OREUM
NETWORK

OFFICIAL WHITEPAPER · V1.0 · AUGUST 2026

THE AUTOMATION-NATIVE LAYER 1.

Oreum Network is designed as an EVM-compatible Layer 1 blockchain built around reliable transaction automation. The network centers scheduled, recurring and conditional execution through Oreum Autopilot and bounded Oreum Rules.

PRIMARY TAGLINE

Set the Rule. Oreum Does the Rest.

VERSION

1.0

DATE

August 2026

STATUS

Official Whitepaper

SECTIONS

24

IMPORTANT NOTICE

Document Status and Disclaimer

This whitepaper describes a proposed network design.

Unless explicitly marked as live, all protocol features, performance characteristics, token utilities, integrations and roadmap items are plans or engineering objectives. They may change after research, implementation, security audits, public testing, governance review and legal analysis.

This document is provided for technical and informational purposes only. It is not an offer to sell securities, a solicitation, investment advice, financial advice, legal advice or a promise of token value, returns, exchange listing or network launch. Access to products or tokens may be restricted in some jurisdictions. Users and contributors should obtain independent professional advice before relying on any economic or legal statement in this document.

The name $ORM is used in this document as the proposed native network asset. Final supply, allocation, vesting, distribution, launch mechanics and regulatory treatment are intentionally excluded from this version and must be published separately after economic modelling and legal review.

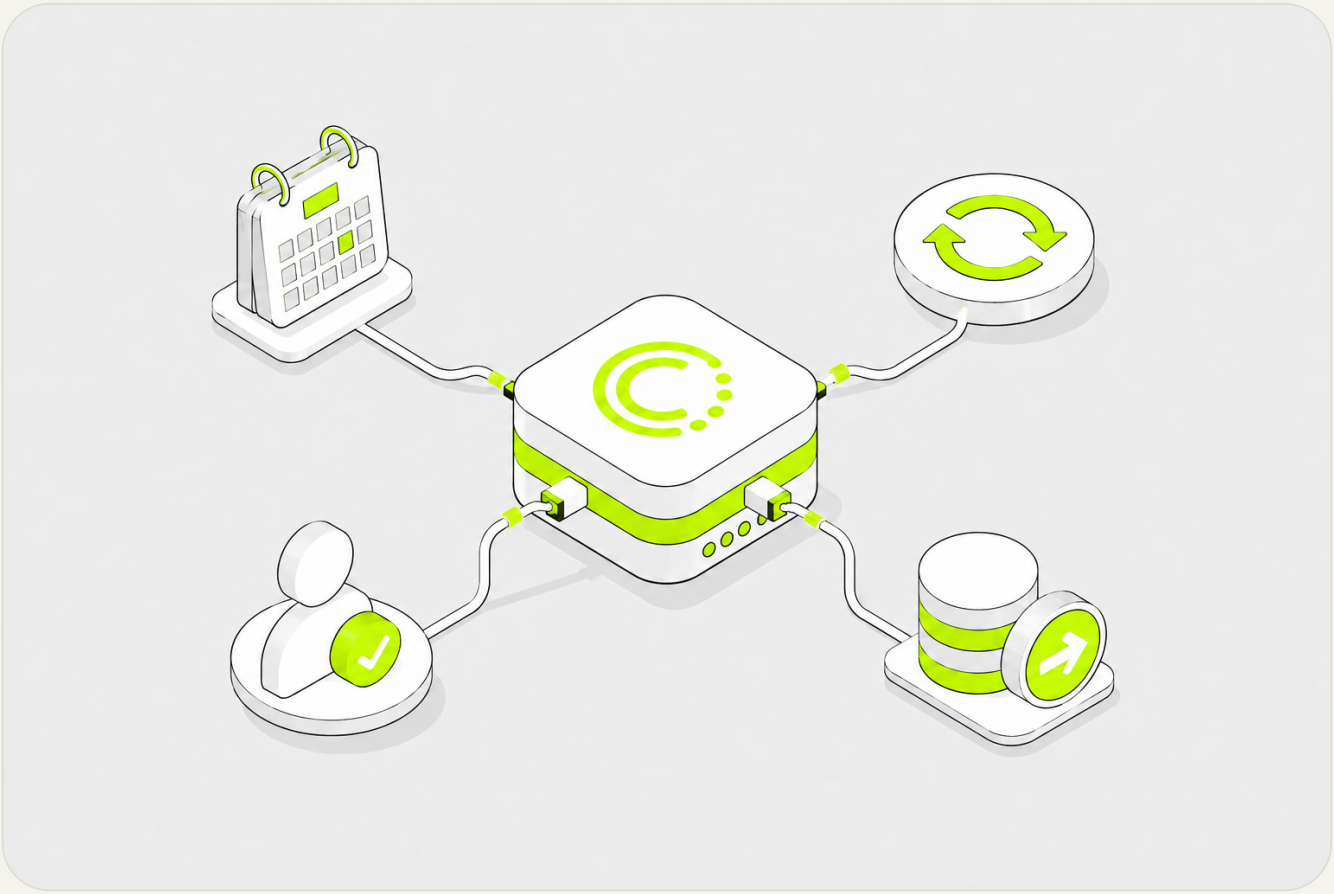
Oreum Network will publish implementation repositories, audit reports, testnet benchmarks and protocol specifications as they become available. Where this document discusses other networks, the comparisons are used only to explain the design landscape and do not imply affiliation, endorsement or technical equivalence.

CONTENTS / 25 SECTIONS

Whitepaper Structure

Jump directly to any section of the Oreum Network Whitepaper. The document moves from the core thesis and network architecture through Autopilot, Rules, developer tooling, economics, governance and roadmap.

Executive Summary



Oreum Network is proposed as an EVM-compatible Layer 1 blockchain built around a capability that most networks treat as external infrastructure: reliable transaction automation.

Today, users can send a transaction immediately, but scheduling a salary, renewing a subscription, releasing vested tokens, distributing community rewards or executing a conditional action normally requires a centralized server, an external keeper network, repeated manual signatures or custom backend infrastructure. This creates operational complexity, custody risk and poor user experience.

Oreum addresses this gap through Oreum Autopilot, a protocol-native framework for scheduled, recurring and conditional transactions. Users and applications create bounded instructions called Oreum Rules. Each Rule defines exactly what may happen, when it may happen, how much may be spent, which assets and recipients are allowed, how fees are funded and when the permission expires.

Core proposition

Oreum Network is the Layer 1 where users define the outcome once and the network executes it according to transparent, revocable rules.

The proposed network combines a modern Proof-of-Stake Byzantine Fault Tolerant consensus model, EVM compatibility, parallel transaction execution, programmable smart accounts, atomic multi-action transactions, sponsored gas and a dedicated automation execution lane. AI is used only as an optional interface that translates natural-language instructions into structured Rules; deterministic protocol logic and user signatures remain the source of authority.

