Solana Wallet Analytics API

Implementation Specification with x402 Payment Protocol

Version: 1.0.0

Date: October 30, 2025

Status: Draft Specification

Executive Summary

This document specifies the implementation of a Solana Wallet Analytics API that leverages the x402 payment protocol to enable frictionless, pay-per-use access without accounts or API keys. The service provides enriched wallet intelligence including portfolio analysis, token holdings, NFT valuations, DeFi positions, and risk assessment.

Key Objectives:

- Enable AI agents and developers to access Solana wallet analytics via instant cryptocurrency payments
- Eliminate subscription friction with pay-per-query micropayments starting at 0.05 USDC
- Deliver comprehensive wallet intelligence aggregated from multiple data sources
- Maintain enterprise-grade security, privacy, and compliance standards

Table of Contents

1. System Architecture

1.1 High-Level Architecture

The system follows a layered architecture pattern optimized for security, performance, and maintainability:

Layer 1: API Gateway & x402 Payment Handler

- Receives incoming HTTP requests from AI agents and clients
- Returns HTTP 402 Payment Required with x402 payment challenge
- Validates payment proofs via PayAI or custom facilitator
- Implements rate limiting, request validation, and DDoS protection

Layer 2: Analytics Engine

- Orchestrates data collection from multiple sources
- Implements business logic for portfolio calculations and risk scoring
- Handles caching strategies to minimize external API calls
- Formats and aggregates data into structured response models

Layer 3: Data Source Integrations

- Solana RPC nodes (Helius, QuickNode, or self-hosted)
- Price feeds (Jupiter, Birdeye, CoinGecko)
- NFT marketplaces (Magic Eden, Tensor API)
- DeFi protocol adapters (Marinade, Jupiter, Orca)

Layer 4: Infrastructure Services

- Redis for distributed caching and rate limiting
- PostgreSQL for analytics, audit logs, and payment records
- Monitoring and observability (Prometheus, Grafana)
- CDN for static content and geographic distribution

1.2 Technology Stack

Runtime & Framework:

- Node.js 20 LTS with TypeScript 5.3+
- Fastify framework for high-performance HTTP server
- Zod for runtime type validation and schema enforcement

Blockchain Integration:

- @solana/web3.js 1.87+ for RPC interactions
- @solana/spl-token for token account parsing
- Metaplex SDK for NFT metadata handling

Payment Protocol:

- x402 protocol implementation (reference TypeScript client)
- PayAI Facilitator SDK for Solana USDC settlement
- ethers.js for multi-chain payment verification (future)

Data & Caching:

- Redis 7+ for distributed caching with TTL strategies
- PostgreSQL 15+ with TimescaleDB for time-series analytics

- ioredis client with cluster support

Security & Infrastructure:

- Helmet.js for HTTP security headers
- rate-limiter-flexible for distributed rate limiting
- Pino for structured JSON logging
- Docker with multi-stage builds for containerization

2. Security & Privacy Framework

2.1 Security Architecture Principles

The security model follows defense-in-depth principles with multiple layers of protection:

Zero Trust Architecture:

- No implicit trust for any request regardless of origin
- Every request must present valid payment proof before processing
- Payment proofs validated cryptographically against blockchain state
- No session state or cookies maintained server-side

Input Validation & Sanitization:

- Wallet addresses validated as base58-encoded Solana public keys
- All inputs validated against Zod schemas before processing
- Query parameters sanitized to prevent injection attacks
- Maximum request sizes enforced (10KB for all endpoints)

Rate Limiting & DDoS Protection:

- Per-IP rate limiting: 10 requests per minute (before payment)
- Per-wallet rate limiting: 100 requests per hour (after payment)
- Cloudflare WAF for L7 DDoS protection and bot mitigation
- Exponential backoff for repeated failed payment attempts

API Security Headers:

- Strict-Transport-Security: max-age=31536000; includeSubDomains
- X-Content-Type-Options: nosniff
- X-Frame-Options: DENY
- Content-Security-Policy: default-src 'none'

Cryptographic Payment Verification:

- Payment proofs validated against on-chain transaction signatures
- Nonces enforced to prevent replay attacks
- Payment amounts verified to match endpoint pricing
- Facilitator signatures validated using Ed25519

2.2 Privacy & Data Protection

Data Minimization:

- No user accounts, email addresses, or personal information collected
- Wallet addresses processed transiently without persistent storage
- Payment proofs retained only for audit compliance (90 days)
- No tracking cookies or analytics scripts deployed

