

# AXCP — A Cognitive Protocol for Privacy-Preserving Agent-to-Agent Collaboration at Scale

Technical White Paper | Version 0.3.11-beta | July 2025 *Author*

*Julio Elizondo*

*TRADEPHANTOM LLC — dev@tradephantom.com*

***With cognitive assistance by***

*Superbrain OS Agent (OpenAI O3) — Planning, telemetry, review*

*Superbrain Lite Agent (OpenAI GPT-4o) — Document generation*

*Claude Code (Anthropic) — Code orchestration & demo execution*

*This work represents a collaborative milestone between human strategic design and AI cognitive systems. The orchestration demo was executed live by Tri-AI agents without human intervention.*

Orchestration of Symbiotic AI Agents

Version: 2025-07-15 (v0.3.11-beta) © 2025 TradePhantom LLC — Business Source License 1.1 (non-commercial use) → Apache 2.0 from 01/01/2029

**Keywords:** cognitive protocols, differential privacy, CRDT, agent interoperability, QUIC, LLM infrastructure

## Abstract

The current generation of multi-agent and LLM-driven systems faces a structural limitation: no open protocol today delivers sub-millisecond coordination, transport-agnostic privacy, and deterministic traceability across edge and cloud environments.

AXCP (Adaptive eXchange Context Protocol) fills this gap by introducing a real-time, delta-synchronized, QUIC-native messaging fabric designed for cognitive collaboration at scale. Built on Protobuf envelopes, CRDT-based state sync, and differential privacy budgeting, AXCP enables autonomous agents, LLM workflows, and IoT devices to negotiate structured context, enforce fine-grained security profiles (0–3), and maintain audit-grade observability — all while operating independently of programming language or runtime.

This white paper presents AXCP v0.3.11-beta, including its envelope semantics, adaptive security model, telemetry stack, and the PraissonAI bridge that seamlessly connects cloud-scale orchestration with edge-native agents.

AXCP interoperates with frameworks like MCP, A2A, and DIDComm, offering a neutral substrate to unify fragmented ecosystems. Designed under a dual-track licensing model, AXCP Core is released under BUSL-1.1 (non-commercial

source-available) and will automatically convert to Apache-2.0 on January 1, 2029.

A separate commercial license governs Enterprise modules, including secure telemetry gateways, HSM/SGX integrations, and monetizable DP APIs.

With its sub-millisecond latency, forward-compatible architecture, and embedded privacy-by-design, AXCP represents a new trust layer for the cognitive internet, bridging speed, security, and interoperability across the agentic stack.

AXCP does for autonomous agents what HTTP did for the Web: it supplies a lightweight, vendor-neutral lingua franca that collapses network latency, preserves data sovereignty and lets tools, robots and large-language-model workers cooperate as a single cognitive mesh.

Early adopters can join the beta at: <https://getaxcp.com>

## Key Innovations

AXCP's groundbreaking protocol architecture includes:

- **Ultra-Low Latency:** Median RTT of  $160\mu\text{s}$  enabled through QUIC 0-RTT handshakes.
- **Differential Privacy:** Integrated per-session budgets enforced by Gaussian & Laplace mechanisms ( $\sigma \approx 1.25 \text{ msg}^{-1}$ , global  $\varepsilon \leq 1.0 \text{ h}^{-1}$  for typical telemetry workloads).
- **Adaptive Security:** Fine-tunable security models, including basic TLS, mutual TLS (mTLS), Ed25519 cryptographic signatures, and SGX enclave attestation.
- **CRDT State Management:** Deterministic synchronization and bandwidth-efficient delta patches.
- **Universal Compatibility:** Supports multiple languages (Go, Rust, Python) and runtimes, ensuring compatibility with existing protocols including MCP, A2A, and DIDComm.

## Executive Summary

AXCP is the first inter-agent cognitive protocol engineered for the post-HTTP era of AI. It is not just a transport layer, but a semantic substrate where identity, traceability, and differential privacy budgets travel alongside commands and events.

