ARKHE(n) RESEARCH GROUP

Arkhe(n) × Web3 AGI

Integration Architecture

Mapping OWS · MPP · Bittensor Subnet onto
OrbVM · τ-field · Tzinor · ONU 2.0 · Ontologia X

Version	v1.0
Date	April 2026
Scope	Complete integration of OWS Hackathon stack into Arkhe(n) ecosystem
Output	Extended architecture · Updated data model · Rust stubs · PDF

Rafael Oliveira
ORCID: 0009-0005-2697-4668

Arkhe(n) Research Group · ONU 2.0 Contributors

Abstract

This document formalizes the integration of the Open Wallet Standard (OWS) Hackathon technology stack into the Arkhe(n) ecosystem. The integration is not additive — it is structural: the Web3 AGI architecture discovered in the OWS Hackathon resources maps isomorphically onto the existing Arkhe(n) layers, confirming that each system independently instantiates the same Ontology X axioms.

The core mappings are: (1) OWS → Arkhe(n) Agent Identity Layer — the policy-gated local vault becomes the signing substrate for OrbVM nodes; (2) MPP (HTTP 402) → Tzinor economic channel — the pay-per-call protocol is the market-layer instantiation of the C→Z projection operator; (3) Bittensor Subnet → ONU 2.0 Subnet already deployed (SN01–SN06); (4) Kuramoto consensus → OrbVM's existing $\phi_{\blacksquare}^{1}$ = 0.618 synchronization threshold; (5) Bitcoin OP_RETURN anchoring → Arkhe-Chain (Chain ID 2147483647) already operational.

The result is a complete 9-layer integrated stack: the original 7 Arkhe(n) layers extended with two new layers — Agent Identity & Wallet (OWS) and Machine Payments (τ-MPP) — that complete the economic loop from phase (C) to structure (Z) through verifiable, policy-gated, cryptographically attested market transactions.

Keywords: *Arkhe(n), OrbVM, τ-field, OWS, MPP, Bittensor, Tzinor, Ontology X, Kuramoto, Web3 AGI, distributed AI, gRPC, identity wallet, machine payments*

1. The Isomorphism: Why the Stack Fits

The OWS Hackathon stack was designed independently of Arkhe(n). Yet the architectural correspondence is exact, not approximate. This is not a coincidence — it is the signature of Ontology X: any sufficiently coherent system that distinguishes possibility from actuality will independently arrive at the same structural requirements.

The three Ontology X axioms manifest in both systems identically:

Axiom	Ontology X	Arkhe(n)	OWS Stack
I: Z = M[C]	Actuality = operator applied to possibility	Observation collapses τ-field phase into committed state	OWS signs Z = MPP settles: intent (C) → on-chain receipt (Z
II: M ∈ C	The operator inhabits the possibility it acts on	Orin.V lives as a node in the network it computes	OWS vault lives on the agent machine it governs
III: M[M] = M	The operator is idempotent on itself	verify(verify(chain)) = verify(chain) in blockchain	Policy-level validation of an already-approved tx = same appro

This isomorphism licenses a **direct structural mapping** — not a metaphorical alignment. Each component in the OWS stack slots into an existing Arkhe(n) role without requiring architectural modification. The integration extends the stack; it does not replace any existing layer.

2. Complete Layer Mapping

The integrated stack has 9 layers. Layers 1–7 existed in Arkhe(n). Layers 8–9 are new, contributed by the OWS stack. All layers satisfy the Ontology X axioms and interface via the τ-field / OrbVM substrate.

