a September 2026 sample of 1,000 relayed transactions on MultiversX, 88% counted as exactly one.

6.3 The plans

Five plans serve applications and teams, three serve AI agents, and custom plans cover larger volumes. Table 8 prints them from the published pricing data.

PLAN	PRICE PER 30 DAYS	TRANSACTIONS INCLUDED	PAYS FOR	PAY-AS-YOU-GO
FOR APPS AND TEAMS				
Starter Shared relayers	$10	1,000	You + 3 named wallets included	$0.013 each
Builder Shared relayers	$45	5,000	Sponsor every user + 10 named wallets included	$0.0125 each
Growth Priority lane	$200	25,000	Sponsor every user + 50 named wallets included	$0.012 each
Scale Priority lane Available on request	$700	100,000	Sponsor every user + 250 named wallets included	$0.0115 each
Enterprise Dedicated relayers Available on request	$3,000	500,000	Sponsor every user + 1,000 named wallets included	$0.011 each
FOR AI AGENTS				
Agent Metered Shared relayers, best effort	No monthly fee	Pay per transaction	Your agent + 1 named wallet included	$0.015 each
Agent Pro Shared relayers	$85	10,000	Sponsor any agent you run + 25 named wallets included	$0.012 each
Agent Fleet Priority lane Available on request	$650	100,000	Sponsor any agent you run + 1,000 named wallets included	$0.011 each

Prices in USDC, set on chain. Prices as of 19 September 2026.

Table 8 Every listed plan, from the pricing data. A plan available on request keeps its price.
co-relayer.com/pricing

Formula 2 gives the price of a plan and formula 3 the price of each transaction beyond those a plan includes (the pay-as-you-go column).

$$P = u \times m \times t \qquad (2)$$

- P the price of a plan, in USDC
- u the plan's price units, in the table of plan terms
- m the 30-day periods bought in one purchase, from 1 to 12
- t the price per transaction

$$x = f \times t \qquad (3)$$

- x the price of one transaction beyond those a plan includes
- f the plan's extra-transaction factor, in the same table

Each plan has a **relayer class** (shared relayers, a priority lane or dedicated relayers) and **speed limits**, the number of transactions an account may send in a given time, named from Entry to Highest (Table 9).

PLAN	AUDIENCE	SPEED LIMITS	PRICE UNITS	EXTRA-TRANSACTION FACTOR
Starter	Own wallets and first tests	Entry	1,000	1.30
Builder	Applications that pay for their users	Standard	4,500	1.25
Growth	Applications that need headroom	High	20,000	1.20
Scale	High-volume applications	Higher	70,000	1.15
Enterprise	The highest volume, on dedicated relayers	Highest	300,000	1.10
Agent Metered	A single agent	Entry	0	1.50
Agent Pro	Platforms running many agents	High	8,500	1.20
Agent Fleet	Fleets of agents and devices	Highest	65,000	1.10

Table 9 Audience, speed limits and the terms of formulas 2 and 3, per plan.
co-relayer.com/pricing

A custom plan covers volume above Enterprise or specific terms, and is priced per agreement through support@co-relayer.com.

6.4 Price commitments

Four rules apply to every plan:

01 **A fixed USDC price per transaction.** One number in the contract sets every price.

02 **Rises announced 48 hours ahead.** The delay is compiled into the contract, so the key that sets the price cannot shorten it.

03 **A paid period keeps its price.** A purchase fixes its terms, and later changes do not reach it.

04 **Payment in advance.** Plans are bought before use; nothing is billed afterwards.

6.5 Credits, renewals and pay-as-you-go

A payment becomes credits, counted from the USDC paid at the price in force, never from what the swap returned. Credits buy plans and extra transactions and are not refundable. A plan's included transactions expire at the end of each period.

Buying ahead. One purchase covers 1 to 12 periods at the price in force, without a prepayment discount.

Changing plans. An upgrade starts at once. Prepaid periods that have not started return to credits, which count towards the new plan's price, and the unused part of the current period, measured once its last transactions are billed, moves to the new plan once. A downgrade starts when the paid periods end.

Pay-as-you-go. When enabled, sending continues at the extra-transaction price, paid only from a **pay-as-you-go budget** set aside from credits. Disabling it returns the remainder once its last transactions are settled; if settlement stalls, anyone can release everything not held for unsettled usage after 7 days. A plan that ends without renewal pauses sending even with pay-as-you-go enabled.

