UAIP Protocol

Quick Start Guide

Get Running in 10 Minutes

Version: 1.0.0
Last Updated: January 2025
Repository: https://github.com/jahanzaibahmad112-dotcom/UAIP-Protocol

---

Table of Contents

What is UAIP?

UAIP (Universal Agent Interoperability Protocol) is the secure settlement layer for autonomous AI agents. Think of it as **"Stripe + Auth0"** for AI agents.

The Problem

AI agents from OpenAI, Anthropic, and Microsoft can't securely transact with each other because there's no:

- ❌ Cryptographic identity verification
- ❌ Secure payment rails
- ❌ Automated compliance checking
- ❌ Audit trails

The Solution

UAIP provides:

- ✅ **Zero-Knowledge Identity Proofs** - Unfakeable authentication
- ✅ **Multi-Chain Settlement** - USDC on Base, Solana, Ethereum
- ⚖️ **Automated Compliance** - EU AI Act, SOC2, GDPR
- ✅ **Complete Audit Trails** - Forensic logging

When to Use UAIP

Use UAIP when you need:

- AI agents to pay each other for services
- Cross-company agent collaboration
- Cryptographic proof of agent actions
- Regulatory compliance for autonomous systems

Installation

Prerequisites

- **Python 3.10+** (Check: `python --version`)
- **pip** (Package manager)
- **Git** (Optional, for cloning)

Step 1: Clone or Download

Option A: Clone with Git

```
git clone https://github.com/jahanzaibahmad112-dotcom/UAIP-Protocol.git
cd UAIP-Protocol
```

Option B: Download ZIP

1. Go to https://github.com/jahanzaibahmad112-dotcom/UAIP-Protocol
2. Click "Code" → "Download ZIP"
3. Extract and navigate to folder

Step 2: Install Dependencies

```
pip install fastapi uvicorn pynacl requests python-multipart
```

Verify installation:

```
python -c "import fastapi, nacl, requests; print('✅ All dependencies installed')"
```

Step 3: Set Environment Variables

Linux/Mac:

```
export ADMIN_KEY="your-secure-admin-key-min-32-characters-long"
export ALLOWED_ORIGINS="http://localhost:3000"
export DB_PATH="uaip_vault.db"
```

Windows (PowerShell):

```
$env:ADMIN_KEY="your-secure-admin-key-min-32-characters-long"
$env:ALLOWED_ORIGINS="http://localhost:3000"
$env:DB_PATH="uaip_vault.db"
```

Windows (Command Prompt):

```
set ADMIN_KEY=your-secure-admin-key-min-32-characters-long
set ALLOWED_ORIGINS=http://localhost:3000
set DB_PATH=uaip_vault.db
```

⚠ **IMPORTANT:** Use a strong, random ADMIN_KEY in production!

First Transaction in 5 Minutes

Step 1: Start the Gateway (30 seconds)

Open a terminal and run:

```
python gateway.py
```

You should see:

```
INFO:     Started server process
INFO:     Waiting for application startup.
INFO:     Application startup complete.
INFO:     Uvicorn running on http://0.0.0.0:8000
```

✅ **Gateway is now running!**

Open your browser to http://localhost:8000 to see the dashboard.

Step 2: Run the Demo (2 minutes)

Open a **NEW terminal** (keep the gateway running) and run:

```
python demo.py
```

You'll see:

```
 UAIP Protocol Demo: Cross-Ecosystem Agent Transaction
═══════════════════════════════════════════════════════

 Scenario:
  Agent A (OpenAI Ecosystem) needs to pay Agent B (Anthropic Ecosystem)
  for data analysis services.

 Step 1: Generating Agent Identities...
   Agent A registered: did:uaip:openai-corp:abc123...
   Agent B registered: did:uaip:anthropic-inc:def456...

 Step 2: Executing Transaction...
  Amount: $150.00
  Task: Analyze Q4 financial data
  Chain: BASE

 Processing...

 SUCCESS! Transaction completed.
```

What just happened?

1. Two agents were created and registered
2. Agent A paid Agent B $150 for a service
3. Cryptographic proofs verified both identities
4. Compliance checked against legal frameworks
5. Payment settled (simulated blockchain)
6. Complete audit trail generated

Step 3: View the Transaction (1 minute)

Go back to your browser at http://localhost:8000

You'll see the transaction in the dashboard with:

- Agent DIDs
- Transaction amount
- Compliance status
- Legal framework grounding
- Timestamp

 Congratulations! You just executed your first UAIP transaction.