Anonymous Access Model:

- Clients identified only by payment wallet address
- No correlation between different payment wallets
- IP addresses not logged in application layer
- CDN logs anonymized and rotated every 24 hours

Data Retention Policy:

- Query requests: Not logged beyond error debugging (7 days)
- Payment proofs: Retained for financial audit (90 days)
- Cached wallet data: Auto-expire after 5 minutes
- Analytics aggregates: Stored indefinitely (no PII)

Compliance Framework:

- GDPR compliant through data minimization and right to erasure
- No CCPA obligations due to absence of personal data collection
- Payment processing complies with AML/KYC via facilitator
- Terms of Service clearly disclose data practices

2.3 Infrastructure Security

Network Security:

- TLS 1.3 enforced for all client connections
- Certificate pinning for upstream API connections
- Private VPC for internal service communication
- Firewall rules restricting traffic to necessary ports only

Secrets Management:

- API keys stored in HashiCorp Vault or AWS Secrets Manager
- Automatic key rotation every 90 days
- No secrets committed to version control
- Environment-specific secrets segregation

Container Security:

- Minimal base images (Alpine Linux) to reduce attack surface
- Non-root user execution inside containers
- Regular vulnerability scanning with Trivy
- Read-only root filesystem with explicit writable volumes

Monitoring & Incident Response:

- Real-time alerting for suspicious payment patterns
- Automated security scanning in CI/CD pipeline
- Centralized logging with tamper-proof audit trails
- 24/7 uptime monitoring with PagerDuty escalation

3. x402 Payment Integration

3.1 x402 Protocol Overview

The x402 protocol extends HTTP 402 Payment Required to enable seamless cryptocurrency payments for API access. The protocol eliminates traditional payment friction by allowing AI agents and applications to pay per request without accounts, subscriptions, or API keys.

Payment Flow:

1. Client sends unauthenticated request to protected endpoint
2. Server returns HTTP 402 with x402 payment challenge headers
3. Client constructs payment transaction to specified address
4. Facilitator validates payment and issues cryptographic proof
5. Client retries request with payment proof in X-Payment-Proof header
6. Server validates proof and returns protected resource

3.2 Facilitator Selection

Primary Option: PayAI Facilitator

PayAI provides production-ready x402 facilitation with native Solana support:

- Multi-network support: Solana, Base, Avalanche, Polygon, Sei
- Zero setup required - start accepting payments in under 1 minute
- No API keys needed for basic integration
- Supports USDC, USDT, and major tokens on each chain
- ~0.5% transaction fee (competitive with alternatives)

Alternative Options:

- thirdweb Facilitator: 170+ chains, 4000+ tokens, enterprise features
- CDP Facilitator: Coinbase integration, fee-free USDC on Base
- Self-hosted x402.rs: Open-source Rust implementation for full control

Recommended Approach:

Start with PayAI for rapid MVP deployment, then evaluate self-hosting x402.rs for cost optimization once transaction volume exceeds 10,000 monthly requests.

3.3 Implementation Details

HTTP 402 Response Structure:

When an unpaid request arrives, the server returns:

```
HTTP/1.1 402 Payment Required
X-Payment-Required: x402
X-Payment-Amount: 0.05
X-Payment-Currency: USDC
X-Payment-Network: solana
X-Payment-Address: <facilitator_address>
X-Payment-Nonce: <unique_nonce>
```

Payment Proof Validation:

The server validates payment proofs through the following steps:

1. Extract X-Payment-Proof header from authenticated request

2. Verify proof signature matches facilitator's public key
3. Confirm payment amount matches endpoint pricing
4. Check nonce hasn't been used previously (replay protection)
5. Verify payment timestamp within acceptable window (5 minutes)
6. Cache validated proof to allow retries within TTL

3.4 Pricing Strategy

Endpoint Pricing Tiers:

- Basic wallet overview: 0.01 USDC per query
- Standard portfolio analysis: 0.05 USDC per query
- Comprehensive risk assessment: 0.10 USDC per query
- Historical activity analysis: 0.15 USDC per query

Cost Structure Analysis:

For standard 0.05 USDC endpoint:

- Solana RPC calls: ~$0.0001 (with caching)
- Price feed API: ~$0.001
- Compute & infrastructure: ~$0.002
- Facilitator fee (0.5%): ~$0.00025
- Total cost: ~$0.003
- Gross margin: ~94% ($0.047 per query)

3.5 Error Handling

Payment-Related Errors:

- 402 Payment Required: Initial challenge response
- 400 Bad Request: Malformed payment proof
- 403 Forbidden: Invalid payment signature or replay attempt
- 409 Conflict: Nonce already used
- 410 Gone: Payment proof expired (outside 5-minute window)
- 503 Service Unavailable: Facilitator temporarily unreachable

4. API Endpoints & Data Models

4.1 Core Endpoints

GET /api/v1/wallet/{address}/portfolio

Returns comprehensive portfolio analysis for a Solana wallet address.