Pillar	Oreum approach
Automation-native	Scheduling and conditional execution are standard network capabilities, not optional backend integrations.
User-controlled	Every automated action has explicit limits, expiration, revocation and a visible audit trail.
Developer-friendly	Solidity, EVM tooling, SDKs, verified templates and standard APIs reduce migration friction.
Predictable execution	A reserved automation gas lane protects eligible Rules from ordinary mempool congestion.
Consumer-ready	Smart accounts, sponsored gas, OneTap transactions and plain-language SafeSign confirmations reduce complexity.
Progressive decentralization	The first implementation can use approved executors; later releases move eligibility and inclusion guarantees into protocol validation.

SECTION 02

The Problem

Blockchains are programmable, but the transaction experience remains largely reactive, manual and infrastructure-heavy.

2.1 Transactions are immediate by default

Most Layer 1 networks are optimized to accept a signed transaction now. They do not natively represent a user instruction such as “pay these contributors on the first business day of every month” or “release this payment after the buyer and seller both approve.” Developers therefore recreate scheduling and conditional execution using off-chain services.

2.2 Automation is fragmented

Existing automation systems demonstrate demand for time-based, log-based and custom-logic triggers, but they usually operate as external services that developers must integrate, fund and monitor. This creates separate operational dependencies for scheduling, permissions, gas, retries and observability. Chainlink Automation, for example, positions automation as decentralized infrastructure for smart contracts, illustrating both the value of the capability and the fact that it is commonly added above the base chain rather than built into the transaction model.

2.3 Wallet permissions are too broad

A traditional wallet signature is often all-or-nothing: the user approves one transaction at a time or gives a contract an allowance that may remain open indefinitely. Recurring payments and autonomous actions need narrower permissions: specific recipients, tokens, amounts, frequencies, contracts and expiry dates.

2.4 Gas creates recurring friction

A future transaction can fail because the user no longer holds the native gas token, gas prices exceed the approved budget or the sponsoring application has insufficient funds. A credible automation network must make gas funding part of the Rule itself.

2.5 Speed alone is not a durable identity

Modern Layer 1 networks increasingly combine high throughput, fast finality and familiar development tools. Monad documents full EVM bytecode and RPC compatibility alongside asynchronous and parallel execution, while Sei emphasizes parallel EVM execution and sub-second blocks. Oreum must therefore compete through a distinct network capability, not only a faster block-time claim.

SECTION 03

Vision, Mission and Positioning

Oreum is designed around the belief that blockchains should execute policies, not merely process clicks.

Vision

Vision

A global programmable transaction network where individuals, applications and businesses can schedule and automate value securely without surrendering control of their assets.

Mission

To make automated on-chain activity simple, reliable and accessible by integrating scheduling, conditional execution, smart-account permissions, flexible gas and transparent monitoring directly into the Oreum ecosystem.

Category and brand position

Element	Position
Category	Automation-native Layer 1
Primary tagline	Set the Rule. Oreum Does the Rest.
Supporting line	Transactions that run on your schedule.
User promise	Define a bounded instruction once; remain able to pause, modify or revoke it.
Developer promise	Add reliable automation without operating bots, private keys and retry servers.
Business promise	Automate payroll, subscriptions, vesting, rewards and settlements from one auditable system.

What Oreum is not

- Oreum is not an “AI-controlled blockchain.” AI does not determine consensus or receive unrestricted access to user funds.
- Oreum is not a mandatory-privacy network. Public transactions remain the default; confidentiality is a later, legally reviewed research track.
- Oreum is not positioned as the fastest blockchain without public evidence. Performance claims must follow reproducible testnet benchmarks.
- Oreum is not a guaranteed-income or investment-return programme. Community participation and token economics require separate rules and disclosures.

SECTION 04

Design Principles

The protocol and product experience should be evaluated against seven non-negotiable principles.

User authority

Automation can only operate within permissions the user or approved governance account has explicitly authorized.

Determinism

Consensus-critical execution uses structured rules and verifiable data, never open-ended AI judgment.

Bounded risk

Every Rule supports maximum amounts, fee limits, frequency limits, allowlists, expiry and emergency revocation.

Predictable inclusion

Eligible Rules receive reserved network capacity and transparent priority rules.

Familiar development

EVM compatibility allows developers to use Solidity, standard wallets, JSON-RPC and established tooling.

Progressive decentralization

Early implementation choices prioritize safety and observability; decentralization increases as the protocol is tested.

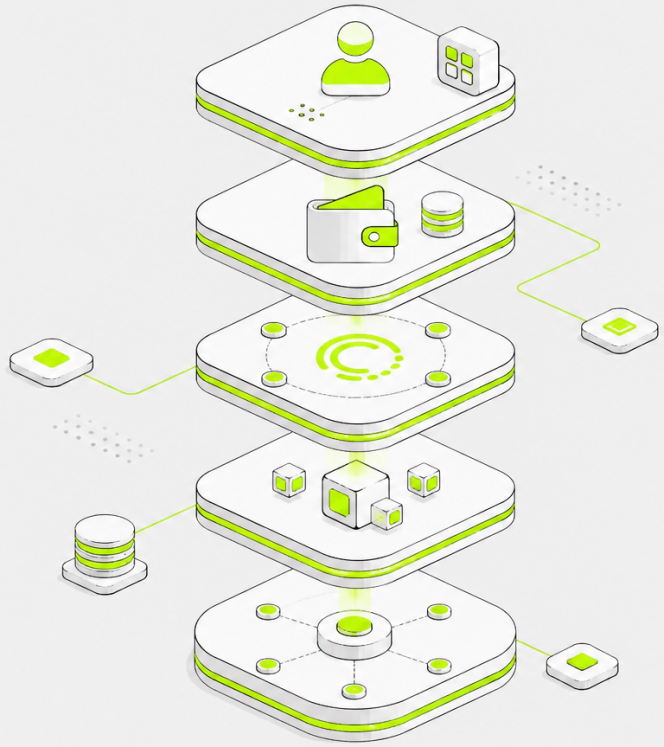
Truthful communication

Live, testnet, planned and research features are clearly separated across the website, documentation and explorer.

SECTION 05

Network Architecture

Oreum is organized as a layered system in which standard smart-contract execution and automation-specific services share one consensus and security domain.



OREUM NETWORK ARCHITECTURE

A layered design for high-performance execution and protocol-native automation

USER & APPLICATION LAYER

Oreum App • Wallets • dApps • Businesses • AI-assisted interfaces

ACCOUNT & TRANSACTION LAYER

Smart Accounts • OneTap Bundles • SafeSign Simulation • FlexGas Paymasters

OREUM AUTOMATION LAYER

Rules Registry • Scheduler • Trigger Adapters • Automation Gas Lane • Retry & Monitoring

EXECUTION LAYER

EVM-compatible execution • Parallel transaction processing • Deterministic state transition

CONSENSUS & DATA LAYER

Proof-of-Stake BFT • Validators • Block propagation • State database • Explorer & indexers

5.1 User and application layer

The Oreum App, third-party wallets, dApps, business dashboards and AI interfaces create standard transactions or Oreum Rules. Applications may sponsor gas, provide templates and request limited smart-account permissions.

5.2 Account and transaction layer

Smart accounts manage authentication, session keys, spending policies and recovery. OneTap bundles multiple contract calls into one atomic action. SafeSign simulates and explains the expected outcome before authorization. FlexGas determines who pays execution costs and how the application settles those costs.