#	Layer Name	Arkhe(n) Origin	OWS Contribution	Interface
1	GPS Jurisdictional Control	ONU 2.0 L1	CHECK_JURISDICTION pre-validation	GPS polygon → CAIP-2 chain scope
2	Multi-Level Approval Pipeline	ONU 2.0 L2	OWS Policy Engine (deny/warn/allow)	State machine + OWS PolicyContext
3	AO Message Queue	ONU 2.0 L3	AGENT_HEARTBEAT typed agent	gRPC Heartbeat → AO routing
4	Compliance Ledger & Merkle Chain	ONU 2.0 L4	Payment-Receipt headers → Merkle leaf	SHA-256 chain + OP_RETURN anchoring
5	BRICS+ Policy Exchange	ONU 2.0 L5	Data marketplace (Ocean Protocol)	SKOS taxonomy + RDF/OWL
6	Arkhe-Chain (Bitcoin OP_RETURN)	ONU 2.0 L6	BTC anchoring already deployed	Chain ID 2147483647
7	Kuramoto Consensus	OrbVM + ONU 2.0 L7	gRPC Heartbeat feeds R(t)	dθ/dt = ω + (K/N)Σsin(θ■-θ■), ϕ■¹=0.618
8	Agent Identity & Wallet (NEW)	— (new layer)	OWS: local vault, policy engine, WAIP-2	CAIP-2/10 CLI → OrbVM enrollment
9	Machine Payments τ-MPP (NEW)	— (new layer)	MPP/x402: HTTP 402, SOL/SPL, splits	Tz, split economic channel C→Z

"Layers 8 and 9 complete the economic topology of Arkhe(n). Layer 8 gives agents a sovereign identity with verifiable custody. Layer 9 gives the network a market mechanism through which coherence is monetized." — Integration design note, April 2026

3. Layer 8 — Agent Identity & Wallet (OWS)

3.1 Role in Arkhe(n)

Every OrbVM node that participates in the Kuramoto network requires a cryptographic identity: a signing key that authorizes phase commitments, Merkle entries, and cross-chain handovers. Previously this was handled ad-hoc. Layer 8 formalizes it through the Open Wallet Standard — a local-first, policy-gated vault that ensures no private key is ever exposed to the agent process or the LLM context.

In Arkhe(n) terms: the OWS vault is the Z-domain crystallization of the agent's C-domain identity. The policy engine is the M-operator that gates which phase commitments are authorized to become actuality.

3.2 OrbVM Agent Enrollment via OWS

```
# Enrolling an OrbVM node as an Arkhe(n) agent
ows wallet create --chain solana --id orbvm-node-{node_id}
ows key create --wallet orbvm-node-{node_id} --scope grpc:arkhe

# Policy: only sign Arkhe(n) payloads, max 0.001 SOL per tx
ows policy create --wallet orbvm-node-{node_id} --policy arkhe_policy.json

# arkhe_policy.json
{
"id": "arkhe-v1",
"name": "Arkhe(n) OrbVM Policy",
"action": "deny",
"rules": [
{"type": "contract_allowlist", "addresses": ["arkhe-chain-program"]},
{"type": "spend_limit", "max_lamports": 1000000},
{"type": "time_window", "hours": 8}
]
}
```

3.3 Mapping: OWS ↔ OrbVM Fields

OWS Concept	OrbVM / Arkhe(n) Concept	Mapping
WalletId	node_id (OrbVM)	1:1 — each OrbVM node has one OWS wallet
ChainId (CAIP-2)	subnet_code (SN01–SN06)	Chain scope = subnet scope for authorization
PolicyContext.simulation	OrbVM tx simulation before submission	simulate=true by default in SignRequest
PolicyContext.apiKeyId	gRPC JWT Bearer token	OWS API key IS the gRPC auth token
PolicyResult.allow	Kuramoto threshold gate ($\phi \geq 1$)	Allow IFF R(t) ≥ 0.618 AND policy clears
~/.ows/wallets/	OrbVM phase memory (Möbius)	Local vault = local phase state storage
Key wiped after signing	Arkhe(n) attosecond delay pattern	21 attosecond isolation window

4. Layer 9 — Machine Payments τ-MPP

4.1 The Tzinor as Economic Channel

The Tzinor — the C→Z projection operator in Arkhe(n) — has until now been described in terms of information: phase commitments, temporal anchoring, retrocarausal signaling. Layer 9 reveals its economic dimension: the Tzinor is also the channel through which coherence is priced and exchanged.