Renewals. Automatic renewal and pay-as-you-go are off by default, each with a ceiling; a ceiling of zero never means unlimited. A renewal above its ceiling, or without enough credits, is skipped, and a late renewal never charges for time not served. Anyone may trigger a renewal for an account that opted in.

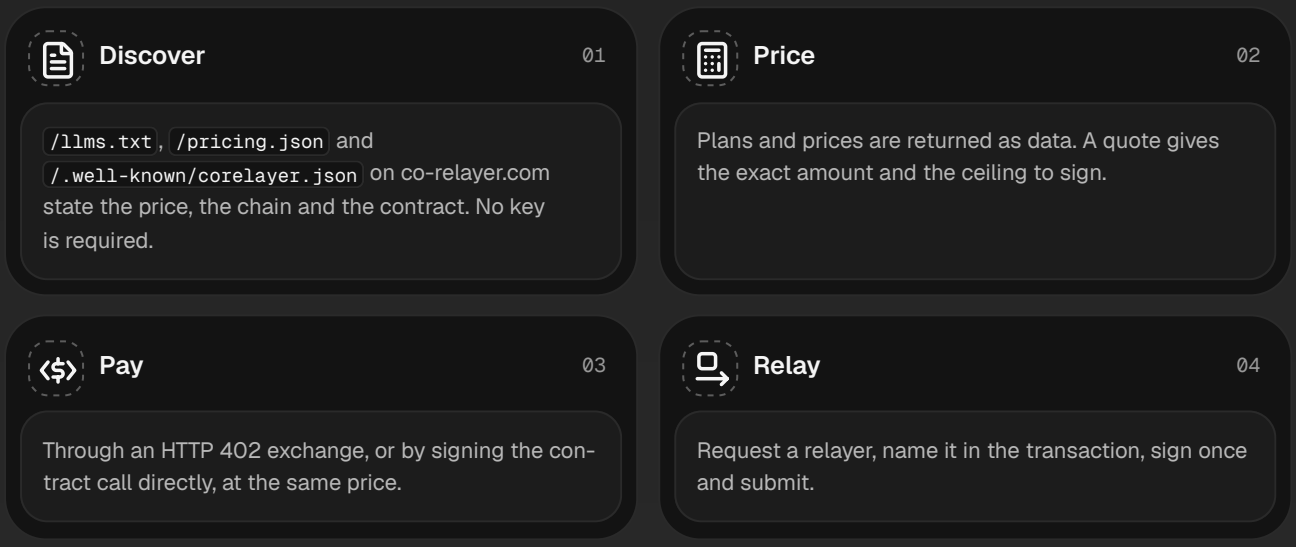
Agent Metered has no monthly fee: it is pay-as-you-go from purchase and spends only what is deposited.

Large purchases. A single payment is capped from the exchange's reserves, so a larger purchase is made in several payments, each converted to credits.

07

AI agents

An agent with USDC and a MultiversX key can find CoRelayer, choose a plan, pay and send transactions without an account, a dashboard or a person.



An agent pins the chain id and the contract address from `/.well-known/corelayer.json`; the SDKs can check each relayer against the on-chain list before signing. One OpenAPI document describes the API.

Every agent plan pays for its own account and includes named wallets: 1 on Agent Metered, 25 on Agent Pro and 1,000 on Agent Fleet, and more for a small one-time fee each. On Agent Pro and Agent Fleet, a platform's sponsor key can pay for every agent address, within the allowed contracts and an optional daily limit per agent.

7.1 Payment over HTTP 402

x402 [8] turns HTTP's 402 Payment Required into a payment exchange: the API states what it requires, the agent signs one payment transaction and repeats the request. CoRelayer uses it to add credits and to buy plans, with the same contract call and the same price as a direct purchase (Figure 3).