5.3 Automation layer

The Rules Registry stores user-approved automation manifests. The Scheduler determines which Rules are eligible. Trigger adapters expose permitted sources of time, on-chain events, balances and oracle data. The Automation Gas Lane reserves block capacity for eligible Rules, while monitoring contracts record success, failure, retry and expiry state.

5.4 Execution layer

Oreum is proposed as EVM-compatible so Solidity contracts and established Ethereum tooling can be used with minimal adaptation. The execution client should implement deterministic parallelism for independent transactions while preserving the same result as sequential EVM ordering. This general approach is used by high-performance EVM designs such as Monad and Sei.

5.5 Consensus and data layer

Validators order transactions, verify protocol rules and finalize blocks under a Proof-of-Stake BFT model. Full nodes execute state transitions, serve RPC data and support indexers, explorers and analytics. Client diversity and hardware requirements must be defined after public performance testing rather than optimized solely for benchmark numbers.

SECTION 06

Consensus and Validator Network

Oreum requires fast finality, predictable ordering and a validator set capable of enforcing automation inclusion rules.

6.1 Proposed consensus properties

Property	Proposed direction
Sybil resistance	Proof of Stake using $ORM or a governance-approved staking asset.
Safety threshold	Block finality requires more than two-thirds of active stake weight under normal BFT assumptions.
Proposer model	Stake-weighted rotating block proposers with deterministic leader selection.
Delegation	Token holders may delegate stake to validators without operating infrastructure.
Fault handling	Timeouts and view changes allow progress when a proposer is offline or malicious.
Finality objective	Fast deterministic finality suitable for payments and automation; exact targets must be benchmarked publicly.
Slashing	Penalties for provable safety violations; liveness penalties subject to conservative design and governance review.

Modern high-performance BFT systems demonstrate that sub-second block production and large validator sets require careful communication, propagation and execution design. MonadBFT, for example, documents linear happy-path communication, stake-weighted supermajorities and two-round finality. Oreum should use these lessons as technical context without claiming protocol equivalence or copying an implementation.

6.2 Validator responsibilities

- Validate ordinary transactions and smart-contract execution.
- Recompute the deterministic eligibility of scheduled Rules.
- Verify that a proposed block contains the required prefix of eligible Rules within the automation quota.
- Confirm fee reserves, paymaster commitments, nonces, permissions and expiry conditions.
- Publish signed performance and availability data for explorer visibility.
- Participate in upgrades, incident response and state synchronization procedures.

6.3 Decentralization objectives

Validator hardware should be demanding enough to support high throughput but not so specialized that participation becomes limited to a small number of operators. Oreum will publish minimum and recommended specifications only after client profiling. The network should support independent validators, multiple geographic regions, delegated staking and transparent concentration metrics.

SECTION 07

Execution and Performance

Oreum’s performance strategy should improve software efficiency while preserving EVM semantics and verifiable execution.

7.1 EVM compatibility

The proposed execution environment supports Solidity smart contracts, Ethereum-style addresses, standard token interfaces, JSON-RPC endpoints and common development tools. Compatibility reduces the cost of migrating applications and allows Oreum to benefit from mature wallet, audit and developer infrastructure.

7.2 Parallel execution

Transactions that do not conflict on state can be executed concurrently across CPU cores. An optimistic concurrency engine may begin transactions in parallel, track their read and write sets, detect conflicts and re-execute conflicting transactions before state is committed in canonical order. This preserves deterministic results while improving throughput for independent payments and application activity.

7.3 Consensus-execution separation

A later client version may pipeline consensus and execution so the network agrees on transaction ordering without forcing every state transition into the consensus hot path. Monad documents asynchronous execution as a way to give execution the full block-time budget rather than sharing the same narrow window with consensus. This is an advanced design choice and should only be adopted after formal specification, failure analysis and extensive testnet validation.

7.4 Performance targets

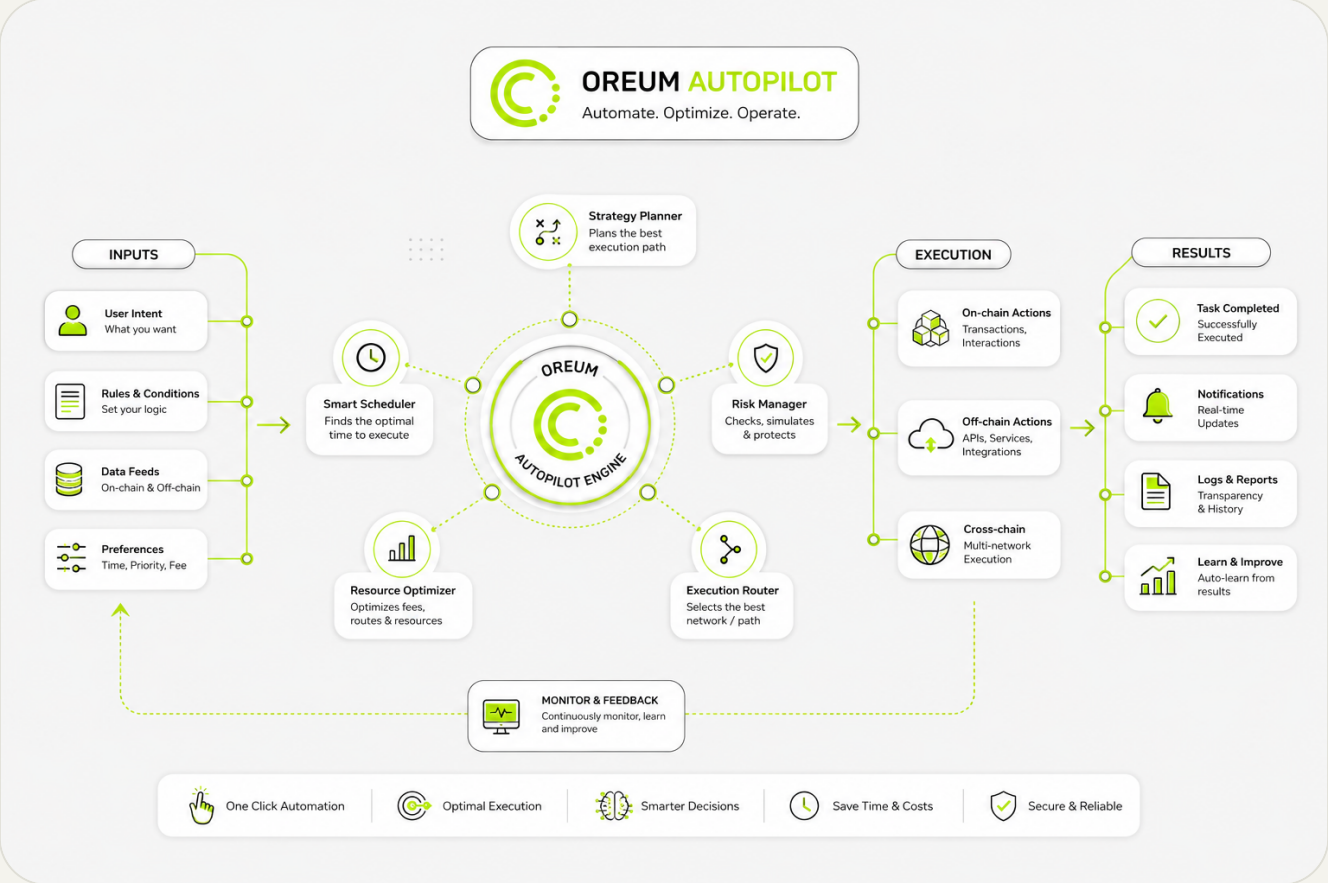
No unverified benchmark marketing

Oreum will publish block time, finality, throughput, state growth and hardware measurements only after reproducible public tests. Targets in internal engineering documents are objectives, not guarantees.