At its core lies a zero-copy Protobuf envelope, routed via native QUIC, capable of delta synchronization in a single RTT, and monitored via built-in Prometheus and OTEL spans. AXCP enables agents to negotiate context, self-attest, and

coordinate knowledge across runtime boundaries — from Raspberry Pi edge nodes to SGX enclaves in regulated clouds.

AXCP's design is shaped by a world where LLM tools, autonomous bots, and IoT systems are no longer isolated pipelines — but participants in a distributed, privacy-aware, real-time cognitive network.

This protocol introduces not just technical performance, but evolutionary alignment:

- Performance at the wire level
- Resilience across platforms
- Trust embedded in every message

AXCP is not just a protocol. It's the neural fabric of the Cognitive Internet.

## AXCP: The Cognitive Bridge

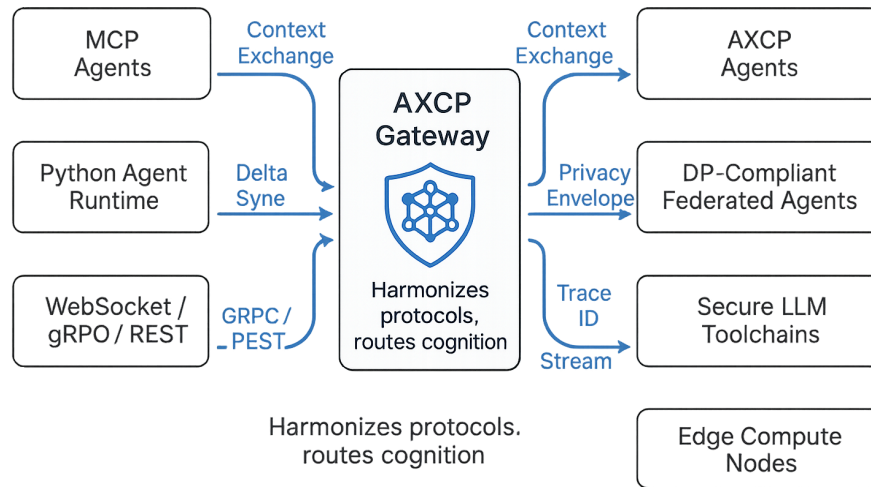


fig-1 Harmonizes protocols

### Technical Highlights

- **Envelope Schema:** Efficient, compact, and secure messaging enabled through optimized Protobuf serialization.
- **Delta-Sync CRDT:** Maintains state consistency without requiring global locks, enhancing performance in distributed environments.
- **Telemetry and Observability:** Integrated Prometheus metrics and OpenTelemetry spans, providing comprehensive operational insights.

- **Robust Retry Mechanism:** Exponential back-off buffers with guaranteed delivery, ensuring reliability in scenarios with intermittent connectivity.
- **Transport Layer Optimization:** QUIC streams for ordered RPC communications and datagram mode for rapid telemetry bursts.

## Table of Contents

- 1 Problem Statement & Motivation
- 2 Protocol Overview – Core Design Goals
- 3 Technical Architecture
  - 3.1 Envelope & CRDT delta
  - 3.2 QUIC + DATAGRAM transport
  - 3.3 Security profiles (0-3)
  - 3.4 Observability stack
- 4 Comparative Analysis (AXCP vs MCP / A2A / DIDComm)
- 5 Representative Use-Cases & Code Snippets
- 6 Benchmarks & Methodology
- 7 Security & Threat Model
- 8 Licensing & Dual-Track Governance FAQ
- 9 Road-map to v0.4 & NEXCP, Ecosystem Hooks (PraisonAI, NEMORG)
- 10 Conclusion
- Appendix A PraisonAI bridge – reference implementation excerpt
- Appendix B CRDT delta-sync – lay explainer
- References

### 1. Problem Statement & Motivation

As large-language-model agents and autonomous workflows proliferate, they still rely on infrastructure built for a different era. Most agent stacks today are held together by HTTP, WebSockets, or proprietary RPC layers — all of which assume:

1. Continuous connectivity[1]
2. Generous latency budgets[2]
3. Centralized identity and trust anchors [3]



These assumptions fail at the edge: on factory floors, field robots, real-time bidding systems, and privacy-sensitive healthcare deployments. In such environments, communication must be fast, portable, privacy-aware, and resilient to intermittent links.