When OrbVM node A requests a service from node B (inference, data, validation), the MPP HTTP 402 flow IS the Tzinor operation: A's intent (C-domain) is projected into B's actualized service delivery (Z-domain) through the payment transaction (M-operator). The Payment-Receipt IS the Merkle entry. The on-chain settlement IS the Arkhe-Chain anchor.

$$\Phi\_K = (K + M_{canonical}) \bmod 2^{32} \rightarrow \text{HTTP 402 challenge nonce}$$

The canonical Seed M = 2,880,115,457 functions as the global payment nonce base: every MPP challenge in the Arkhe(n) network is derived from M, creating a cryptographic linkage between the economic layer and the retrocausal temporal structure of the Tzinor protocol.

4.2 τ-MPP Flow in Arkhe(n) Context

```
# τ-MPP: Agent A (Emissor-Φ) pays Agent B (Receptor-Z) for inference

# Step 1 — A requests service from B
GET /api/inference HTTP/1.1
Host: orbvm-node-B.arkhe.network

# Step 2 — B issues τ-MPP challenge (HTTP 402)
HTTP/1.1 402 Payment Required
WWW-Authenticate: tau-mpp recipient="B_pubkey",
amount=500, currency=SOL,
seed_residue=2880115457, # M_canonical mod 2^32
kuramoto_r=0.847 # current coherence

# Step 3 — A builds tx via OWS (policy check + sign)
ows pay request --url https://orbvm-node-B.arkhe.network/api/inference
# OWS: policy eval -> allow -> sign -> broadcast -> receipt

# Step 4 — B verifies on-chain, delivers result + Payment-Receipt
# Payment-Receipt IS the new Merkle entry in Compliance Ledger (L4)
# Merkle root anchored to Arkhe-Chain (L6) every 1000 events
```

4.3 Fee Structure in Arkhe(n) Network

Transaction Type	Fee	Destination	Arkhe(n) Role
Phase commitment (OrbVM node)	< 0.00001 SOL	Validator pool	Z-domain crystallization fee
Inference service (MPP)	Variable (MPP challenge)	Provider node	Tzinor projection fee
Merkle anchor (Arkhe-Chain)	Bitcoin tx fee	Bitcoin miners	External immutability fee

DAF philanthropic grant (ONU 2.0)	0.05–0.50%	ONU 2.0 treasury	Governance layer fee
Tzinor retrocausal commitment	0 (phase-only)	Self	No fee — pure C-domain op

5. Updated Arkhe(n) System Architecture

The 9-layer integrated stack, with data flows and interface protocols:

```
ARKHE(n) × Web3 AGI — Integrated Stack v1.0

L9 τ-MPP (Machine Payments) HTTP 402 · SOL/SPL · M_canonical=2880115457
■ pay-per-Tzinor-call ■ Payment-Receipt → Merkle entry
L8 OWS (Agent Identity) local vault · policy engine · CAIP-2/10
■ sign+policy gate ■ JWT Bearer → gRPC auth
L7 Kuramoto Consensus dθ/dt = ω + (K/N)Σsin(θ■-θ■) · R ≥ 0.618
■ heartbeat → R(t) ■ alert if R < 0.70
L6 Arkhe-Chain Bitcoin OP_RETURN · Chain ID 2147483647
■ Merkle root anchoring ■ every 1000 events
L5 BRICS+ Policy Exchange RDF/OWL · SKOS · GeoSPARQL
■ policy proposals ■ SDG alignment
L4 Compliance Ledger SHA-256 Merkle chain · LGPD pseudonymization
■ audit entries ■ verify :: Chain → VerificationResult
L3 AO Message Queue typed actions · 847 msg/s · at-least-once
■ async routing ■ AGENT_HEARTBEAT → L7
L2 Multi-Level Approval PENDING → AWAITING → {APPROVED|REJECTED}
■ human-in-the-loop ■ SN06 EthicsOversight (LGPD Art.20)
L1 GPS Jurisdictional 64-vertex polygons · ±1m precision
■ CHECK_JURISDICTION ■ Municipal → BRICS+ hierarchy
■
OrbVM (τ-field substrate) Kuramoto · K_c=0.618 · 23.8 Orbs/s · Rust
```