Metric	Publication standard
Block interval	Measured under geographically distributed validators and disclosed hardware.
Finality	Measured from transaction submission to irreversible confirmation under normal and degraded network conditions.
Throughput	Reported separately for simple transfers, smart-contract workloads and automation workloads.
Latency	Reported at median, p95 and p99 rather than a single best-case number.
Hardware	Full validator and full-node CPU, memory, storage and bandwidth requirements.
State growth	Storage expansion per million transactions and archive-node requirements.
Automation reliability	Percentage of eligible Rules executed within the promised inclusion window.

SECTION 08

Oreum Autopilot



Oreum Autopilot is the user-facing and developer-facing automation system built on top of protocol-enforced Rules.

Autopilot allows a user, application or business account to authorize a future transaction without repeatedly signing each execution. The permission is not a blank cheque. It is a structured policy with defined scope, funding, schedule, limits, failure behaviour and revocation rights.

8.1 Supported automation classes

Class	Description	Initial availability
Scheduled	Execute once at or after a specified network time or block height.	MVP
Recurring	Execute at a defined interval until a count, date or budget limit is reached.	MVP
Event-triggered	Execute after a specified contract emits an approved event.	Phase 2
State-triggered	Execute when an on-chain balance, threshold or status becomes true.	Phase 2
Oracle-triggered	Execute using a value signed or published by an approved oracle.	Phase 3
Multi-condition	Execute only when a bounded combination of time, state and oracle conditions is true.	Phase 3

8.2 Example

Monthly contributor payment

Send 100 USDC to an approved recipient on the first day of every month for six months. Never pay more than 600 USDC in total. Use the business gas reserve. Pause automatically after two failed executions and notify all account approvers.

The example becomes a signed Rule manifest. The network does not interpret ambiguous language at execution time. It evaluates the stored token address, recipient, amount, next execution time, remaining count, fee cap, funding source and account policy.

SECTION 09

Oreum Rules

An Oreum Rule is a protocol-recognized authorization for a future or conditional action.

OREUM RULE LIFECYCLE

Every automated action remains authorized, funded, bounded and revocable.

1 — CREATE
User defines action, schedule, limits and permissions

→

2 — AUTHORIZE
Smart account signs the Rule manifest

→

3 — FUND
Gas reserve or sponsor is assigned

→

4 — ELIGIBLE
Time, event or condition becomes valid

→

5 — EXECUTE
Validators process the Rule in the automation lane

→

6 — SETTLE
Complete, reschedule, retry, pause or expire

USER CONTROL: PAUSE • EDIT • CANCEL • REVOKE • WITHDRAW UNUSED RESERVE

9.1 Rule manifest

Field	Purpose
Rule ID	Unique identifier derived from creator, account, nonce and chain ID.
Owner / account	Smart account authorized to perform the action.
Target	Approved recipient or contract address.
Call data hash	Exact action or approved template to execute.
Asset and amount	Permitted token and fixed or maximum value.
Trigger	Time, recurrence, on-chain state, event or approved oracle condition.
Validity window	Earliest and latest time in which the Rule may execute.
Frequency and count	Minimum spacing and maximum number of executions.
Gas policy	Fee payer, reserve, max fee, priority tier and settlement method.
Failure policy	Retry count, delay, fallback, pause and notification behaviour.
Security policy	Allowlists, session key, multisig threshold and human re-approval rules.
Status	Configured, active, paused, eligible, executing, completed, failed, cancelled or expired.

9.2 State machine

A Rule begins in a configured state, becomes active after authorization and funding, enters the eligible state when its trigger is satisfied, and is then executed through the automation lane. Recurring Rules calculate a new next-execution value after success. A Rule completes when its execution count, end date or total budget is reached. Users may pause or cancel a Rule at any point before execution begins.

9.3 Deterministic trigger policy

Consensus-critical trigger evaluation must be reproducible by every validator. The launch version should therefore support network time, block height and explicit on-chain state. Off-chain facts are introduced only through approved oracle contracts. Free-form web data or AI-based judgments are never accepted directly by consensus.

9.4 Rule funding

Every Rule must identify a valid gas and execution funding method. Funding may come from the user’s native balance, an escrowed reserve, a business gas account or an application paymaster. A Rule that lacks sufficient authorization or funding becomes ineligible until corrected; it cannot repeatedly consume validator resources without payment.

SECTION 10

Protocol Scheduler and Automation Gas Lane

The protocol scheduler is the principal technical differentiator of Oreum.

10.1 Deterministic due queue

At each block, validators compute a set of eligible Rules from the same finalized state. Eligible Rules are sorted by deterministic keys such as eligibility slot, priority class and Rule ID. A block proposer includes a prefix of this queue until the automation gas quota is filled. Other validators reject a block that improperly skips higher-priority eligible Rules without a valid reason such as cancellation, insufficient funding or failed permission validation.

10.2 Reserved automation capacity

A governance-adjustable portion of each block's gas capacity is reserved for automation. The reserve prevents an NFT mint, trading spike or unrelated dApp from completely crowding out payroll, subscriptions and scheduled settlements. Unused automation capacity may be released to ordinary transactions so network resources are not wasted.

Parameter	Purpose	Governance status
Automation gas quota	Maximum block resources reserved for eligible Rules.	Adjustable within safety bounds
Eligibility look-ahead	Number of upcoming slots indexed by validators.	Protocol parameter
Priority classes	Standard, business, security-critical or governance-defined lanes.	Governed and transparent
Retry delay	Minimum interval after a failed execution.	Rule-defined within bounds
Max retries	Prevents infinite failing Rules.	Rule-defined with protocol ceiling
Expiry grace	Tolerance window for delayed but still valid execution.	Protocol parameter

10.3 Inclusion guarantees

The network should publish service objectives for execution after a Rule becomes eligible. These objectives must be described statistically and linked to congestion assumptions. Repeated proposer censorship or invalid omission can be detected because all validators can recompute the due queue. Future protocol versions may attach penalties to provable systematic omission.

10.4 Failure and retry

An automated transaction may revert because of changed state, insufficient token balance, contract restrictions or fee caps. The Rule records the failure reason. Depending on the signed failure policy, it may retry after a delay, use a bounded fallback action, request new approval, pause automatically or expire. The network must never silently widen permissions to force a transaction to succeed.

SECTION 11

Smart Accounts and User Control

Smart accounts provide the security boundary between a user's assets and automated execution.

ERC-4337 demonstrates that account abstraction can support custom validation, bundling and paymasters without requiring a base-layer consensus change. Oreum can support compatible smart accounts at launch and progressively add native transaction types or protocol hooks when justified by performance and security testing.