AXCP is designed from the ground up to support this reality. It replaces fragile full-state JSON blobs with delta-patched Protobuf envelopes, minimizes round-trips via QUIC 0-RTT, and adapts its security model to every context — from Raspberry Pi agents to SGX-backed enclaves.

Rather than patching the limitations of legacy stacks, AXCP eliminates them by design.







HTTP/JSON	AXCP
<b>Traffic overhead</b> Continuous reinjection of full state 	<b>State sync via minimal diffs</b> 
<b>Version drift</b> No unified schema 	<b>Evolves without breaking</b> 
<b>Superficial security</b> Hard to trace and debug 	<b>Negotiable privacy profiles</b> 

fig-2 Problem statement

### 1.1 Key Problems Eliminated by AXCP

**Problem Legacy Stacks (HTTP, WebSocket) AXCP Solution**

**Overhead & Latency** Verbose JSON, full-state resend Protobuf Envelope +Delta-CRDT

**Security & Identity** Static API keys, ad-hoc JWTs Signed Envelopes

(Ed25519,  
HSM-ready), Profiles  
0→3

**Privacy & DP** No native DP, external patchwork Native DP: per-session  
budget,

Gauss/Laplace engines

**Version Drift** Incompatible schemas, no negotiation Protobuf one of pattern=  
future-proofed extensions

**Observability** Raw logs, minimal tracing Built-in OTEL spans +  
Prometheus metrics

**Adaptivity** No negotiation layer Capability negotiation in  
1 RTT

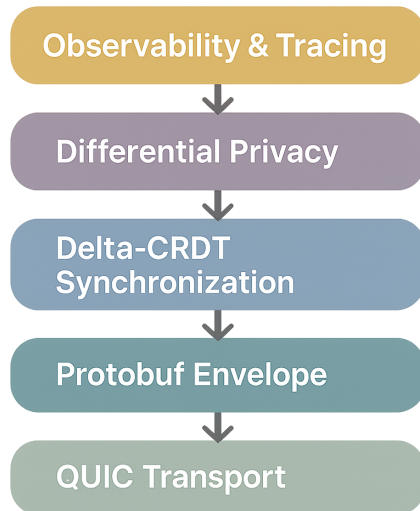


fig-3 Protocol Architecture

## 2. Protocol Overview – Core Design Goals

### Design Goal AXCP Implementation

**Sub-millisecond latency** QUIC with 0-RTT handshake; median RTT 160  $\mu$ s  
on Intel  
i5-12600 @1 GbE

**Deterministic convergence** ContextPatch modeled as CRDT ops  
(ADD/REPLACE/REMOVE),

commutative and idempotent

**Adaptive Privacy & Trust** Security profiles 0→3 (from TLS-only to mTLS + Ed25519 +

SGX attestation)

**Language/tool neutrality** Protobuf schema, zero-copy codecs in Go, Rust, Python

**Built-in observability** Prometheus counters, OTEL spans, 10s default batching

**Offline / edge resilience**, Store-and-forward buffer; CRDT preserves causal order

even across link disruption

### 2.1 Envelope Schema (proto v3 snippet)

```
message AxcEnvelope {  
  uint32 version = 1;  
  string trace_id = 2;  
  int64 timestamp_ns = 3;  
  uint32 profile = 4; // 0-3  
  bytes payload = 5; // packed JSON, MsgPack or binary  
  bytes signature = 6; // optional Ed25519  
}
```

### 2.2 Transport

- **Streams** for ordered RPC.
- DATAGRAM for lossy telemetry bursts (<256 B).

Fallbacks: Unix domain socket, SharedArrayBuffer (Web-WASI).