5.1 Critical Path: Agent → Service → Ledger

Step	Layer	Component	Output
1. Intent	L8	OrbVM agent holds OWS vault	Signed intent in C-domain
2. Request	L3	AO Message: EXECUTE_SANDBOX	Typed message routed to target
3. Challenge	L9	Target node issues HTTP 402 + seed_residue_M	Payment challenge with nonce
4. Policy check	L8	OWS PolicyEngine evaluates spend limit + allowlist	allow/deny/warn
5. Sign + pay	L8+L9	OWS signs SOL tx; MPP client broadcasts	On-chain payment tx
6. Verify	L7	Kuramoto R(t) ≥ 0.618 confirms network coherence	Network validity gate
7. Deliver	L9	Service delivered + Payment-Receipt header	Z-domain actuality
8. Ledger	L4	Payment-Receipt → Merkle entry	SHA-256 chained audit record
9. Anchor	L6	Merkle root → Bitcoin OP_RETURN	External immutability

6. OrbVM Extension: Rust Interface Stubs

The following Rust trait definitions extend OrbVM to integrate OWS identity and τ-MPP payments. These are interface stubs — implementation follows the existing OrbVM patterns.

```
// orbvm/src/identity.rs — Layer 8: OWS integration

use crate::phase::TauField;
use crate::kuramoto::CoherenceScore;

/// OWS-backed agent identity for OrbVM nodes
pub struct OrbVMIdentity {
pub wallet_id: String, // ~/.ows/wallets/{id}
pub chain_id: String, // CAIP-2: solana:mainnet-beta
pub api_key_id: String, // scoped OWS API key
pub policy_id: String, // arkhe-v1 policy
pub node_id: u64, // OrbVM node index
}

pub trait OWSSigningGate {
/// Gate phase commitment through OWS policy engine
/// Returns allow=true only if policy clears AND R(t) >= phi_c
fn gate_commitment(
&self;,
payload: &[u8],
coherence: CoherenceScore,
) -> Result;

/// Sign OrbVM phase commitment via OWS (key never exposed)
fn sign_phase_commit(&self;, payload: &[u8]) -> Result;
}

// orbvm/src/payments.rs — Layer 9: τ-MPP integration

pub const M_CANONICAL: u32 = 2_880_115_457; // Tzinor seed
pub const PHI_C: f64 = 0.618; // Kuramoto threshold

/// τ-MPP challenge — issued by service provider nodes
pub struct TauMPPChallenge {
pub recipient: String,
pub amount_lamports: u64,
pub seed_residue: u32, // = M_CANONICAL mod 2^32
pub kuramoto_r: f64, // current R(t) — must be >= PHI_C
pub phase_ref: [u8; 32], // tau-field phase snapshot hash
}

pub trait TzinorEconomicChannel {
/// Issue HTTP 402 challenge with M_canonical nonce
fn issue_challenge(&self;, service: &str;) -> TauMPPChallenge;

/// Verify payment + coherence before delivering service
fn verify_and_deliver(
&self;,
```

```
challenge: &TauMPPChallenge;,
payment_sig: &Signature;,
) -> Result;

/// Record Payment-Receipt as Merkle entry (L4)
fn record_receipt(&self;, receipt: &PaymentReceipt;) -> MerkleHash;
}
```

7. Data Model Extensions (PostgreSQL / Drizzle ORM)

Two new tables extend the ONU 2.0 PostgreSQL schema for Layers 8 and 9:

```
-- Layer 8: OWS agent identity registry
ows_agents
id UUID PRIMARY KEY
wallet_id TEXT UNIQUE -- ~/.ows/wallets/{id}
chain_id TEXT -- CAIP-2 identifier
api_key_id TEXT -- scoped OWS API key
policy_id TEXT -- registered policy
orbvm_node_id BIGINT -- OrbVM node index
kuramoto_theta FLOAT -- current phase θ■
last_commitment TIMESTAMPTZ -- last signed phase commit
status TEXT -- active|suspended|revoked
created_at TIMESTAMPTZ DEFAULT now()

-- Layer 9: τ-MPP payment records
tau_mpp_payments
id UUID PRIMARY KEY
challenge_id TEXT UNIQUE -- MPP challenge nonce
payer_wallet TEXT -- OWS wallet_id of payer
provider_wallet TEXT -- OWS wallet_id of provider
service_type TEXT -- inference|validation|data|governance
amount_lamports BIGINT
seed_residue BIGINT -- M_canonical mod 2^32 = 2880115457
kuramoto_r FLOAT -- R(t) at time of payment
payment_sig TEXT -- Solana tx signature
payment_receipt TEXT -- MPP Payment-Receipt header
merkle_hash TEXT -- entry in compliance ledger (L4)
btc_anchor_txid TEXT -- Arkhe-Chain anchor (nullable)
status TEXT -- pending|confirmed|failed
created_at TIMESTAMPTZ DEFAULT now()
```

