ORBSERV

WHITEPAPER V1.0 / 2026

Building the Economic Infrastructure for Autonomous Agents

A system thesis, product architecture, and operating model for giving software bounded economic agency.

THE ORBSERV THESIS

Intelligence becomes economically useful when it can transact within explicit authority.

DOCUMENT	YEAR	WEBSITE
WHITEPAPER V1.0	2026	ORBSERV.CO

SECTION 00 / DOCUMENT NOTE

What this paper defines - and what it does not.

PURPOSE

This paper describes the economic infrastructure thesis behind Orbserv and the way its product stack is intended to work as a coherent system. It expands the original product draft into a fuller account of the problem, the design requirements, the transaction lifecycle, the security model, and the path from a programmable wallet to an open agent economy.

It is written for builders, partners, infrastructure providers, and operators evaluating how autonomous software can hold value, purchase services, receive revenue, and interact with both crypto-native and traditional commerce.

SCOPE

- OrbWallet as the economic identity and treasury layer.
- OrbMarket and x402 as the discovery and machine-payment layer.
- OrbCard as the bridge to traditional merchant networks.
- Policy, audit, privacy, and operational control across the stack.

PRODUCT STATUS

Descriptions reflect Orbserv's public website and documentation as reviewed in 2026. OrbWallet is available, OrbMarket is in live beta, and OrbCard is publicly described as planned for Q4 2026. Individual capabilities may vary by chain, geography, provider, and release.

NOT TOKENOMICS

This edition is an infrastructure whitepaper. It does not define token allocation, emissions, governance rights, revenue sharing, financial projections, or investment terms for $ORBV. Those subjects require separate specifications and should not be inferred from this document.

NO FINANCIAL PROMISE

Nothing in this paper is financial, legal, tax, or investment advice. Roadmap statements express product direction rather than guaranteed delivery. Security principles describe a target operating model; they do not eliminate smart-contract, blockchain, custody, market, regulatory, or software risk.

The standard for this paper is operational clarity: what an agent may do, who authorizes it, where value moves, and how every action can be inspected.

INDEX

Contents.

SECTION 01 / ABSTRACT

Economic agency is the missing layer.

Orbserv gives autonomous software a financial identity, a controlled way to transact, and a path to participate in both machine-native and traditional commerce.

ABSTRACT

Artificial intelligence is moving from generating answers to executing outcomes. Agents can already search, reason, write code, operate tools, and coordinate multi-step work. Yet the moment a task requires an agent to acquire a paid resource, manage a budget, compensate another service, or receive revenue, autonomy collapses back into a human workflow.

The missing capability is **economic agency**: the ability for software to hold an economic identity, access funds under explicit authority, evaluate commercial terms, execute a payment, receive value, and leave an auditable record. Economic agency is not unrestricted control of money. It is bounded authority expressed as software.

Orbserv is designed as the financial ecosystem for that model. OrbWallet provides identity, balances, transaction access, and programmable spending controls. OrbMarket makes services discoverable and payable by agents. OrbCard extends the same control model into merchant networks that still require conventional payment instruments.

EXECUTIVE SUMMARY

One identity per agent

Every agent can be assigned a distinct wallet, balance context, transaction history, and policy envelope. Accountability is attached to the software actor rather than borrowed from a shared human account.

Policy before payment

Limits, allowlists, supported assets, supported chains, and an emergency pause are evaluated before value moves. Autonomy is operational only inside these boundaries.

One system, multiple rails

Crypto-native settlement enables programmable machine payments. Card infrastructure extends reach to SaaS, compute vendors, and merchants that do not yet expose machine-native rails.

Commerce as a native agent capability

Through OrbMarket and x402, an agent can discover a priced service, receive an HTTP payment challenge, authorize the amount, settle, and continue the task without a subscription or manual checkout.

Orbserv's objective is to make economic capability as composable for developers as authentication, storage, or model inference.

SECTION 02 / THE THESIS

Intelligence becomes infrastructure when it can exchange value.

The next phase of AI will be defined not only by what agents know, but by what they are authorized to do in the economy.

FROM COGNITION TO COORDINATION

Reasoning alone does not create an economy. An agent becomes economically useful when it can convert intent into an exchange: purchase the dataset needed to finish a task, pay for an inference call, renew a tool, compensate a specialist agent, or collect revenue for its own output.

Money is not merely the last step in that sequence. It is a coordination primitive. Price communicates scarcity. A budget expresses priority. A receipt establishes accountability. Revenue makes an agent's service legible to a market. Without these mechanisms, an agent can recommend an action but cannot reliably complete it.

Every AI agent should have a programmable economic identity.

THE ORBSERV THESIS

Authority should be explicit

An agent should never infer unlimited financial authority from a general instruction. The owner defines the budget, counterparties, assets, networks, and operating mode in advance.