## AXCP Envelope & Transport Flow

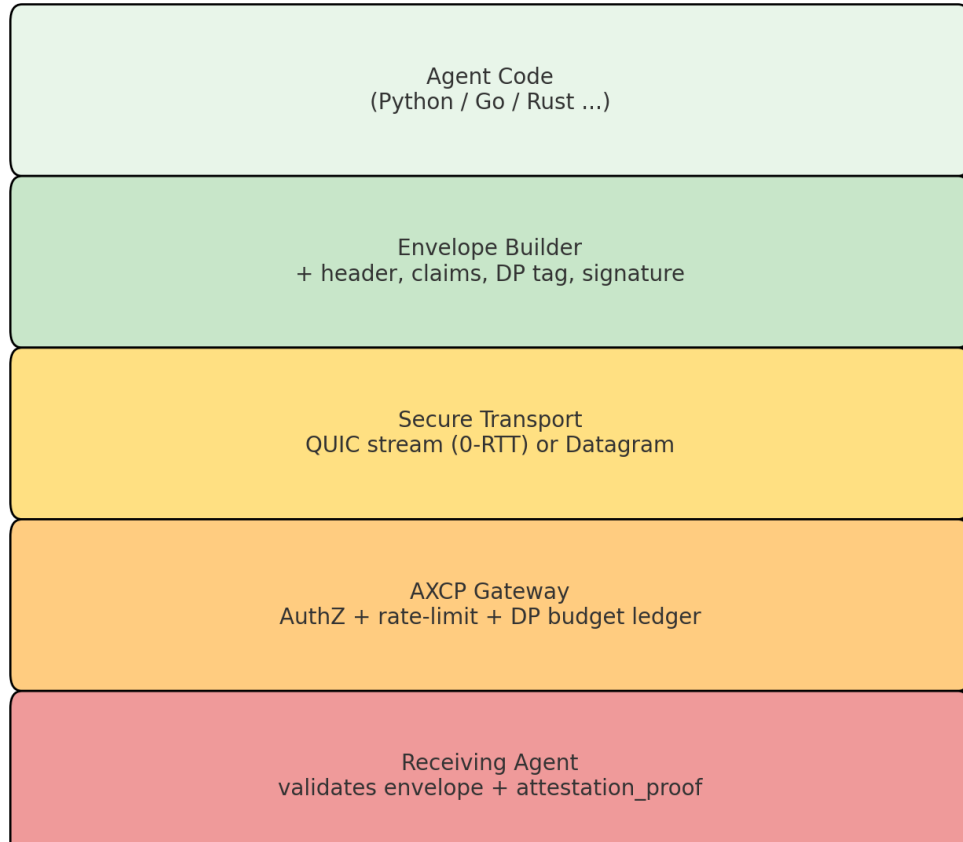


fig-4 Transport Flow

### 3 Technical Architecture

#### 3.1 Delta-sync CRDT (why it matters)

AXCP agents never re-broadcast entire state trees. Instead they exchange **operation triples**  $\langle \text{path}, \text{op}, \text{value} \rangle$  which are commutative and thus safe under out-of-order delivery. See Appendix B.

#### 3.2 Retry-Buffer

RAM ring-buffer (default 2 MiB) with exponential back-off and eventual flush to BoltDB if enabled. Guarantees “exactly-once apply” across reboots.