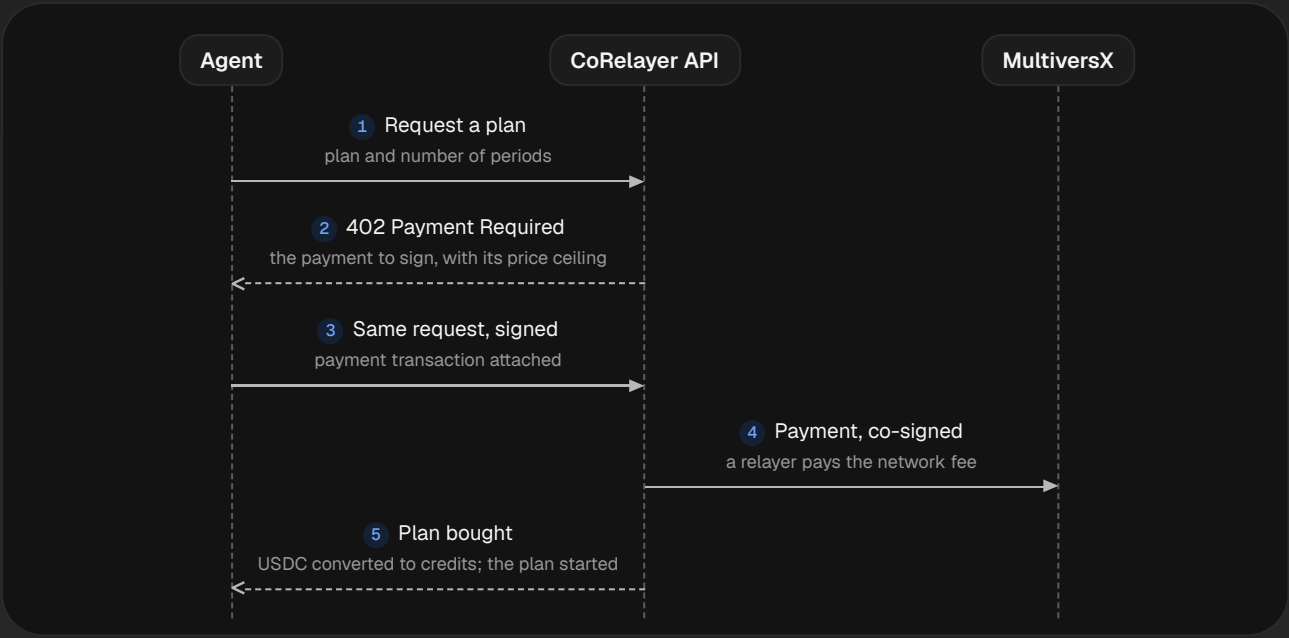


Figure 3 Buying a plan over HTTP 402: one signature, no account.
CoRelayer documentation: buying over x402

Spending limits are part of the transactions an agent signs, so the contract enforces them: a purchase above its ceiling is refused, a renewal above its ceiling is skipped, and sending pauses when the pay-as-you-go budget is spent. An agent can spend only the USDC it deposited. Every error carries a stable code and states whether to retry.

At `mcp.co-relayer.com`, prices, network status, relayers and quotas are available as tools of the Model Context Protocol (MCP) [7], each wrapping one API route. Three routes at the API's root follow the interface of a widely used open-source agent relayer kit, so existing integrations change only a base URL.

08

Payments

Plan payments go to the CoRelayer contract, not to a CoRelayer wallet. In the same transaction the contract converts each payment and divides it in a fixed order, relayers first.

The contract swaps the USDC to WEGLD (wrapped EGLD) on xExchange and divides the result:

01 **Relayers.** Exactly enough EGLD, counted at the worst case, for every transaction the payment makes available: the payment's **relayer funds**. A payment whose swap would leave them short is refused.

02 **Network-fee reserve.** Refills the reserve that pays the network fees of CoRelayer's own transactions, such as settlement.

03 **Running costs.** The month's server costs, up to a **running-cost budget** capped by the contract.

04 **Remainder.** Described in the whitepaper of CoreLayer Units, CRL-aaf104 on MultiversX.