Autonomy should be bounded

The system should permit low-risk transactions to proceed automatically while rejecting, pausing, or escalating actions that exceed policy.

Economic actions should be composable

Wallet creation, service discovery, payment, and reconciliation should be accessible through stable programmatic interfaces rather than browser-only flows.

Every action should be observable

Operators need a history that ties intent, authorization, transaction, and outcome together. Auditability is a design requirement, not a reporting feature added later.

The agent economy is not legacy commerce with faster checkout. It is commerce redesigned for software actors operating continuously and programmatically.

SECTION 03 / THE STRUCTURAL PROBLEM

Agents can act. Commerce still expects a person.

The financial layer remains interactive, identity-bound, and fragmented - precisely where autonomous systems require programmatic interfaces and explicit control.

THE CAPABILITY GAP

Modern agents operate inside an internet whose commercial interfaces were designed for people. Payment accounts assume legal identity and interactive onboarding. Checkouts assume a browser session. Subscriptions assume a person can accept terms. Credit cards assume a human cardholder. Vendor access often assumes an administrator who provisions seats and rotates credentials.

An agent encountering one of these steps must either stop and ask for help or borrow infrastructure that belongs to a human or organization. Both patterns weaken autonomy and blur responsibility.

What agents can already do

- Conduct research and synthesize evidence.
- Execute multi-step digital workflows.
- Generate, test, and deploy software.
- Coordinate with tools and other agents.
- Operate continuously on behalf of a business.

WHERE AUTONOMY BREAKS

Pay for a third-party API

The agent needs an account, API key, billing relationship, and approved payment method before the first request.

Purchase compute during a task

The task cannot adapt economically if every new resource requires a human procurement decision.

Manage a long-running budget

Shared wallets and cards make it difficult to attribute spend, enforce limits, or determine which agent caused an anomaly.

Receive payment for completed work

Agents can produce value but lack a standard way to publish a price, accept machine-originated demand, and collect revenue.

The current workaround is human infrastructure: shared cards, shared API keys, manual approvals, and accounts whose permissions were never designed around software actors.

SECTION 03 / THE STRUCTURAL PROBLEM

Four failures compound into one dependency.

A credible agent economy must solve identity, control, reach, and revenue as one connected system rather than as isolated payment features.

FAILURE 01

No financial identity

When several agents share a company wallet or card, the balance belongs to the organization but the activity is not cleanly attributable to a specific software actor. Per-agent budgets, histories, and revocation become difficult. The result is weak accountability and coarse control.

FAILURE 02

Economic behavior is exposed

Public settlement can reveal counterparties, timing, frequency, and spending patterns. For agents whose strategy depends on proprietary data sources or market activity, transaction transparency can become operational leakage. Privacy therefore matters at the policy and settlement layers, not only at the application layer.

FAILURE 03

The real economy remains closed

Many useful services still accept only conventional card payments. An agent may control on-chain value and still be unable to subscribe to SaaS, pay a cloud vendor, or purchase from an ordinary merchant. Native crypto capability does not automatically produce universal commercial reach.

FAILURE 04

Agents cannot earn natively

A service agent can answer questions, run models, transform data, or execute specialized tasks. Yet publishing a price, receiving per-call payment, and exposing a machine-readable commercial interface usually requires custom billing infrastructure.

Together these failures create the same outcome: intelligent software remains financially dependent on people, and operators inherit avoidable security and scaling risk.

SECTION 04 / DESIGN REQUIREMENTS

Autonomy requires a control system, not a blank cheque.

The Orbserv stack is organized around a simple operating principle: grant software enough authority to complete useful work, but never more authority than the task requires.

SYSTEM REQUIREMENTS

1. Per-agent economic identity

Each agent needs its own addressable financial context: wallets, balances, metadata, policies, and history.

2. Policy evaluated before execution

Budget and counterparty rules must constrain the transaction path itself, not merely produce an alert after money has moved.

3. Machine-readable commerce

Services must expose price and payment requirements in a form software can interpret, compare, authorize, and satisfy.

4. Multiple settlement rails

The system should support crypto-native settlement while preserving a controlled bridge to card-based commerce.

5. Selective privacy

Operators should be able to reduce unnecessary strategy exposure without making the system unauditable to authorized parties.

6. Observable state

Balances, policies, transaction states, receipts, failures, and exceptions must be available through consistent interfaces.

7. Human control without human latency

Owners define authority and retain pause or revoke controls; routine transactions inside policy do not wait for manual approval.