Price: 0.05 USDC per request

Rate Limit: 100 requests per hour per paying wallet

Cache TTL: 5 minutes

Response Schema:

```
{
    "address": "string",
    "totalValueUSD": number,
    "solBalance": number,
    "tokens": TokenHolding[],
    "nfts": NFTHolding[],
    "defiPositions": DefiPosition[],
    "updatedAt": "ISO8601 timestamp"
}
```

GET /api/v1/wallet/{address}/activity

Returns transaction history and activity metrics for a wallet.

Price: 0.10 USDC per request

Query Parameters:

- days: Number of days to analyze (default: 30, max: 365)
- limit: Maximum transactions to return (default: 100, max: 1000)

Response Schema:

```
{
    "address": "string",
    "metrics": {
        "totalTransactions": number,
        "avgDailyTransactions": number,
        "firstTransactionDate": "ISO8601",
        "lastTransactionDate": "ISO8601",
        "mostUsedPrograms": string[]
    },
    "recentTransactions": Transaction[]
}
```

GET /api/v1/wallet/{address}/risk

Returns risk assessment and security analysis for a wallet.

Price: 0.10 USDC per request

Response Schema:

```
{
    "address": "string",
```

```
    "riskScore": number, // 0-100, higher = riskier
    "riskLevel": "low" | "medium" | "high" | "critical",
    "flags": {
        "interactedWithScams": boolean,
        "unusuallyHighActivity": boolean,
        "newWallet": boolean,
        "botLikeBehavior": boolean
    },
    "details": string[]
}
```

GET /api/v1/wallet/{address}/overview

Returns basic wallet summary (lightweight, fastest endpoint).

Price: 0.01 USDC per request

Response Schema:

```
{
    "address": "string",
    "solBalance": number,
    "totalValueUSD": number,
    "tokenCount": number,
    "nftCount": number,
    "isActive": boolean
}
```

4.2 Data Models

TokenHolding Model:

```
{
    "mint": "string", // Token mint address
    "symbol": "string",
    "name": "string",
    "balance": number,
    "decimals": number,
    "priceUSD": number,
    "valueUSD": number,
    "percentOfPortfolio": number
}
```

NFTHolding Model:

```
{
    "mint": "string",
    "collectionName": "string",
    "name": "string",
    "imageUrl": "string",
    "floorPriceSOL": number,
    "estimatedValueUSD": number
}
```

DefiPosition Model:

```
{
    "protocol": "string", // e.g., "Marinade", "Jupiter"
    "type": "staking" | "liquidity" | "lending",
    "tokens": string[], // Token symbols
    "valueUSD": number,
    "details": object // Protocol-specific data
}
```

5. Data Sources & Aggregation

5.1 Primary Data Sources

Solana RPC Infrastructure:

Core blockchain data retrieved via RPC methods:

- getBalance: Native SOL balance
- getTokenAccountsByOwner: SPL token holdings
- getProgramAccounts: DeFi protocol positions
- getSignaturesForAddress: Transaction history

RPC Provider Options:

Option A - Helius (Recommended for MVP)

- Free tier: 100,000 credits/month
- Enhanced APIs: Digital Asset API, Webhooks, Name Service
- Excellent documentation and support

Option B - QuickNode

- Free tier: 100,000 requests/month
- Marketplace add-ons for token data and NFTs
- Global infrastructure with low latency

Option C - Self-Hosted Validator (Production Scale)

- Full control and zero per-request costs
- High infrastructure overhead (~$500-1000/month)
- Justified only above 1M+ monthly requests

Price Feed Aggregation:

Jupiter Aggregator API (Primary)

- Free tier: Unlimited requests with rate limits
- Real-time prices from 20+ Solana DEXes
- Price endpoint: https://price.jup.ag/v4/price?ids={mints}

Birdeye API (Fallback)

- Free tier: 100 requests/minute
- Historical price data and trading volume
- Multi-chain support for future expansion