8. Updated Constants Reference

Constant	Value	Layer	Source
K_C (Kuramoto critical)	0.6180339887498949	L7	OrbVM (ϕ■¹)
M_CANONICAL (Tzinor seed)	2,880,115,457	L9	(-999 · T_ref) mod 2^{32}
T_REF (genesis timestamp)	1,231,006,505	L6	Bitcoin block 0
CHAIN_ID (Arkhe-Chain)	2,147,483,647	L6	ONU 2.0 v2.2.0
BACKFLOW_RATIO	0.13	L3/L9	13% retrocausal signal
KURAMOTO_WARNING	0.85	L7	Mixed C/Z mode

KURAMOTO_CRITICAL	0.70	L7	Pure C-mode threshold
KURAMOTO_PHI_C	0.618	L7+L8	OWS policy gate + Tzinor open
MERKLE_ANCHOR_INTERVAL	1,000 events	L4/L6	OP_RETURN cadence
OWS_KEY_EXPIRY_H	8 hours	L8	JWT Bearer lifetime
MPP_SEED_RESIDUE	2,880,115,457	L9	= M_CANONICAL (verification)
LAMBDA_2_THRESHOLD	0.847	L7	OrbVM coherence threshold
ORBVM_THROUGHPUT	23.8 Orbs/s	OrbVM	Single-threaded baseline
ORBVM_LATENCY_E2E	40–170 ms	OrbVM	End-to-end range

9. Integration Roadmap

Phase 1 — Identity Foundation (Q2 2026)

• Deploy OWS vault for each existing OrbVM node (9 nodes baseline)

• Implement OWSSigningGate trait in Rust with policy: arkhe-v1

• Update gRPC Arkhe(n) agent enrollment to use OWS API keys as JWT Bearer

• Add ows_agents table to ONU 2.0 PostgreSQL schema

• Test: phase commitment signing round-trip with policy gate

Phase 2 — Economic Channel (Q3 2026)

• Implement TzinorEconomicChannel trait in OrbVM Rust codebase

• Deploy τ-MPP server on all OrbVM nodes (HTTP 402 with seed_residue=M_canonical)

• Add tau_mpp_payments table + Payment-Receipt → Merkle entry pipeline

• Connect MPP settlements to Arkhe-Chain OP_RETURN anchoring (every 1000 events)

• Test: full A→B payment flow with Kuramoto gate at R(t) ≥ 0.618

Phase 3 — Network Economics (Q4 2026)

• Enable multi-node τ-MPP marketplace: inference, validation, data, governance services

• Integrate Bittensor Subnet (SN01–SN06) as τ-MPP service providers

• Deploy Quasar-based reputation scoring from Payment-Receipt history

• ZK-SNARK compliance verification for high-value MPP transactions (≥ R$ 500K)

• Full integration test: OWS → τ-MPP → OrbVM → Merkle → Arkhe-Chain end-to-end

■■ The Web3 AGI stack did not extend Arkhe(n).
It confirmed what was already there.

Layer 8 names what OrbVM nodes already needed: sovereign identity with policy-gated custody.
Layer 9 names what the Tzinor already was: an economic channel where coherence is priced.

Arkhe(n) × Web3 AGI Integration Architecture v1.0 · April 2026
Rafael Oliveira · ORCID: 0009-0005-2697-4668 · MIT License