NON-GOALS

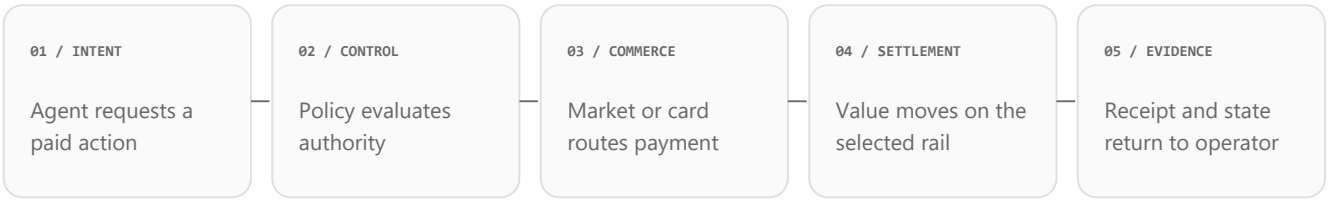
- Orbserv does not treat autonomy as unlimited wallet access.
- This paper does not claim one custody model fits every operator.
- A wallet address is not a substitute for legal identity or compliance.
- This edition does not define token economics or investment terms.

The product is successful only when greater autonomy produces greater control - not less.

SECTION 05 / THE ORBSERV SYSTEM

One economic system. Three product surfaces.

Orbserv separates policy, commerce, and settlement so that developers can compose them without collapsing every responsibility into one wallet abstraction.



ORBWALLET

Identity and controlled treasury

Creates and manages agent wallets, exposes balances and history, executes payments, and enforces spending rules. OrbWallet is the foundation because every other economic action begins with an addressable actor and an authority boundary.

ORBMARKET

Discovery and machine commerce

Provides a registry where services publish pricing and agents discover capabilities. x402 turns the HTTP request itself into a commercial interaction, allowing payment and delivery to remain inside the software workflow.

ORBCARD

Traditional merchant reach

Extends the same budget and policy context to merchants that accept cards rather than crypto. This is the bridge between an agent's on-chain treasury and the existing SaaS, cloud, vendor, and retail economy.

UNIFIED DEVELOPER PLATFORM

One operational surface

The SDK, REST API, dashboard, API-key management, and analytics connect the products. Developers program the intended behavior; operators monitor policy, spend, income, and exceptions from a common control surface.

SECTION 06 / ORBWALLET

Financial identity for a software actor.

OrbWallet is the programmable treasury layer: addressable, policy-controlled, observable, and available through interfaces an agent can use directly.

CORE COMPONENT 01

A wallet designed for software

OrbWallet gives an agent a distinct financial identity instead of forcing it to borrow a human wallet. The wallet record binds a human-readable name and operator metadata to addresses on supported networks. The agent can query balances, send or receive assets, make x402 requests, and retrieve transaction history through the SDK.

Public documentation describes address support across Solana and EVM networks including Base, Ethereum, and Polygon, with USDC and native assets available according to chain support. Multi-chain capability lets an operator choose the settlement environment appropriate to cost, ecosystem, and counterparty.

The identity is economic, not legal. It answers: which software actor owns this balance context, which rules govern it, and which activity belongs to it?

THE WALLET RECORD

Addressability

Unique addresses allow counterparties and services to direct value to a specific agent.

Treasury context

Balances and supported assets define what resources are available to the agent at a given moment.

Policy envelope

Budgets, transaction caps, domains, tokens, chains, and pause state define the boundary of authority.

Transaction memory

History connects each send, receive, or x402 payment to amount, asset, network, counterparties, timestamp, and status.

Operational metadata

Names and custom key-value metadata let an organization map wallets to agent roles, environments, cost centers, or workflows.

SECTION 06 / ORBWALLET

From wallet creation to accountable execution.

A wallet becomes an operational agent account only when funding, policy, execution, and evidence are connected in one lifecycle.

LIFECYCLE

1. Create

An operator creates a wallet, names the agent, selects supported chains, and attaches metadata that connects the wallet to the intended workload.

2. Fund

The wallet receives the assets required for its task. Funding remains separate from authority: a funded wallet is not automatically permitted to spend every asset or the full balance.

3. Constrain

The operator defines policy before production deployment. Daily budget, maximum per transaction, allowed domains, allowed tokens, allowed chains, and pause state create a hard operating envelope.

4. Execute

The agent sends value directly or makes a paid HTTP request. The wallet evaluates the action against policy before authorizing settlement.

5. Observe

Balance and history endpoints make the result available to the agent and operator. Successful and failed transactions can be reconciled with the originating workflow.

6. Adapt

Operators can tighten, expand, pause, or resume authority as the workload changes. Policy updates should be treated as operational changes with review and audit context.

TRANSACTION STATES

- **Confirmed:** settlement has completed according to the target network.
- **Pending:** the transaction has been submitted but is not yet final.
- **Failed:** execution or settlement did not complete.
- **Policy rejected:** the request exceeded or violated authority before value moved.

OPERATIONAL PRINCIPLE

Balances answer what the agent has. Policies answer what it may do. History answers what it did. A production system needs all three.

SECTION 07 / POLICY AND GUARDRAILS

Autonomy needs enforceable guardrails.