CoinGecko API (Legacy tokens)

- Free tier: 10-50 calls/minute
- Best coverage for established tokens
- 5-minute cache recommended

NFT Marketplace Data:

Magic Eden API

- Collection stats and floor prices
- NFT metadata and images
- Rate limits: Documented per endpoint

Tensor API

- Professional NFT analytics
- Real-time floor price tracking
- API key required (free tier available)

Metaplex SDK

- On-chain NFT metadata parsing
- No API limits (direct blockchain access)
- Best for individual NFT lookups

5.2 Data Aggregation Strategy

Caching Architecture:

Multi-layer caching reduces external API costs and improves response times:

Layer 1: Redis Cache (Primary)

- Wallet portfolio data: 5-minute TTL
- Token prices: 2-minute TTL
- NFT floor prices: 10-minute TTL
- Activity metrics: 15-minute TTL

Layer 2: CDN Edge Caching (Secondary)

- Static metadata and token info: 1-hour TTL
- API documentation and schemas: 24-hour TTL
- Geographic distribution reduces latency

Layer 3: Application Memory (Tertiary)

- Token metadata (symbol, decimals): Indefinite
- Known scam addresses: Daily refresh
- Program IDs and well-known protocols: Indefinite

Fallback & Resilience:

- Primary source failure triggers automatic fallback to secondary
- Stale data served with warning header if all sources unavailable
- Circuit breaker pattern prevents cascade failures
- Exponential backoff for failed upstream requests

6. Performance & Scalability

6.1 Performance Targets

Response Time SLAs:

- Basic overview endpoint: < 200ms (p95)
- Portfolio analysis: < 800ms (p95)
- Activity history: < 1500ms (p95)
- Risk assessment: < 1000ms (p95)

Throughput Capacity:

- Target: 1000 requests per second (peak)
- Sustained: 500 requests per second
- Cache hit rate: > 80% for portfolio queries

Availability SLA:

- Uptime target: 99.5% (43.8 hours downtime/year)
- Planned maintenance windows: Monthly, announced 48h advance
- Zero-downtime deployments via blue-green strategy

6.2 Scalability Architecture

Horizontal Scaling:

- Stateless API servers enable infinite horizontal scaling
- Load balancer distributes traffic across server pool
- Auto-scaling based on CPU and request queue depth
- Kubernetes deployment with HPA (Horizontal Pod Autoscaler)

Database Scaling:

- Redis Cluster for distributed caching (6+ nodes)
- PostgreSQL with read replicas for analytics queries
- Connection pooling via PgBouncer to prevent exhaustion
- Time-series data partitioned by month in TimescaleDB

Geographic Distribution:

- Multi-region deployment (US-East, US-West, EU, Asia)
- GeoDNS routes clients to nearest region
- Redis replication across regions for cache coherency
- Active-active configuration with automatic failover

6.3 Optimization Techniques

Query Optimization:

- Parallel data fetching from multiple sources via Promise.all
- Batch token price lookups (max 100 mints per request)
- Lazy loading of expensive data (DeFi positions, history)
- Streaming large result sets to avoid memory pressure

Resource Management:

- Connection pooling for all external services
- Request timeouts prevent hanging operations
- Worker threads for CPU-intensive calculations
- Memory limits enforced per request to prevent leaks

Monitoring & Observability:

- Distributed tracing with OpenTelemetry
- Real-time metrics exported to Prometheus
- Custom dashboards in Grafana for key metrics
- Automated alerts for anomalies and degradation

7. Implementation Roadmap

7.1 Phase 1: MVP (Weeks 1-3)

Week 1: Foundation

- Set up development environment and repository
- Initialize Fastify server with TypeScript configuration
- Integrate PayAI facilitator for x402 payments
- Implement basic HTTP 402 middleware
- Create wallet address validation utilities
- Connect to Helius RPC (free tier)

Week 2: Core Features

- Implement /api/v1/wallet/{address}/overview endpoint
- Build token balance aggregation logic
- Integrate Jupiter API for token pricing
- Add Redis caching layer with 5-minute TTL
- Implement basic error handling and logging
- Write unit tests for core functions

Week 3: Testing & Launch

- Deploy to staging environment (Railway or Fly.io)
- Conduct end-to-end payment flow testing
- Set up basic monitoring with Prometheus
- Create API documentation with examples
- Deploy to production and announce to x402 ecosystem
- Submit to x402.org ecosystem directory