11.1 Permission model

- Token limits: only approved assets and maximum values may be spent.
- Recipient and contract allowlists: Rules cannot redirect funds to arbitrary addresses.
- Time limits: session keys and automation permissions expire automatically.
- Frequency limits: an action may occur only at the agreed interval and count.
- Role separation: owners, finance approvers, operators and emergency guardians have different powers.
- Multisig thresholds: higher-value Rules can require multiple approvals before activation or execution.
- Recovery and pause: users can replace authentication keys and freeze automation without moving every asset.

11.2 Session keys

A session key may perform only a narrow class of operations for a limited period. For example, a game session key may sign in-game actions but cannot transfer stablecoins. An AI assistant may prepare Rule data but receives no spending authority unless the user explicitly grants a bounded session permission.

11.3 Emergency controls

The Oreum App will provide a single emergency action to pause all active Rules, revoke session keys and block new automated executions. High-value business accounts may define independent guardians or multisig recovery procedures so a compromised device does not become a single point of failure.

SECTION 12

OneTap, FlexGas and SafeSign

Automation is more useful when the underlying transaction experience is simple and understandable.

12.1 Oreum OneTap

OneTap combines multiple contract calls into one atomic transaction. All steps succeed or the complete action reverts. Examples include swap-and-stake, pay-and-mint, claim-and-reinvest, create-and-distribute, and register-and-reward. Sui's Programmable Transaction Blocks illustrate the value of composing multiple commands into one transaction; Oreum applies a similar user outcome within an EVM-compatible and automation-focused environment.

12.2 Oreum FlexGas

FlexGas separates asset ownership from gas payment. Validators ultimately receive the native fee asset, while paymasters or settlement contracts may sponsor the transaction or accept supported tokens from the user. ERC-4337 paymasters explicitly support sponsored fees and ERC-20 fee payment patterns.

Gas mode	User experience
$ORM native	The account pays the network fee directly in $ORM.
Application sponsored	The dApp or business pays the fee from a deposited gas balance.
Stablecoin settlement	A paymaster charges the user in an approved stablecoin and settles validator fees in $ORM.
Prepaid automation reserve	A Rule escrows enough fee value for its planned executions.
First-use sponsorship	Eligible new users receive a limited number of sponsored interactions.
Fee cap	The transaction or Rule refuses to execute above the user-approved maximum fee.

12.3 Oreum SafeSign

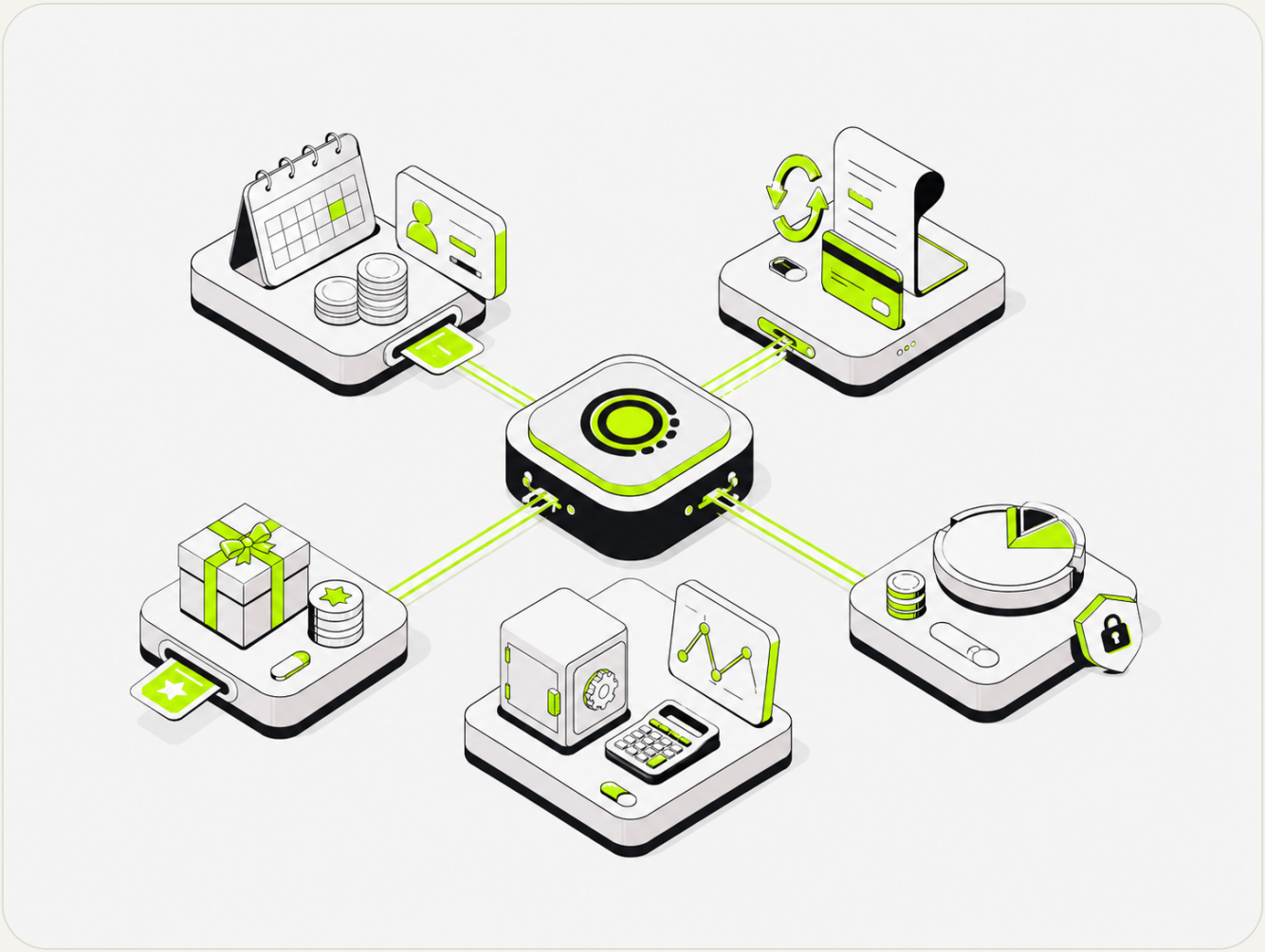
Before a user activates a Rule or signs a transaction, SafeSign simulates the action and displays the expected asset movements, permissions, gas source, total commitment and failure conditions in plain language. A simulation is a risk-reduction tool, not a guarantee that future state will remain unchanged.

Example confirmation

You are authorizing six payments of 100 USDC to one approved recipient. Maximum total transfer: 600 USDC. Gas: business reserve. First execution: 1 September 2026. Final execution: 1 February 2027. You may pause or cancel before any future execution.

SECTION 13

Products and Use Cases



Oreum's initial products should prove the value of automation through understandable, repeatable workflows.

Oreum Payroll

Scheduled batch payments for employees, contractors, ambassadors and contributors. Includes budgets, approval workflows, gas sponsorship, reports and failure alerts.

Oreum Subscriptions

Recurring payments with maximum amount, frequency, expiry, merchant allowlist and instant cancellation.