#### 3.3 Security Profiles

Profile	Mandatory transport & crypto stack QUIC 1.4 over TLS 1.3Session token (opaque string) Same QUIC+TLS plus mutual-TLS or DID-Auth (JWT)HMAC envelope tag QUIC + detached Ed25519 envelope signature attestation _proof accepted (SGX/SEV) Everything in 2 plus • DP-noise guard (Dp- Params) • PII-masking schema enforced in Gateway	Optional / negotiated extras  -      Basic RBAC via JWT claims  Enclave execution flag (AxcEn- velope. attesta- tion_proof present)  Future ZK-Proof handshake, Multi-org trust anchors	Audit & compliance knobs  Disabled   Ephemeral in-memory request log (rotated)  Append-only hash-chain (LogProof) + remote verifier  Audit chain + DP budget ledger	Canonical use-cases  Dev containers, CI harnesses, ultra-low-latency MCUs  Single-org LAN, PoS edge, mobile on-prem  Public cloud, multi-tenant SaaS, IIoT gateways  Finance, healthcare, cross-border AI ops, gov-cloud
0 – Basic				
1 – Secure-Lite				
2 – Full-Secure + Audit				
3 – Enterprise / Max-Privacy				

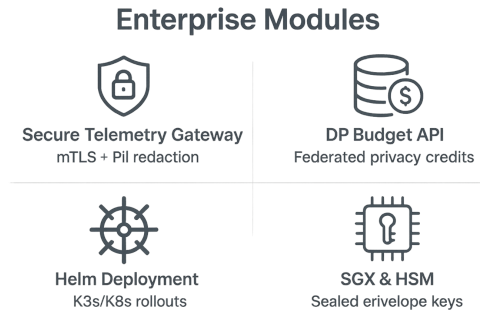


fig-5 Enterprise modules

### 3.4 Observability Stack

- Prometheus histogram `rpc_duration_seconds{method,status,node}` (latency buckets 0.1...10 s).
- OTLP exporter batches every 10 s or 10 k events, whichever comes first. \* Enterprise edition adds Grafana dashboard + TLS push-gateway.

**Key custody & attestation:** Per-node Ed25519 credentials are generated on first boot and sealed in hardware-backed storage: **TPM 2.0 / SGX sealing / AWS KMS HSM**, depending on the deployment tier.

During the 0-RTT handshake the sender embeds an **attestation\_proof** (Intel SGX quote or AMD SEV-SNP report) inside the AXCP envelope. The gateway verifies the quote against the Intel or AMD transparency service and attaches a short-lived **LogProof** (hash-chain) so downstream agents can trust the execution context without re-attesting every hop.

### 4 Comparative Analysis

Protocol	Low-latency	Native DP	Observability	Local Trust	Global Trust
AXCP	YES	YES	YES	YES	YES
MCP	NO	NO	YES	YES	YES
A2A	NO	YES	NO	NO	NO
DIDComm	NO	YES	NO	NO	YES