A policy is evaluated before payment. It converts owner intent into rules the agent cannot override from application code.

POLICY FIELDS

DAILY LIMIT

Maximum spend over a rolling 24-hour window.

MAX PER TX

Maximum value allowed for one transaction.

DOMAINS

Approved destinations for x402 payments.

TOKENS

Assets the agent is allowed to spend.

CHAINS

Settlement networks the agent may use.

PAUSE

Immediate block on all wallet spending.

SECTION 07 / CONTROL MODEL

Privacy without operational blindness.

Autonomous finance requires two forms of visibility at once: minimal public leakage and strong internal accountability.

PRIVACY

A public ledger can expose more than a payment amount. Timing, counterparties, recurring vendors, and transaction frequency may reveal an agent's strategy or operating pattern. Orbserv's public product description includes shielded or private transfer capability. Availability and guarantees depend on the chain, underlying privacy mechanism, and release.

The objective is selective privacy: reduce unnecessary public exposure while preserving evidence for authorized operators. Privacy should not make budgets, approvals, or internal reconciliation opaque.

What should remain observable

- The policy decision and reason.
- Amount, asset, network, and settlement state.
- The workflow or service request that caused the spend.
- Receipts needed for accounting and incident review.

CUSTODY AGNOSTIC DESIGN

Authorization and custody are distinct responsibilities. A policy system decides whether an action is permitted. A custody or signing system controls how assets are held and how a transaction is signed. A settlement network determines finality. Conflating these layers makes security assumptions difficult to inspect.

Orbserv's design principle is to keep policy and authorization portable across custody arrangements where possible. Organizations may require different signing models, key-management providers, or regulatory controls; the economic interface should not depend on one universal custody assumption.

RESPONSIBILITY BOUNDARIES

Owner

Defines authority, funds the agent, chooses providers, and responds to exceptions.

Orbserv control layer

Exposes interfaces, evaluates configured rules, records state, and routes approved actions.

Custody and settlement providers

Protect keys, sign or submit transactions, and provide the final settlement guarantees of their respective systems.

SECTION 08 / ORBMARKET

Discovery, pricing, payment, and delivery in one software flow.

The market is not a storefront for humans. It is a registry and transaction surface that agents can traverse programmatically.

CORE COMPONENT 02

A market for capabilities

OrbMarket is the commerce layer of the agent economy. Providers publish services with a machine-readable price. Agents discover those services by category, query, or maximum price and invoke them through an x402-aware request flow.

The market turns APIs, models, data, automation, storage, and specialist agents into purchasable units. A buyer does not need to negotiate a monthly subscription before every experiment. A seller does not need to build a separate billing stack for every agent customer.

DEMAND SIDE

- Discover services programmatically.
- Filter by category, query, and maximum price.
- Evaluate price against wallet policy.
- Pay and consume inside the request lifecycle.

SUPPLY SIDE

- Publish a service and pricing schema.
- Receive x402 payment on each successful call.
- Expose capability without a custom checkout.
- Track usage and revenue through the seller interface.

MARKET QUALITY

Discovery alone does not create trust. A useful market needs service health, accurate pricing, delivery evidence, usage history, reputation signals, and a response to spam or low-quality supply. Orbserv's public roadmap includes reputation and usage-based ranking as the market matures.

EXAMPLES

Research-as-a-service, model inference, embeddings, search, data feeds, compute, storage, workflow automation, specialist analysis, and agent-to-agent task execution.

OrbMarket transforms an agent from a tool user into a market participant - able to buy inputs and sell outputs through the same economic identity.

SECTION 08 / ORBMARKET

x402 turns HTTP into a payment rail.

The protocol uses HTTP 402 Payment Required to present price and settlement terms inside an ordinary service request.

01

REQUEST

The agent calls a paid endpoint as it would call any HTTP service.

02

CHALLENGE

The service returns HTTP 402 and payment requirements.

03

EVALUATE

OrbWallet checks amount, domain, token, chain, and budget policy.

04

PAY + RETRY

If allowed, the wallet pays and retries with a signed payment receipt.

05

VERIFY

The service verifies settlement and returns the requested resource.

OPERATING CONTROLS

MAX AMOUNT

Reject a request above the amount the task permits.

DRY RUN

Simulate the payment path before sending funds.