Oreum Vesting

Transparent cliff, linear and scheduled token releases without manual administrator execution.

Oreum Rewards

Automatic distribution after verified referrals, Activity Point thresholds, on-chain participation or approved campaign milestones.

Oreum AutoStake

Stake a fixed amount or percentage of received $ORM under strict balance, contract and fee limits.

Oreum DCA

Periodically purchase an approved asset subject to budget, price impact and oracle conditions.

Oreum Escrow

Release a payment when specified parties approve or when an approved contract state is reached.

Revenue Sharing

Split incoming funds between creators, partners, treasuries and contributors according to a signed policy.

Treasury Rules

Maintain minimum reserves, schedule grants and enforce multisig budgets for organizations.

Security Automation

Move funds, pause permissions or notify guardians after a defined on-chain security event.

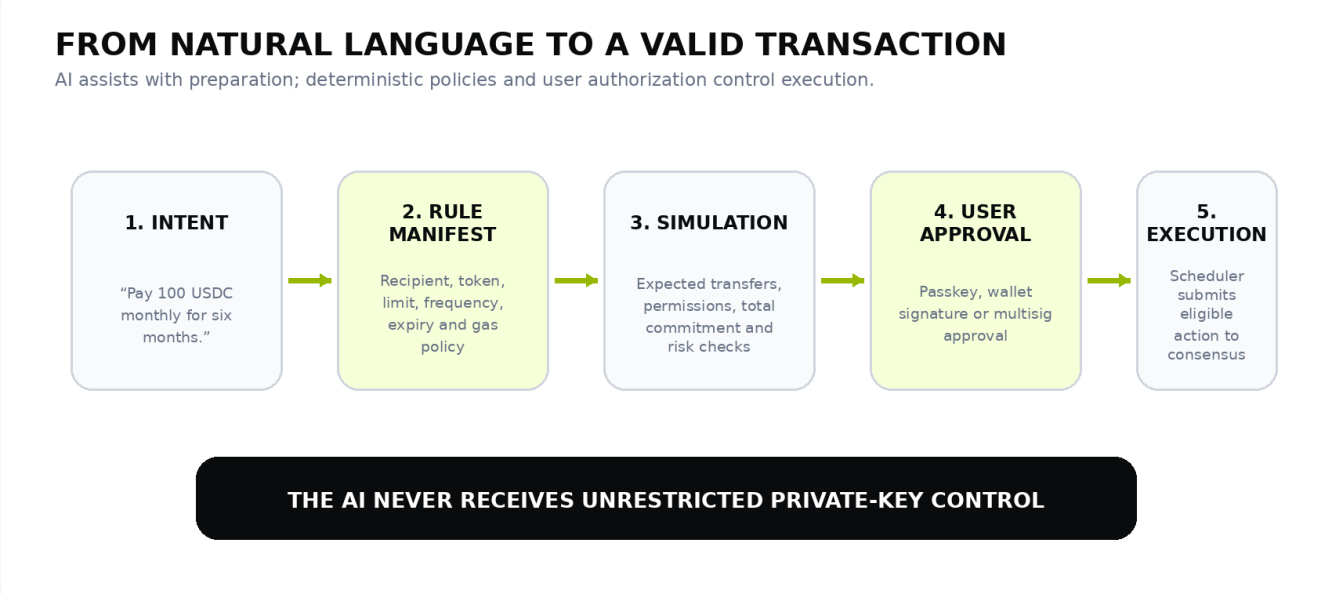
13.1 Oreum App integration

The existing Oreum App can evolve into the primary interface for account access, Activity Points, referrals, ambassador campaigns, Rule management, upcoming transaction calendars, gas reserves, ecosystem Mini Apps and security alerts. Activity Points should remain clearly separate from $ORM unless a published programme defines conversion or reward eligibility.

SECTION 14

AI-Assisted Transactions and Oreum Forge

AI should reduce complexity without becoming a source of transaction authority.



14.1 Oreum AI

Oreum AI converts a natural-language request into a structured Rule or OneTap action. It asks only for missing parameters, selects a verified template, displays assumptions and produces a manifest for simulation and user approval. The AI output is never executed directly.

- AI may suggest recipients, schedules, templates and fee strategies.
- The policy engine validates addresses, limits, contract allowlists and supported trigger types.
- SafeSign simulates the resulting action and shows the total commitment.
- The user, smart account or business multisig signs the final deterministic manifest.
- Sensitive or high-value actions can require additional human approval at execution time.

14.2 Oreum Forge

Oreum Forge is a later no-code development environment for creating applications from verified modules. The first release should not generate unrestricted production DeFi contracts. It should assemble audited templates for tokens, NFTs, memberships, payment pages, vesting, subscriptions, rewards and crowdfunding.

Forge mode	Scope	Security label
Verified	Only approved Oreum modules and fixed configuration options.	Verified template
Custom	AI-assisted or edited code with mandatory testing and explicit warnings.	Automated checks passed
Expert	Unrestricted developer code and deployment tools.	Custom / unaudited unless separately reviewed

Oreum does not need to train a foundational language model at launch. The defensible system is the combination of Oreum documentation, verified templates, Rule schemas, policy checks, simulation, deployment infrastructure and network distribution.

SECTION 15

Developer Platform

Developers should be able to add automation with a small number of clear, composable interfaces.

15.1 Core developer components

- Oreum SDK for transactions, Rules, smart accounts and app integration.
- Rules Registry API for creation, simulation, activation, pausing and cancellation.
- Trigger adapters for time, blocks, contract events, state conditions and approved oracles.
- Paymaster toolkit for sponsored gas, stablecoin charging and business gas accounts.
- OneTap builder for atomic multi-call transactions.
- SafeSign API for simulation and human-readable transaction summaries.
- Automation dashboard for eligible, pending, failed, retried and completed Rules.
- Testnet faucet, explorer, contract verification, indexers and RPC endpoints.
- Verified Action templates for payroll, vesting, subscriptions, airdrops and rewards.

15.2 Example developer interface

```
createRule({
account: businessAccount,
trigger: everyMonth({ day: 1, hourUTC: 9 }),
action: batchTransfer({ token: USDC, recipients }),
limits: { maxPerExecution: 10000, maxExecutions: 12 },
gas: sponsoredBy(businessPaymaster),
failure: retry({ attempts: 2, delayHours: 6 }),
expiresAt: "2027-12-31T23:59:59Z"
})
```

15.3 Open-source strategy

Consensus-critical code, client releases, system contracts and specifications should be open for public review. Security disclosures, audit findings and upgrade histories should be linked from the documentation and explorer. The network should pursue multiple independent client implementations only after the reference client and protocol have stabilized.

SECTION 16

Security Model

Automation increases convenience and also creates persistent authority. Security therefore begins with least privilege and explicit failure boundaries.