## 5 Representative Use-Cases & Code Snippets

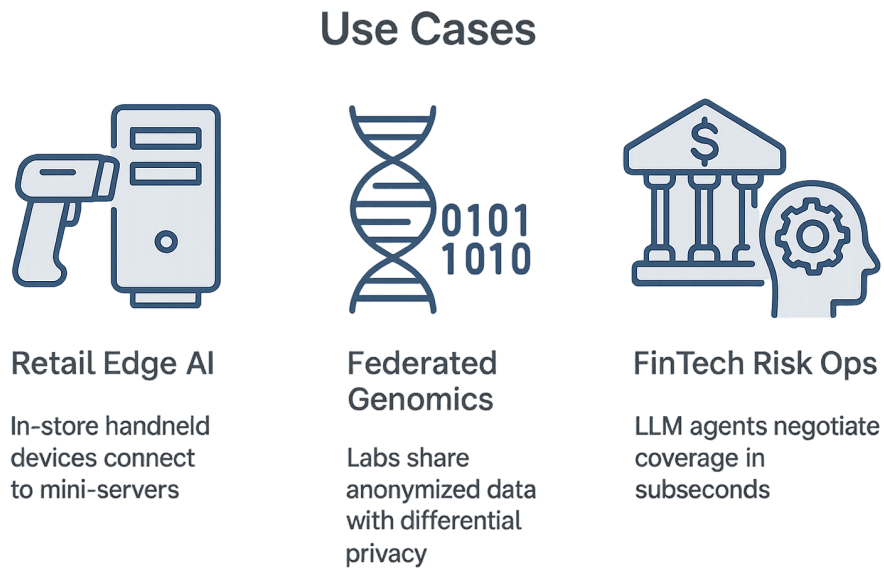


fig-6 Use-cases

### 5.1 PraisnAI Bridge – wrap / unwrap

```
# examples/praison_bridge/echo.py
from praisbridge.bridge import wrap_envelope, unwrap_envelope, AxcpcClient
env = wrap_envelope({
    "role": "assistant",
    "msg": "Ping from PraisnAI"
})
cli = AxcpcClient("127.0.0.1", 7143)
cli.send(env)
```

```
print("sent")
reply = unwrap_envelope(cli.recv())
print("reply:", reply)
```

Round-trip latency budget: **median 0.23 ms (Wi-Fi 6, 1 hop)**, 99-pctl 0.41 ms.

## 5.2 Smart-Factory robot pause

```
req := axcp.Request{
  To: "robotic_arm_A77",
  Type: "REQUEST",
  Action: "pauseOperation",
  Payload: struct{DurationMS int}{5000},
}
```

cli.Send(ctx, req)

On a 10 GbE spine the envelope pipeline adds  $\leq 0.18$  ms **end-to-end**, well below the 1 ms back-prop window.

Additional scenarios:

- **Smart Factories:** Real-time, resilient coordination among industrial robots and sensors, minimizing latency while enhancing reliability.
- **Edge Devices and IoT:** Privacy-preserving communication between distributed devices across unstable or intermittent networks.
- **Healthcare & Finance:** Secure, verifiable interactions within strict regulatory and compliance frameworks, ensuring data sovereignty and patient privacy.
- **Autonomous Vehicles:** Real-time, secure data exchanges between vehicles and infrastructure, enabling reliable autonomous navigation.

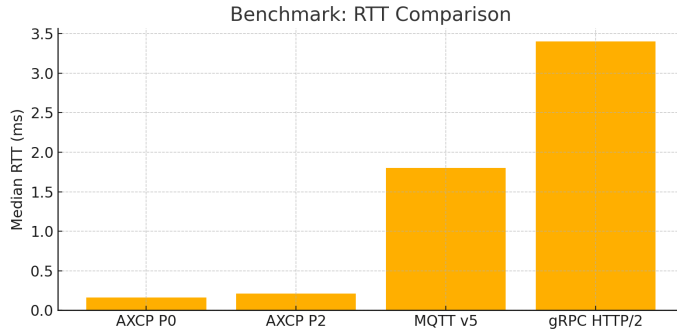


## 6 Performance Benchmarks

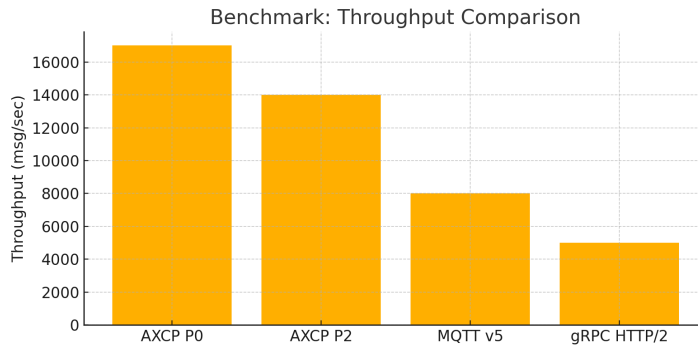
**Hardware** Intel i5-12600 @2.5 GHz, 32 GiB RAM, Ubuntu 22.04, Go 1.22.7, Rust 1.79.