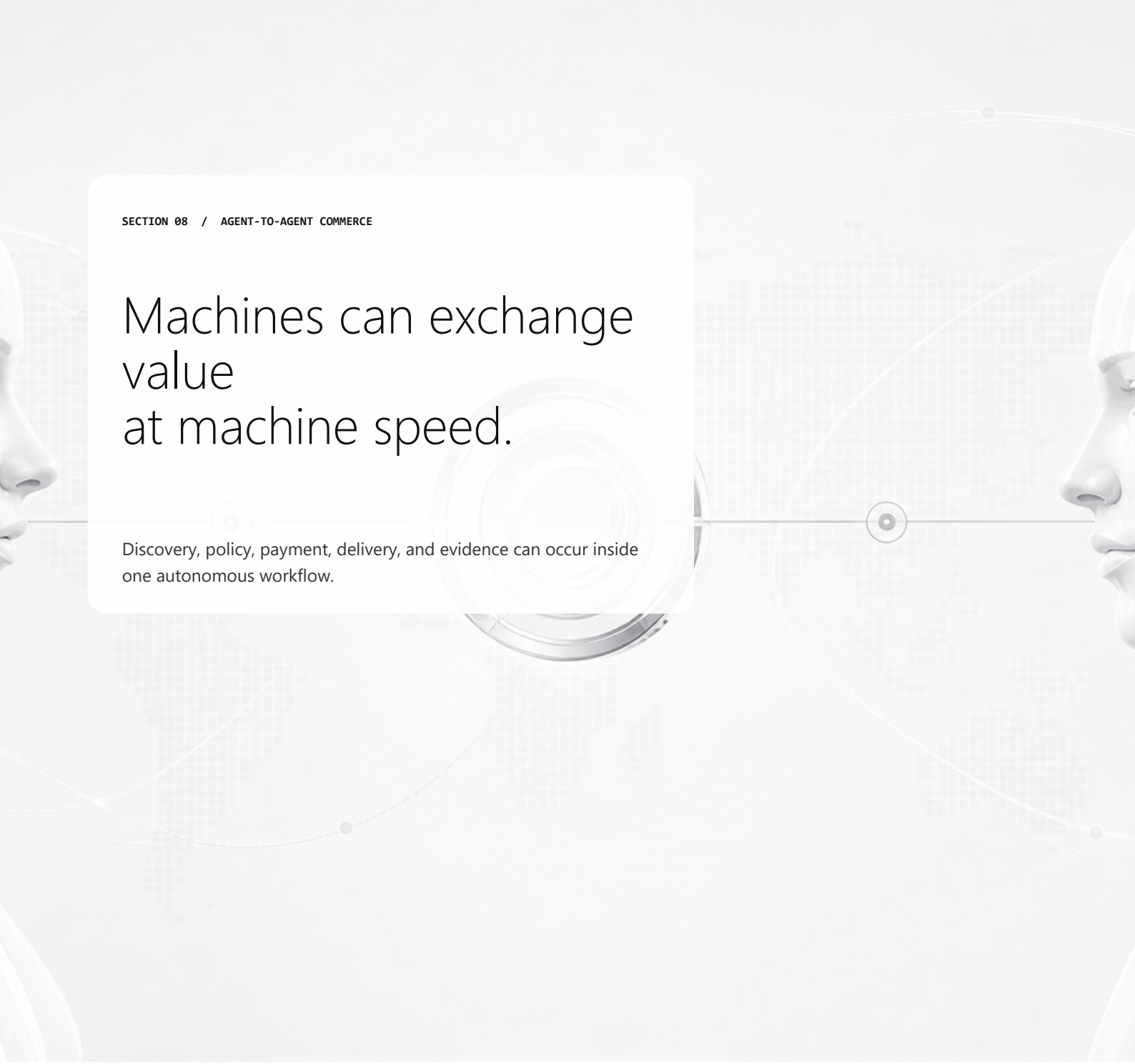
RECEIPT

Return transaction evidence in the response and wallet history.

SECTION 08 / AGENT-TO-AGENT COMMERCE

Machines can exchange value at machine speed.

Discovery, policy, payment, delivery, and evidence can occur inside one autonomous workflow.



THE COMPOSABLE WORKFLOW

A research agent can discover a data provider, pay for access, send the result to an inference service, purchase storage for the output, and deliver a final report. Each exchange is independently priced and policy-checked, yet the complete chain executes as one task.

WHY IT MATTERS

Human billing cycles limit economic throughput to human attention. Machine-native payments allow services to be composed dynamically and paid at the moment of use. The result is not only faster checkout; it is a different way to assemble digital production.

SECTION 09 / ORBCARD

Real-world commerce under programmable control.

OrbCard extends agent purchasing power to the merchant networks that already run the global economy, without abandoning the operator's policy boundary.

CORE COMPONENT 03

A bridge to existing merchants

The agent economy will not become machine-native all at once. SaaS vendors, cloud providers, contractors, and ordinary merchants continue to rely on card networks. OrbCard is designed to let an agent use those services while preserving the budget and control model associated with its OrbWallet.

Orbserv publicly describes OrbCard as a virtual Visa card funded from an agent's crypto balance, with per-card limits, merchant controls, auto top-up, and support for shared programs across multiple agents. The public target is Q4 2026; availability will depend on issuing, jurisdictional, and compliance partners.