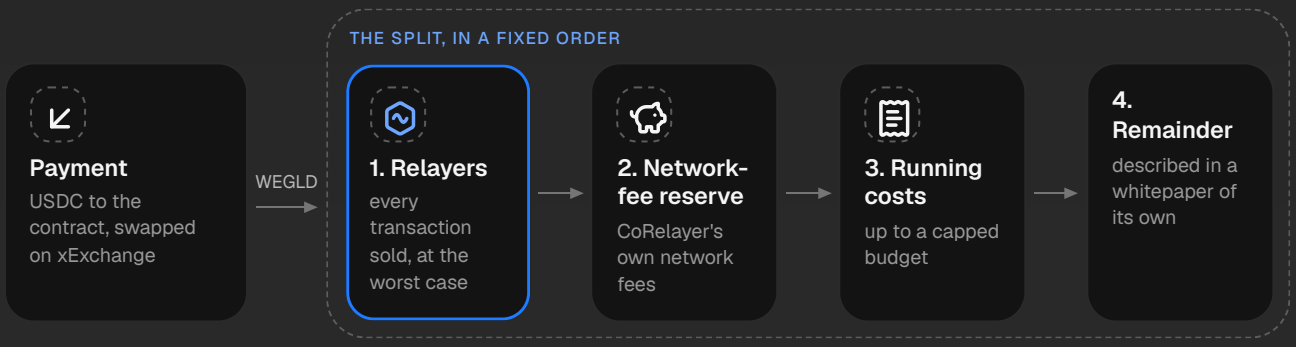


Figure 4 The path of one payment: one transaction, one swap and a split in a fixed order.

The CoRelayer contract

Credits are counted from the USDC paid, not from the swap's result, so a poor swap costs CoRelayer, not the customer. If the exchange pauses the swap, a purchase is refused as a whole and the USDC stays in the buyer's wallet; plans already paid continue.

The contract keeps one invariant: the EGLD delivered to the relayers covers, at the worst case, every transaction any customer can still use. The running-cost budget rises one step at a time along a ladder fixed in the contract, at most once in 30 days and 48 hours after announcement; a cut applies at once. Real transactions usually cost less than the worst case: after a period ends, the difference is proven from the relayers' actual fees and released after 48 hours, then follows the same order after the relayers.

Anyone can also give EGLD to the relayers through the contract. A donation passes on in the same transaction to the active relayers, by weight; none of it stays in the contract, and it buys no plan and no credits.

8.1 Usage on record

Each payment's split is an event of the payment's own transaction, and the contract's views show what the relayers must cover and have received.

The contract records what each account bought. CoRelayer counts usage and reports only pay-as-you-go debits to the contract, in batches. The reporting key moves no tokens and touches no credits: it can only debit a pay-as-you-go budget, within a rate limit the contract keeps for each account, and the contract prices every debit from the prices stored at purchase.

Each settlement batch writes into the contract a Merkle root over its window's usage and a rolling head over every earlier batch. Any billed transaction can be proven against that root with a **usage proof**, and no earlier batch can be rewritten without changing a head already on chain. The dashboard reports transactions per day, with a CSV export.

09

Operations

Relayers operate in every MultiversX shard and are kept funded.

Assignment. A relayer is assigned just before the user signs, from the healthy, funded relayers in the sender's shard.

Funding. A balance watcher keeps every assignable relayer above a floor; a relayer below it stops being assigned before it can run short.

Redundant signing. Each relayer key runs on more than one server. Relayer signatures are deterministic, so every server produces the byte-identical transaction.

Access points. Each co-signed transaction is sent through several independent **access points**, MultiversX node interfaces or gateways, at once.

Own observing squad. CoRelayer runs its own **observing squad**, MultiversX's term for observer nodes covering every shard and the metachain behind a proxy [12], and sends and reads through it alongside the public access points. CoRelayer reads the chain from its own squad first; a public access point answers when the squad is behind the network or does not answer. Each transaction is followed to finality in the shard that executed it, and it is reported final only once public access points, run independently of CoRelayer, confirm its outcome.

Spares. Spare relayers are registered on chain in advance and activated in one on-chain step.

Incidents are published at status.co-relayer.com [9], hosted separately from the service. The dashboard shows each transaction's timeline and relayer, usage per day, sponsor keys, named wallets, spending limits and notices. Notices also go by e-mail, to an address verified by link: a renewal three days ahead, a plan that has ended, a plan's transactions used up, and incidents. Each kind can be switched off.

SDKs for TypeScript, Go, Rust and Python relay a transaction with one signature, raise a typed error for every error answer, support sponsor keys, check relayers against the contract when enabled, and cover every API operation. One OpenAPI 3.1 document [10] [11] describes the HTTP API and generates the reference, the SDK types and a Markdown summary for agents; the document and the SDKs are licensed under Apache 2.0. The compatibility, x402 and MCP surfaces are adapters in front of the same relay pipeline. A breaking change creates a new major version, which runs alongside the previous one.