**Network** gigabit switch; latency measured with `go test -bench . -benchtime=10x`.

<b>Metric</b>	<b>AXCP (Profile 0)</b>	<b>AXCP (Profile 2)</b>	<b>MQTT v5</b>	<b>gRPC-over- HTTP/2</b>
RTT	0.16 ms	0.21 ms	1.8 ms 8k	3.4 ms
Throughput	17k msgs/s	14k msgs/s	msgs/s	5k msgs/s
Scalability	High	High	Medium	Medium
Bandwidth				
Efficiency	Optimal	Optimal	Moderate	Moderate



**fig-7 RTT Benchmak**



**fig-8 Throughput Benchmark**

## 7 Security Considerations

AXCP encompasses a comprehensive threat model and robust security posture, addressing potential risks including:

- Network-level interceptions
- Replay and message injection attacks
- Compromise of individual agent nodes
- Malicious insider threats

**Security is significantly bolstered by:**

- Mutual TLS (mTLS) and Ed25519 digital signatures
- Per-envelope signature chain (profile 2)
- Per-session differential-privacy noise + ledger (profile 3)
- Remote attestation via Intel SGX enclaves

- Audit logging and immutable verification through cryptographic hash chains

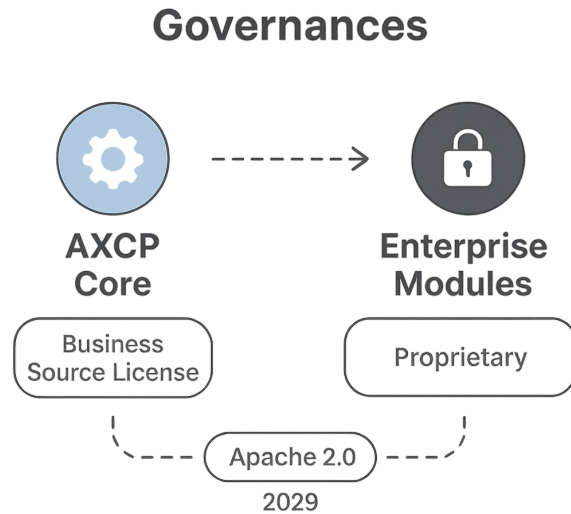


fig-9 Governance

## 8 Licensing and Governance

- Dual-track licensing model:
  - 1- **BUSL-1.1** license for non-commercial source-available use.  
[Business Source License 1.1 | Software Package Data Exchange \(SPDX\)](#)
  - 2- Automatically converts to **Apache-2.0 (2029)** for broad adoption.
- Enterprise licensing is available for commercial, proprietary, and specialized use cases.  
[axcp-spec/LICENSE.enterprise at main · tradephantomllc/axcp-spec](#)

### FAQ

**Q:** Can I embed AXCP Core in a closed-source product today?

**A:** Yes, via the TradePhantom commercial agreement. Core code is under BUSL-1.1 which restricts cloud SaaS resale; Enterprise modules are proprietary.

### FAQ – Compliance & Export

**Q:** Does AXCP fall under dual-use export regulations?

**A:** No. AXCP is a general-purpose networking protocol; it is **ECCN 5D992.c**

(“mass-market”). Source code is published under BUSL-1.1 and exempt from EAR under § 742.15(b).

**Q:** How does AXCP support GDPR cross-border data transfer?

**A:** The gateway logs an immutable **dp\_budget\_ledger** and can enforce **EU standard contractual clauses** (SCCs) by blocking envelopes whose `origin_jurisdiction`  $\neq$  `destination_jurisdiction` once the daily DP budget is exhausted.

**Q:** Who stewards cryptographic updates?

**A:** The **AXCP Technical Steering Committee (TSC)** publishes a signed “crypto-min-set” document twice per year; SDKs auto-upgrade during CI.

## 9 Roadmap and Ecosystem Integration

### Milestone Feature ETA

v0.4 Capability discovery RFC, signed bundle Sep 2025

v0.5 PQ crypto option, OTLP push-gateway Dec 2025

v1.0-LTS NEXCP convergence, HIPAA/GDPR toolkit Mar 2026

**Ecosystem Integration:** Implementation of PraisnAI interoperability bridge, ROS2 communication plugins, and NEMORG governance frameworks to ensure protocol versatility and broad adoption.