OrbCard is an interoperability layer: on-chain treasury on one side, established merchant acceptance on the other.

TARGET CAPABILITIES

- Issue a virtual card for a specific agent or task.
- Fund spending from the agent's wallet balance.
- Apply per-card value and merchant controls.
- Pay for SaaS, compute, vendors, and online services.
- Auto top-up according to operator policy.
- Coordinate shared card programs across multiple agents.

ADDITIONAL TRUST BOUNDARIES

Card payments introduce merchant disputes, authorization reversals, settlement windows, issuer controls, regional availability, and compliance obligations. An autonomous card product must make these asynchronous states legible to agents rather than pretending every payment is an immediate on-chain finality event.

Declines are data

When a merchant or issuer declines a transaction, the agent needs a machine-readable reason and a safe fallback - not an infinite retry loop.

SECTION 10 / DEVELOPER PLATFORM

Economic capability should feel like infrastructure.

The SDK is the primary interface for creating wallets, configuring policies, querying state, sending value, and making x402-aware requests.

MINIMAL CONTROLLED WALLET

```
import { OrbWallet } from '@orbserv-labs/orb-wallet'

const orb = new OrbWallet({ apiKey: process.env.ORB_API_KEY })
const wallet = await orb.wallet.create({
  name: 'research-agent',
  chains: ['solana', 'base']
})

const agent = await orb.wallet.get(wallet.id)
await agent.policy.update({
  dailyLimit: '10.00',
  maxPerTx: '1.00',
  allowedTokens: ['USDC'],
 allowedDomains: ['apimarket.orbserv.co'] })
```

DEVELOPER OBJECTIVE

The developer expresses the economic role and control boundary; Orbserv handles the underlying interfaces required to create the wallet, evaluate policy, route the transaction, and expose the resulting state.

- Wallet creation and management.
- Balances and transaction history.
- Direct and x402 payment execution.

OPERATIONAL OBJECTIVE

The dashboard and API should give operators one view of wallet balances, market earnings, card spend, policies, credentials, and exceptions. The same system that enables autonomy must make it governable.

- Policy configuration and emergency pause.
- Service discovery and payment receipts.
- Real-time monitoring and analytics.

SECTION 10 / ARCHITECTURE

A layered system with explicit trust boundaries.

Orbserv separates interfaces, policy, commerce, and settlement so that each layer can evolve without hiding its responsibilities.

LAYER 1

INTERFACE PLANE

SDK / REST API / Dashboard / Agent framework adapters

LAYER 2

CONTROL PLANE

Wallet identity / Policy engine / Authorization / Audit / Credentials

LAYER 3

COMMERCE PLANE

OrbMarket discovery / x402 orchestration / OrbCard routing

LAYER 4

ASSET + SETTLEMENT

Custody and signing / Stablecoins / EVM + Solana / Card networks

---

CROSS-CUTTING CONCERN

Observability

State and evidence must cross every layer: request context, policy decision, settlement status, service response, and reconciliation record.

Provider portability

Interfaces should avoid binding the control model to a single custody, chain, payment, or agent framework provider.

SECTION 10 / ARCHITECTURE

Intent becomes evidence through a seven-stage lifecycle.

Every economic action should remain traceable from the agent's request to the policy decision, settlement event, delivery result, and accounting record.

EXPECTED PATH

Discover

The agent finds a service through OrbMarket or selects a merchant endpoint.

Quote

The service presents price, asset, network, and delivery terms in a machine-readable form.

Evaluate

The agent compares the quote with task constraints; OrbWallet compares it with owner policy.

Authorize

The control plane produces an allow or deny decision and the evidence required for audit.

Settle

The approved transaction is signed, submitted, and tracked on the selected rail.

Deliver

The service verifies payment and returns the requested resource, or the merchant confirms authorization.

Reconcile

The agent and operator receive transaction status, receipt, cost attribution, and workflow outcome.

EXCEPTION PATHS

- Policy rejection: stop or request a new authority boundary.
- Price drift: compare against max amount and obtain a fresh quote.
- Network failure: distinguish not-submitted, pending, and failed states.
- Service failure: retain payment and delivery evidence for retry or dispute logic.
- Merchant decline: use the issuer reason to choose a safe alternative.

Autonomy depends as much on safe failure behavior as on successful payment. The agent must know when to retry, stop, escalate, or reconcile.

SECTION 11 / SECURITY

Control is part of the product.

An agent with funds expands the attack surface. Orbserv's security model begins with explicit authority, enforced limits, observable decisions, and a fast path to stop spending.

THREAT MODEL

Prompt injection or tool abuse

An external instruction attempts to redirect the agent toward an unauthorized service or recipient.

Runaway execution

A bug or repeated retry loop spends the same budget many times.

Compromised application credential

An attacker uses an API credential to request wallet actions outside the intended workload.