10

Growth

CoRelayer starts as an operated fleet and opens to independent operators as it grows. Each stage starts when usage requires it and revenue pays for it.

No **growth stage** has a date. Relayer funds grow with sales; a stage that adds servers is paid by the next step of the running-cost budget; running costs are covered before any remainder. Two stages add further observing squads beside CoRelayer's own, which chapter 9 describes.

More relayers per shard

OPENS WHEN A shard's sustained traffic needs more fee payers than its active relayers provide.

FUNDED BY The relayers' part of that traffic's sales.

ADDS

Spare relayers activated in that shard.

Dedicated processors

OPENS WHEN Measured processing time on shared servers drifts from its objective.

FUNDED BY The next step of the running-cost budget.

ADDS

Core servers move from shared to dedicated processors.

High-volume plans, self-serve

OPENS WHEN The first customer whose volume needs one of them.

FUNDED BY The relayer funds held in every shard cover an hour of those plans' sending.

ADDS

Scale, Enterprise and Agent Fleet bought in the app.

Second observing squad

OPENS WHEN — The squad in steady use, or before its first maintenance upgrade.

FUNDED BY — The next step of the running-cost budget.

ADDS

A second squad in another region.

More signing servers

OPENS WHEN — Relayer funds or active relayers outgrow the signing servers in place, or a priority-lane or dedicated-relayer customer arrives.

FUNDED BY — The next step of the running-cost budget.

ADDS

More signing servers in more regions, each relayer key on more than one.

Dedicated database servers

OPENS WHEN — Stored history outgrows the core servers, or recording slows.

FUNDED BY — The next step of the running-cost budget.

ADDS

The system of record on its own servers.

API and signing separated

OPENS WHEN — Sustained volume well beyond what the squads in place are sized for.

FUNDED BY — The top step of the running-cost budget.

ADDS

Public API servers apart from signing servers, and a third observing squad.

PHASE	CONTENT	IT ENDS WHEN
0. Foundation	An operated fleet in every shard, prices set in the contract, plans an agent can buy on its own.	Latency is measured and published for every shard.
1. Policy and fleets	Sub-accounts per sender and value limits.	Fleets can split one plan across teams with their own limits.
2. Latency classes and SLAs	Published objectives become contractual SLAs with credits, with more dedicated relayers in more regions.	A published record of measured delivery times exists for every class and shard.
3. Open cooperative	Third-party operators run relayers under the same on-chain rules and measured standards.	Independent operators serve production traffic, and a governance document exists.
4. Beyond one chain	One fleet per sovereign chain, gas stations that take payment on one chain and sponsor on MultiversX, and CoRelayer as the default relayer behind x402 facilitators and MCP servers.	The sovereign-chain documentation is complete and the first sovereign customer runs on it.

Table 10 The five public phases. None has a date.
CoRelayer

Each stage adds servers or enables what the contract already provides, behind the same API, so integrations do not change. A price change accompanying a stage follows the same 48-hour rule, and paid periods keep their price. Each stage leaves a public record: new relayers in the relayer list, plan changes on the pricing page, other changes in the documentation changelog.

11

Summary

01 A relayer in the sender's shard pays the network fee, through relayed transactions v3, a feature of MultiversX (chapters 2 and 3).