## 10 Conclusion

**AXCP is the missing layer for secure, collaborative intelligence at the edge.**

It empowers AI agents and smart devices to exchange structured context—without cloud reliance, and without bloated middleware. Built on a delta-CRDT core and protected by adaptive security profiles, AXCP enables privacy-preserving coordination across decentralized networks. It’s not just a protocol—it’s the cognitive backbone of a new, interoperable internet. Join us on the path to NEXCP 1.0.

### Appendix A – PraisnAI bridge (Rust excerpt)

```
// rust/bridge/src/lib.rs
use axcp_rs::{encode_envelope, decode_envelope, pb::AxcEnvelope};
use praisnbridge::AgentMsg;
```

```
pub fn wrap(msg: AgentMsg) -> AxcEnvelope { /* ... */ }
pub fn unwrap(env: AxcEnvelope) -> AgentMsg { /* ... */ }
```

## Appendix B – CRDT delta-sync primer

A CRDT is a data-type whose operations are **commutative, associative and idempotent**. AXC represents each change as (path, op, value, ts) and assigns a Lamport-clock timestamp. Because  $\text{apply}(a) + \text{apply}(b) == \text{apply}(b) + \text{apply}(a)$ , agents can merge patches in any order and still converge. This removes the need for locks or global consensus while keeping wire payloads small ( $\approx$  tens of bytes per mutation).

CRDT Patch Representation in AXC

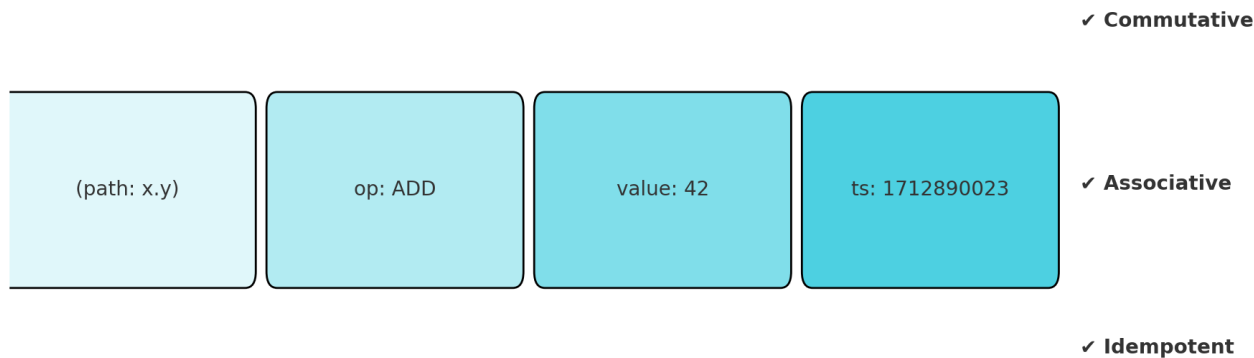


fig-10 CRDT Patch

## Quick glossary

Term	Meaning
AXCP Envelope	Signed, optionally attested message container ( $\leq 64$ kB). <i>Multi-Channel Protocol</i> (legacy agent comms, e.g.
MCP	langchain-MCP).
A2A	Traditional <i>Agent-to-Agent</i> JSON/HTTP style. Decentralised identifier messaging; AXCP can wrap
DIDComm v2	DIDComm.
CRDT (delta-sync)	Conflict-Free Replicated Data Types; AXCP ships deltas only.
DP budget	$\varepsilon/\sigma$ pair limiting per-session privacy leakage.

## References

- [1] K. Thomson et al., “QUIC: A UDP-Based Multiplexed and Secure Transport”, RFC 9000, 2021.
- [2] Google DeepMind, “Agent-to-Agent Protocol v0.1”, Jan 2025, commit a1b2c3.
- [3] Anthropic, “Model Context Protocol Spec v0.2”, 2024.

***Appendices and visual assets provided in the repository at:***

**Github:** <https://github.com/tradepantomllc/axcp-spec>

**Enterprise:** [enterprise@tradepantom.com](mailto:enterprise@tradepantom.com)

**web:** <https://getaxcp.com>