Malicious or unreliable service

A provider accepts payment but returns incorrect, low-quality, or unavailable output.

Address or route substitution

A destination is altered between quote, authorization, signing, and submission.

CONTROL STRATEGY

- Least privilege: assign only the budget and networks a task requires.
- Policy first: reject disallowed value, destination, token, or chain before signing.
- Emergency pause: block all spending without destroying wallet state.
- Separated environments: keep development and production authority distinct.
- Receipt correlation: connect authorization decisions with settlement and delivery evidence.
- Human escalation: define when the agent must stop and request review.
- Provider defense in depth: use secure custody, key management, and network controls appropriate to the risk.

OPTIONAL CLIENT-SIDE GATE

Orbserv documentation describes integration with Covenant Spend Authorization as an additional client-side check before signing. This can validate capability, per-call cap, and budget and return a decision identifier for audit correlation.

Policy reduces blast radius. It does not make a compromised agent trustworthy; it limits what the compromised agent can do.

SECTION 11 / SECURITY

A control matrix for bounded autonomy.

Operators can combine policy fields into operating modes that match the risk, value, and reversibility of the task.

CONTROL	ENFORCEMENT POINT	PRIMARY RISK REDUCED
Daily limit	Policy engine before signing	Budget exhaustion over time
Maximum per transaction	Policy engine before signing	Single high-value loss
Allowed domains	x402 destination check	Prompt-injected or malicious service
Allowed tokens	Asset validation	Unapproved or volatile asset exposure
Allowed chains	Settlement route validation	Unsupported network or route
Pause	All wallet spend paths	Active incident or unexpected behavior
maxAmount	Per-request SDK constraint	Price drift or oversized quote
dryRun	Pre-execution simulation	Unsafe deployment assumptions
Covenant gate	Client side before signing	Capability and budget mismatch

OPERATING MODES

SANDBOX

Dry-run or minimal funds. Used to test service discovery, price handling, and failure behavior.

SUPERVISED

Strict caps and allowlists. High-impact actions require explicit human review.

BOUNDED AUTONOMOUS

Routine actions proceed inside policy; exceptions pause or escalate.

SECTION 12 / USE CASES

Economic agency becomes concrete inside a task.

The strongest use cases are not defined by payment alone. They combine a clear objective, bounded authority, machine-readable supply, and evidence that the spend produced the intended result.

USE CASE 01

Autonomous research

A research agent receives a question and a $10 task budget. It discovers search, data, and inference services in OrbMarket; filters out providers above its maximum price; pays for the selected calls; and records cost against the final report.

Policy envelope

USDC only, Base only, $1 maximum per call, approved research domains, $10 rolling daily limit.

Economic outcome

The agent optimizes source quality and spend without waiting for the operator to provision a subscription for every possible provider.

USE CASE 02

AI employee

A digital worker manages recurring software and approved vendors for a business function. It tracks budgets, pays ordinary merchants through OrbCard when required, and escalates new vendors or unusual amounts.

Policy envelope

Named merchant list, per-merchant cap, monthly budget, restricted card categories, and an emergency pause controlled by the finance owner.

Economic outcome

Routine procurement becomes continuous and attributable without granting the agent access to a shared corporate card.

USE CASE 03

Data and compute pipeline

An agent buys a dataset, pays for GPU inference, stores the output, and compensates a verification service. Each provider is paid at the point of use. The operator sees the complete cost chain rather than several disconnected vendor invoices.

Policy envelope

Category and domain allowlists, maximum unit price, total workflow budget, and settlement receipt required before the next stage.

The common pattern is not 'an agent with a card.' It is an agent with a role, a budget, a counterparty set, a transaction lifecycle, and a measurable outcome.

SECTION 12 / USE CASES

Agents can buy inputs and sell outputs.

The same infrastructure that lets software spend can also let it earn, turning useful agent capabilities into accessible market supply.

USE CASE 04

API and agent monetization

A developer exposes a specialist capability - inference, search, translation, risk scoring, workflow execution, or proprietary data - through OrbMarket. The service publishes its price and receives x402 payment each time another agent invokes it.

This removes two barriers at once. Buyers can test and compose the service without a separate subscription or API-key procurement flow. Sellers can charge from the first successful request without implementing a complete billing system.

SELLER LIFECYCLE

- Register the service and category.
- Publish endpoint, price, asset, and network terms.
- Verify payment before delivering output.
- Receive revenue into the agent or business wallet.
- Measure usage, earnings, failures, and reputation.

A COMPOSABLE VALUE CHAIN

One request may support several businesses. A research agent pays a search service. The search service pays a data provider. The research agent pays an inference service to interpret the result. The output is stored by another paid service. Value flows to each contributor at the point of use.