02 The user signs once, in an existing wallet; the relayer is named before signing (chapter 3).

03 One USDC price per transaction, set on chain; priced per transaction, never per user (chapter 6).

04 Price rises take effect 48 hours after announcement; a paid period keeps its price (chapter 6).

05 A relayer cannot alter a signed transaction, cannot send it twice and cannot move any balance (chapter 5).

06 A relayer can delay or decline; every relayer is listed on chain and can be checked before signing (chapter 5).

12

Glossary

Access point	A MultiversX node interface, or gateway, for sending and reading transactions.
Agent Metered	The plan with no monthly fee, paid per transaction from a USDC deposit.
Chain id	The network identifier in every signed MultiversX transaction.
CoRelayer contract	The smart contract on MultiversX shard `1` that holds prices, plans, credits, the relayer list and the payment split.
CoreLayer Units	The token CRL-aaf104 on MultiversX, described in its own whitepaper.
Credits	The balance a payment becomes in the contract, counted from the USDC paid. Credits buy plans and extra transactions and are not refundable.
EGLD	MultiversX's native token, in which network fees are paid.
Growth stage	A capacity addition that starts on a usage signal and is paid by revenue.
Named wallets	Addresses recorded on chain so that an account's plan pays for them.
Network fee	The fee MultiversX charges for every transaction, paid in EGLD.
Network-fee reserve	A reserve in the contract that pays CoRelayer's own network fees.
Nonce	A sender's transaction counter; the network accepts each value once.
Observing squad	Observer nodes for every shard and the metachain, behind a proxy.
Owner, operations and reporting keys	The contract's three roles. The offline owner key sets prices and upgrades; the operations key pauses; the reporting key only reports usage.
Pause switch	A switch that stops one money path; only the owner key resumes it.
Pay-as-you-go budget	Credits set aside to continue sending after a plan's transactions are used.
Phase	One of five public roadmap steps, from Foundation to Beyond one chain.
Plan	A purchase covering 30 days at a time, 1 to 12 periods at once.

Price ceiling	The most a purchase or renewal may cost, signed by the buyer.
Price per transaction	The tariff: the one number in the contract that sets every USDC price.
Relay Unit	The counting unit of the documentation and the API: one per 0.001 EGLD of worst-case network fee. It is distinct from CoreLayer Units.
Relayed transaction	MultiversX's built-in format (relayed v3) in which a relayer named in the signed transaction pays the fee. The relayer must be in the sender's shard.
Relayer	A MultiversX wallet CoRelayer operates that co-signs a user's transaction and pays its network fee, with no authority over any balance.
Relayer class	Shared relayers, a priority lane or dedicated relayers, set by the plan.
Relayer funds	The part of each payment that funds the relayers, at the worst case.
Relayer list	The on-chain list of CoRelayer's relayers and their states.
Running-cost budget	The amount per 30 days that pays for servers, capped by the contract.
Sending pauses	At the plan's limit, nothing more is sent or charged until the customer acts.
Settlement	The periodic call that debits pay-as-you-go budgets and commits usage.
Shard	One of MultiversX's three parallel partitions of accounts.
Signing server	A server that holds relayer keys and adds the relayer's signature.
Speed limits	Per-plan limits on how many transactions may be sent in a given time.
Sponsor key	A secret key on a customer's server that makes the plan pay for transactions signed by any user, for the listed contracts.
Usage proof	A Merkle proof that a billed transaction is in a usage root on chain.
USDC	The stablecoin customers pay in: `USDC-c76f1f` on MultiversX.
Verified build	Deployed code anyone can rebuild from its source, byte for byte.
x402	An open protocol that turns HTTP 402 into a payment exchange.
xExchange	The MultiversX exchange on which the contract swaps each payment.

13

References

[1] MultiversX documentation: relayed transactions.
docs.multiversx.com/developers/relayed-transactions

[2] CoRelayer documentation: delivery, failures and charging.
docs.co-relayer.com/concepts/delivery-guarantees

[3] CoRelayer documentation: the trust model.
docs.co-relayer.com/security/overview

[4] CoRelayer documentation: verifying a relayer.
docs.co-relayer.com/concepts/verify-a-relayer

[5] The CoRelayer contract's address and chain id.
co-relayer.com/.well-known/corelayer.json

[6] CoRelayer documentation: transaction counting, with the published test vectors.
docs.co-relayer.com/concepts/relay-units

[7] Model Context Protocol specification.
modelcontextprotocol.io/specification

[8] x402: an open standard for payments over HTTP 402.
www.x402.org

[9] CoRelayer status page.
status.co-relayer.com

[10] The CoRelayer OpenAPI document.
docs.co-relayer.com/openapi.yaml

[11] OpenAPI Specification, version 3.1.
spec.openapis.org/oas/v3.1.0

[12] MultiversX documentation: the observing squad.
docs.multiversx.com/integrators/observing-squad

13.1 Further reading

- One-pager → co-relayer.com/papers/corelayer-one-pager.pdf
- API overview → co-relayer.com/papers/corelayer-api-overview.pdf
- CoreLayer Units whitepaper (CRL-aaf104) → co-relayer.com/papers/corelayer-units-whitepaper.pdf
- Documentation → docs.co-relayer.com



CoRelayer

COOPERATIVE RELAYER INFRASTRUCTURE
FOR AUTONOMOUS TRANSACTIONS

co-relayer.com · support@co-relayer.com

© 2026 CoRelayer · Whitepaper · Revision 1.0