7.2 Phase 2: Enhanced Features (Weeks 4-6)

Portfolio Analysis:

- Implement /api/v1/wallet/{address}/portfolio endpoint
- Add NFT detection via Metaplex SDK
- Integrate Magic Eden API for floor prices
- Calculate portfolio composition percentages
- Optimize parallel data fetching

DeFi Integration:

- Detect Marinade staking positions
- Parse Jupiter LP token holdings
- Add Orca liquidity pool detection
- Calculate total DeFi position values

7.3 Phase 3: Advanced Analytics (Weeks 7-9)

Activity Analysis:

- Implement /api/v1/wallet/{address}/activity endpoint
- Parse and categorize transaction types

- Calculate daily/weekly activity metrics
- Identify most-used programs and protocols

Risk Assessment:

- Implement /api/v1/wallet/{address}/risk endpoint
- Integrate scam database for address checks
- Detect bot-like behavior patterns
- Calculate composite risk scores
- Generate human-readable risk explanations

7.4 Phase 4: Production Hardening (Weeks 10-12)

Reliability & Performance:

- Implement circuit breaker for upstream failures
- Add fallback data sources for resilience
- Optimize caching strategies based on metrics
- Conduct load testing and performance tuning

Observability:

- Deploy comprehensive Grafana dashboards
- Set up alerting for critical metrics
- Implement distributed tracing
- Create runbooks for common incidents

8. Operational Considerations

8.1 Cost Analysis

Monthly Infrastructure Costs (10,000 requests):

- Hosting (Railway/Fly.io): $20-40
- Redis Cloud (500MB): $0 (free tier)
- PostgreSQL (Neon/Supabase): $0 (free tier)
- Helius RPC: $0 (free tier sufficient)
- Monitoring: $0 (Grafana Cloud free tier)
- Domain + SSL: $12/year

Total: ~$30-50/month

Revenue Projection (10,000 requests @ $0.05):

- Gross revenue: $500
- Facilitator fees (0.5%): -$2.50
- Infrastructure costs: -$40
- Net profit: $457.50

Profit margin: 91.5%

8.2 Maintenance & Support

Regular Maintenance:

- Weekly: Review error logs and performance metrics
- Monthly: Update dependencies and security patches
- Quarterly: Capacity planning and cost optimization
- Annually: Security audit and penetration testing

Incident Response:

- Automated alerting for service degradation
- On-call rotation for critical incidents
- Post-mortem analysis for all outages
- Status page for transparency (status.yourapi.com)

8.3 Legal & Compliance

Terms of Service:

- No warranty or liability for financial decisions
- Data sourced from public blockchain and third parties
- Prices subject to change with 30-day notice
- Service availability not guaranteed (best effort)

Privacy Policy:

- No personal information collected
- Payment proofs retained for audit (90 days)
- Aggregated analytics may be published
- Compliance with GDPR, CCPA via minimization

Financial Regulations:

- Not a financial advisor - informational purposes only
- AML/KYC handled by payment facilitator
- Tax reporting responsibility on end users
- Compliance monitoring for sanctions screening

8.4 Growth Strategy

Marketing Channels:

- Submit to x402.org ecosystem directory
- Engage with AI agent developer communities
- Demo integrations with MCP servers
- Twitter threads showcasing use cases
- Blog posts on x402 payment integration

Partnership Opportunities:

- AI agent frameworks (AutoGPT, LangChain)
- Crypto analytics platforms (Nansen, Dune)
- DeFi protocols needing wallet verification
- NFT marketplaces for buyer/seller insights

Conclusion

This implementation specification provides a comprehensive roadmap for building a Solana Wallet Analytics API with x402 payment integration. The architecture prioritizes security, privacy, and performance while maintaining operational simplicity for rapid MVP deployment.

Key Success Factors:

- Leveraging x402 protocol eliminates account friction and enables instant payments
- Pay-per-use model captures value from both casual and power users
- High margins (90%+) ensure profitability at modest scale
- Growing AI agent ecosystem provides expanding market opportunity
- Minimal compliance burden through privacy-by-design architecture

By following this specification, development can proceed with confidence that security, scalability, and user experience requirements are properly addressed. The phased roadmap enables iterative value delivery while maintaining production quality standards.

Next Steps:

1. Review and approve this specification
2. Set up development infrastructure and tools
3. Begin Phase 1 implementation following the roadmap
4. Iterate based on early user feedback and usage patterns