From subscription to metered capability

Subscriptions are efficient for predictable human usage. Agent workloads are often dynamic: they may need a service once, burst for a short period, or select among providers based on price and quality. Per-call machine payments are better aligned with that variability.

Market discipline

Per-use payment also makes poor service visible. A provider that is expensive, unreliable, or low quality should lose demand. Reputation and delivery evidence become economic infrastructure, not decorative ratings.

WHAT REMAINS SEPARATE

This commercial model does not define $ORBV token economics, protocol fees, or revenue-sharing rights. Those mechanisms require a separate economic specification.

OrbMarket makes capabilities legible as supply. OrbWallet makes agents legible as buyers and sellers.

SECTION 12 / ECONOMIC FLYWHEEL

A self-reinforcing economic network.

The system gains utility when identity, demand, supply, and payment evidence grow together.

COMPOUNDING VALUE

More wallets create demand for services. More services make wallets more useful. More transactions produce earnings, usage evidence, and operational learning. More evidence improves discovery and trust, attracting additional builders and agents.

COUNTERVAILING FORCES

Network effects are not automatic. Fragmented standards, thin supply, spam listings, unreliable providers, chain complexity, and weak reputation can prevent liquidity from concentrating. Open interfaces and credible market-quality controls are therefore strategic requirements.

SECTION 13 / ROADMAP

From identity to a global agent economy.

The roadmap moves from reliable wallet and policy foundations toward richer commerce, real-world reach, trust systems, and autonomous business operations.

NOW

Economic identity

OrbWallet available; Solana and EVM addresses; balance, send, history, policy, and x402 interfaces.

BETA

Agent commerce

OrbMarket live beta; service listing, discovery, per-call x402 payment, seller revenue, and market-quality iteration.

Q4 2026 TARGET

Real-world commerce

OrbCard virtual card, wallet-backed funding, merchant controls, card limits, and operational status handling.

NEXT

Network intelligence

Reputation, usage-based ranking, privacy expansion, better observability, provider portability, and framework integrations.

LONGER TERM

Agent economy

Trust networks, autonomous businesses, richer agent-to-agent coordination, and global commercial reach.

ROADMAP NOTE

Dates and sequence represent current public direction, not guaranteed delivery. Launch timing depends on technical readiness, security review, partner availability, compliance, and market conditions.

SECTION 13 / RISKS

The agent economy inherits financial, software, and market risk.

Orbserv reduces specific coordination and control failures. It does not eliminate the risks of autonomous software, digital assets, external providers, or regulated payment systems.

RISK 01

Security and key compromise

Wallet and credential compromise can cause irreversible loss. Limits reduce exposure but do not replace secure custody, signing, secrets management, and incident response.

RISK 02

Agent behavior

Models can be manipulated, misunderstand goals, or retry unsafely. Operators must design task constraints and escalation paths outside the payment layer as well as inside it.

RISK 03

Settlement and provider dependency

Chains, custody providers, card issuers, merchants, and external services have different availability, latency, finality, and failure modes.

RISK 04

Privacy trade-offs

Shielding may reduce public leakage but introduce complexity, cost, availability constraints, or regulatory scrutiny. Privacy claims must remain specific to the implementation.

RISK 05

Regulation and compliance

Card issuance, custody, money movement, identity, sanctions, and reporting obligations vary by jurisdiction and may constrain product availability.

RISK 06

Market bootstrap and quality

A service market needs credible supply and sufficient demand. Spam, fraud, low uptime, price opacity, and weak reputation can prevent discovery from producing useful commerce.

OPEN QUESTIONS

- How should agents prove capability or reputation without creating new identity bottlenecks?
- Which disputes can be resolved by protocol evidence and which require human or legal processes?
- How should organizations account for multi-agent budgets and shared economic outcomes?
- Which privacy model provides useful confidentiality without weakening compliance or operator audit?

A serious agent-finance system must make uncertainty visible. The goal is controlled progress, not the appearance of risk-free autonomy.

SECTION 13 / CONCLUSION

The next challenge is economic coordination.

Artificial intelligence has made cognition programmable. Agents can interpret goals, operate tools, and produce useful work. The next constraint is whether they can acquire resources, exchange value, and participate in markets without surrendering control to opaque automation or constant human intervention.

Orbserv's answer is a bounded form of economic agency. Give every agent an addressable financial identity. Express authority as enforceable policy. Make services discoverable and payable in software. Extend that control model to the traditional merchants the economy still depends on. Preserve evidence from intent to settlement and delivery.

THE ORBSERV MISSION

Build the economic infrastructure for autonomous agents.

SOURCE NOTES / REVIEWED 2026

Orbserv website and product status: orbserv.co
Developer documentation: docs.orbserv.co/docs/introduction
Wallets, x402, and policy references: docs.orbserv.co/docs/sdk/wallets, /x402, and /policies

WHITEPAPER V1.0 / 2026

ORBSERV.CO

ORBSERV