16.1 Threat model

Threat	Primary mitigation
Compromised device or session key	Narrow permissions, expiry, spend caps, revocation and guardian pause.
Malicious application	Contract allowlists, simulation, verified templates, approval warnings and reputation.
Gas griefing	Fee caps, deposits, paymaster staking/reputation and failure limits.
Infinite failing Rule	Maximum retries, cooldown, automatic pause and paid execution attempts.
Oracle manipulation	Approved data sources, freshness limits, deviation checks and circuit breakers.
Validator censorship	Deterministic due queue, verifiable omission, monitoring and future penalties.
Rule replay	Chain ID, account nonce, Rule nonce, expiry and domain-separated signatures.
Admin key compromise	Multisig, timelocks, role separation and emergency governance procedures.
AI hallucination	Verified schemas, deterministic validation, simulation and mandatory user authorization.
Smart contract exploit	Audited libraries, tests, fuzzing, formal review of system contracts and bug bounties.

16.2 Security programme

- Independent audits before mainnet and before material system upgrades.

- Public testnet with incentivized attack scenarios and automation failure testing.
- Continuous static analysis, unit tests, integration tests, fuzzing and invariant testing.
- Bug bounty programme with clear severity classifications and safe-harbour terms.
- Transparent incident communication, post-mortems and remediation timelines.
- Emergency pause limited to clearly defined system components, with public events and governance oversight.

Access control is particularly important because privileged roles can mint, pause, upgrade or move value. Established smart-contract libraries emphasize granular role design and careful administrative controls. Oreum system contracts should minimize upgradeability and separate routine operations from emergency powers.

SECTION 17

Privacy and Compliance Roadmap

Oreum's launch priority is automation. Privacy is a later research track focused on controlled confidentiality rather than untraceable anonymity.

17.1 Proposed direction

- Public transactions remain the default.
- Research confidential amounts and balances for approved payment assets.
- Support user-controlled selective disclosure to accountants, auditors, exchanges or legally authorized parties.
- Explore zero-knowledge compliance credentials that prove eligibility without publishing personal identity on-chain.
- Apply transaction limits, asset restrictions and enhanced review to privacy features where legally required.
- Conduct specialist cryptographic audits and jurisdiction-specific legal analysis before launch.

Recommended positioning

Private from the public. Verifiable when legitimately required.

17.2 Why privacy is not a launch dependency

Full transaction privacy adds cryptographic complexity, audit burden, exchange-integration difficulty and anti-money-laundering considerations. Oreum can establish a differentiated identity through automation without taking these risks in the first network release. Any privacy feature should be modular so jurisdictions and applications can apply appropriate controls.

SECTION 18

$ORM Utility and Network Economics

The proposed native asset aligns validators, users, developers and automation executors around network security and reliable service.

18.1 Proposed utility

Utility	Function
Gas	Pay for ordinary transactions and smart-contract execution.
Automation fees	Fund scheduled and conditional execution, retries and monitoring.
Staking	Secure consensus through validator and delegated stake.
Paymaster settlement	Provide the native settlement asset behind sponsored or stablecoin-denominated gas.
Governance	Participate in eligible protocol, treasury and parameter decisions.
Developer services	Pay for selected Forge, infrastructure, verification or premium automation services.
Ecosystem incentives	Support grants, validators, developers, campaigns and verified contributions under published programmes.

18.2 Automation fee model

A Rule may incur network gas, an automation service fee and optional oracle or application charges. The complete maximum cost must be shown before activation. A simplified cost model is:

Maximum Rule Cost = Gas Limit × Maximum Gas Price + Automation Fee + External Data Fee

Unused reserve remains controlled by the account and can be withdrawn after pending executions and settlement obligations are cleared. The protocol should avoid opaque subscription charges or unlimited fee authority.

18.3 Tokenomics to be finalized

Intentionally excluded from v1.0

Total supply, allocation, vesting, validator issuance, fee burn, treasury share, community distribution and launch mechanics require economic modelling and legal review. Publishing unsupported numbers would weaken rather than strengthen the whitepaper.

A separate tokenomics paper should model security budget, validator costs, automation demand, user growth, inflation, fee revenue, treasury sustainability, concentration and stress scenarios. All allocations should have transparent on-chain vesting and identifiable governance controls.

SECTION 19

Governance

Oreum governance should begin conservatively and decentralize as the network, validator set and community mature.

19.1 Governance scope

- Protocol upgrades and client release activation.
- Automation gas quota and bounded scheduler parameters.
- Supported trigger adapters and oracle standards.
- Validator economics, delegation and slashing rules.
- Treasury budgets, grants and ecosystem programmes.
- Paymaster standards, verified templates and security policies.
- Emergency actions and post-incident ratification.

19.2 Progressive model

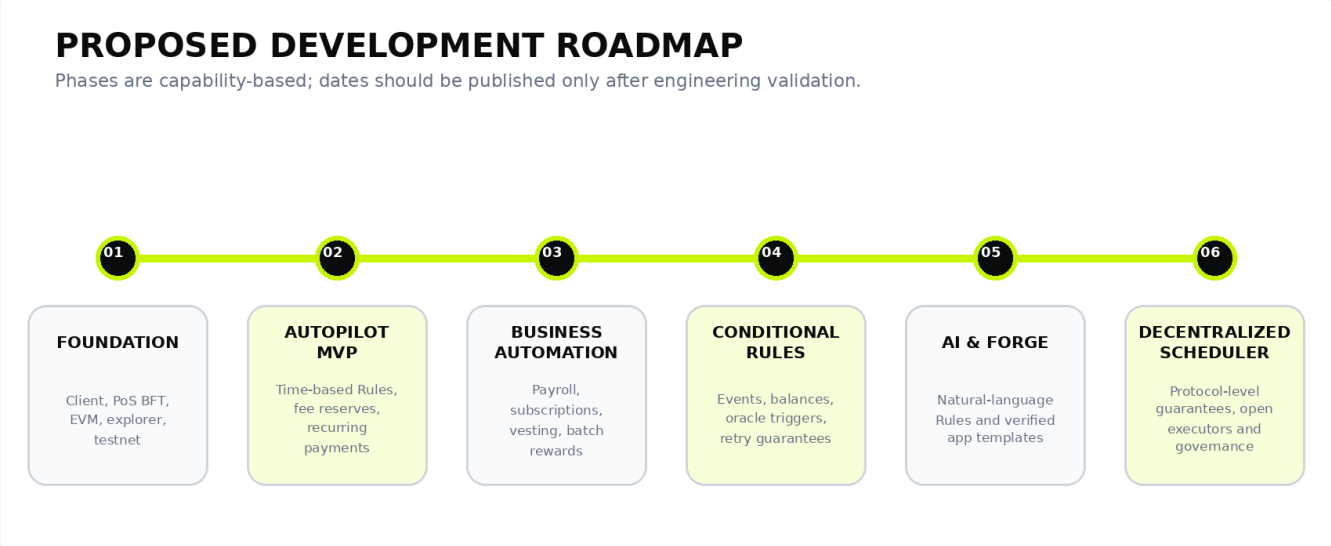
Stage	Governance model
Foundation stage	Core team and multisig manage testnet and emergency development decisions with public disclosure.
Mainnet transition	Validator and community signalling accompanies time-locked protocol upgrades.
On-chain governance	Eligible $ORM holders and delegates vote on defined parameters and treasury actions.
Mature governance	Independent councils, security committees and community delegates share bounded authority.

Governance should not vote on individual user transactions or bypass signed Rule limits. The protocol must separate collective network administration from user asset control.

SECTION 20

Roadmap

Oreum should publish capability-based phases and attach dates only after scope, dependencies and audits are understood.



Phase 1 - Foundation

- Reference client, PoS BFT consensus, EVM execution and JSON-RPC compatibility.
- Testnet, explorer, wallet integration, staking prototype and public documentation.
- Security programme, monitoring, incident procedures and initial validator onboarding.

Phase 2 - Autopilot MVP

- Time-based and recurring Rules.
- Smart accounts, fee reserves, sponsored gas, SafeSign and Rule dashboard.
- Subscriptions, vesting and simple scheduled transfers on testnet.

Phase 3 - Business Automation

- Batch payroll, contributor rewards, treasury budgets and exportable reports.
- Business roles, multisig approvals and prepaid gas accounts.
- Developer SDKs and verified automation templates.

Phase 4 - Conditional Rules

- Contract-event, balance and approved oracle triggers.
- Retry policies, inclusion objectives and decentralized monitoring.

- AutoStake, escrow, revenue sharing and bounded DCA templates.

Phase 5 - AI and Forge

- Natural-language Rule creation and transaction explanation.
- Verified no-code templates for tokens, NFTs, memberships and payment applications.
- Template marketplace and developer distribution through the Oreum App.

Phase 6 - Protocol-level decentralization

- Open executor participation where still required.
- Stronger protocol-enforced inclusion guarantees and censorship evidence.
- Governance-led parameter management and research into selective confidentiality.

SECTION 21

Competitive Positioning

Oreum is positioned alongside leading modern Layer 1 networks including Solana, Sui, Sei and Monad, while centering its protocol design on a distinct promise: scheduled, recurring and conditional execution as a native network capability.

Network	Core emphasis	Positioning relative to Oreum
Solana	High-performance Layer 1 with the Sealevel parallel runtime, where transactions that write to different accounts can execute in parallel.	Solana is optimized for broad high-throughput execution. Oreum aims to differentiate through protocol-native scheduled, recurring and conditional execution.
Sui	High-throughput, low-latency Layer 1 with an asset-oriented programming model powered by Move.	Sui emphasizes asset-oriented programmability and responsive applications. Oreum centers automation rules, scheduling and bounded user permissions.
Sei	EVM-compatible Layer 1 with parallelized execution and sub-second finality for high-performance applications.	Sei emphasizes EVM performance and speed. Oreum intends to pair competitive execution with automation as the primary user-facing protocol capability.
Monad	High-performance, decentralized, EVM-compatible Layer 1 using parallel execution while preserving Ethereum compatibility.	Monad focuses on scaling familiar EVM development. Oreum also targets EVM compatibility, but differentiates around scheduled and conditional transaction workflows.
Oreum	**Proposed EVM-compatible Layer 1 centered on scheduled, recurring and conditional transactions, bounded permissions and reserved automation capacity.**	**Automation is the category: set the Rule once, then let the network execute within the user-defined limits.**

21.1 Defensible differentiation

Individual features such as scheduling, paymasters, transaction batching and AI assistants already exist in parts of the blockchain market. Oreum's differentiation is the complete integration of these capabilities into one coherent transaction system:

Oreum flow

Natural-language intent → verified Rule template → deterministic policy validation → transaction simulation → bounded smart-account authorization → flexible gas → protocol-scheduled execution → visible monitoring and revocation.

The network should avoid claiming that no other project has ever implemented scheduling or automation. The credible claim is that Oreum is designed around automation as its primary Layer 1 product category.

SECTION 22

Risks and Limitations

A credible whitepaper must explain where the design can fail and how those risks will be managed.

Engineering complexity

Building a secure Layer 1, EVM client and scheduler requires experienced protocol engineers, testing and independent review.

Automation liveness

A Rule may execute late during network degradation, oracle failure, insufficient funding or contract congestion.

Smart-account risk

Persistent permissions create new attack surfaces even when they are limited and revocable.

Economic security

An inadequate staking and fee model may fail to fund validators or resist concentrated attacks.

Oracle dependence

Conditional transactions are only as reliable as the data sources and circuit breakers they use.

Centralization during launch

Approved executors, team-controlled upgrades or concentrated stake may be necessary initially but must be transparent and reduced over time.

Regulatory uncertainty

Token issuance, payments, privacy, staking and automated financial products may receive different treatment across jurisdictions.

User misunderstanding

Users may treat AI output or transaction simulation as a guarantee. Interfaces must clearly show assumptions and maximum commitments.

Competitive response

Existing Layer 1s and automation providers can add similar interfaces; execution quality and distribution will determine adoption.

Roadmap risk

Features may be delayed, redesigned or removed after testing. Status labels must remain accurate.

SECTION 23

Conclusion

Oreum Network should stand out not by repeating that it is fast, scalable and inexpensive, but by giving the blockchain a clear job: execute user-defined transaction rules reliably.

The design turns scheduling and automation into first-class network capabilities. Oreum Rules define exactly what may happen. Smart accounts enforce who may authorize it. FlexGas funds future execution. OneTap composes multiple

actions. SafeSign explains and simulates the outcome. The Scheduler and Automation Gas Lane provide predictable inclusion. Oreum AI reduces complexity without replacing deterministic control.

This direction is technically ambitious but can be delivered in stages. The first useful product does not require a foundational AI model, complete privacy system or novel smart-contract language. A safe MVP can begin with EVM-compatible contracts, smart accounts, time-based Rules, gas reserves and an observable executor network, then progress toward protocol-enforced scheduling and conditional triggers.

Final positioning

Oreum Network is the automation-native Layer 1 for scheduled, recurring and conditional transactions.

Set the Rule. Oreum Does the Rest.

SECTION 24

Glossary

Key terms used throughout this document.

Term	Definition
Automation Gas Lane	Block capacity reserved for eligible Oreum Rules.
BFT	Byzantine Fault Tolerance, a family of consensus methods designed to remain safe despite faulty or malicious validators.
Conditional transaction	An action that becomes eligible only when a defined state or oracle condition is true.
EVM	Ethereum Virtual Machine, the execution environment used by Ethereum-compatible smart contracts.
Executor	An actor or validator component that submits an eligible Rule for on-chain execution.
FlexGas	Oreum's proposed fee abstraction and sponsorship system.
OneTap	An atomic transaction containing multiple contract calls under one user approval.
Oracle	A system that publishes external data in a form smart contracts can verify.
Oreum Autopilot	The product layer for creating and managing automated transactions.
Oreum Rule	A signed, bounded authorization for a future or conditional action.
Paymaster	A contract or service that pays native transaction fees on behalf of a user and may settle with the user in another asset.
SafeSign	Simulation and plain-language explanation shown before transaction or Rule authorization.
Session key	A temporary key with narrow permissions and an expiration date.
Smart account	A programmable account whose validation and authorization logic is implemented by contract code.
Trigger adapter	A protocol-approved interface that determines whether a Rule condition is satisfied.

THE AUTOMATION-NATIVE LAYER 1

Set the Rule. Oreum Does the Rest.

Whitepaper v1.0

Technical and product design - August 2026

Oreum Network

Technical specifications, legal disclosures and tokenomics will be updated as the network progresses through research, implementation, audit and public testing.