Step 4: Check the Audit Log (30 seconds)

Look in your project folder for:

```
uaip_forensic_records.json
```

Open it to see the complete audit trail in JSON format:

```
{
  "audit_id": "AUDIT-A3F5B2C8",
  "timestamp": "2025-01-15T10:30:00Z",
  "agent": "did:uaip:openai-corp:abc123",
  "task": "Analyze Q4 financial data",
  "amount": 150.00,
  "status": "PASSED",
  "grounded_law": "UAIP Policy: Routine transaction logging"
}
```

Core Concepts

1. Agents

What: Autonomous AI systems that can transact

Identity: Each agent has:

- **DID** (Decentralized Identifier): `did:uaip:company:hash`
- **Ed25519 Key Pair**: For signing transactions
- **ZK Commitment**: For privacy-preserving authentication

Example:

```
from sdk import UAIP_Enterprise_SDK

agent = UAIP_Enterprise_SDK(
    agent_name="FinanceBot",
    company_name="Acme Corp",
    secret_code=12345678  # Keep this secure!
)

print(agent.did)  # did:uaip:acme-corp:a3f5b2c8
```

---

2. Transactions

What: A payment from one agent to another

Required Fields:

- `task` : What the agent is doing (e.g., "process_invoice")
- `amount` : How much to pay (in USD)
- `intent` : Why (e.g., "Q1 vendor payment")
- `chain` : Which blockchain (BASE, SOLANA, ETHEREUM, POLYGON)

Example:

```
result = agent.call_agent(
    task="analyze_data",
    amount=50.00,
    intent="Financial analysis for Q1 report",
    chain="BASE"
)

if result['status'] == 'SUCCESS':
    print(f"✅ Paid ${result['amount']}")
    print(f"Fee: ${result['fee']}")
```

---

3. Zero-Knowledge Proofs

What: Prove you know a secret without revealing it

Why: Agents can authenticate without exposing their secret keys

How it works:

1. Agent has secret key `x`
2. Agent generates public commitment `y = G^x mod P`
3. Agent creates proof they know `x` without showing `x`
4. Gateway verifies proof

You don't need to understand the math—the SDK handles it automatically!

---

4. Compliance Checking

What: Automatic verification against legal frameworks

Checks:

- ✅ No prohibited keywords (laundering, fraud, etc.)
- ✅ Amount within risk thresholds
- ✅ Human approval for high-value transactions (≥$1,000)
- ✅ Complete audit trail generated

Example:

```
# This will be BLOCKED automatically
result = agent.call_agent(
    task="offshore money laundering",  # ❌ Prohibited keyword
    amount=10.00,
    intent="Transfer funds"
)
# Result: TERMINATE with compliance violation

# This will require HUMAN APPROVAL
result = agent.call_agent(
    task="large_payment",
    amount=50000.00,  # ≥$1,000 triggers human review
    intent="Enterprise license payment"
)
# Result: PENDING_APPROVAL
```

---

5. Settlement Chains

What: Blockchains where payments are processed

Options:

Chain	Fee	Speed	Best For
BASE	$0.01	2 sec	Most transactions (default)
Solana	$0.0001	<1 sec	High-frequency operations
Ethereum	$2-5	12 sec	Large amounts, max security
Polygon	$0.01	2 sec	Enterprise, backup

Example:

```
# Use Solana for fast, cheap transactions
result = agent.call_agent(
    task="micro_payment",
    amount=0.50,
    intent="API call payment",
    chain="SOLANA"  # Ultra-low fees
)
```

---

6. Fee Structure

3-Tier Pricing:

Tier A: Nano (≤ $10)

- Fee: **$0.01 flat**
- Example: $5 payment → $0.01 fee (0.2%)

Tier B: Mid-Range ($10 - $10,000)

- Fee: **1.0% of amount**
- Example: $1,000 payment → $10 fee (1.0%)

Tier C: Enterprise (> $10,000)

- Fee: **$10 + 0.5% of amount**
- Example: $100,000 payment → $510 fee (0.51%)

Check fee before transacting:

```
from settlement import UAIPFinancialEngine

engine = UAIPFinancialEngine()
projection = engine.calculate_projected_fee(amount=1500.00)

print(f"Amount: ${projection['amount']}")
print(f"Fee: ${projection['fee']} ({projection['fee_percentage']:.2f}%)")
print(f"Tier: {projection['tier_name']}")
print(f"You'll pay: ${projection['payout']}")
```

Common Use Cases

Use Case 1: Cross-Company Agent Collaboration

Scenario: GPT agent (OpenAI) needs Claude agent (Anthropic) to analyze data

```
from sdk import UAIP_Enterprise_SDK

# OpenAI agent
gpt_agent = UAIP_Enterprise_SDK(
    agent_name="GPT-DataFetcher",
    company_name="OpenAI",
    secret_code=111111
)

# Pay Claude agent for analysis
result = gpt_agent.call_agent(
    task="analyze_customer_sentiment",
    amount=75.00,
    intent="Sentiment analysis for product feedback",
    chain="BASE"
)

print(f"Analysis cost: ${result['fee']} fee, ${result['payout']} to Claude agent")
```

Use Case 2: Enterprise Workflow Automation

Scenario: Salesforce AgentForce agent pays ServiceNow agent for ticket data

```
# Salesforce agent
sf_agent = UAIP_Enterprise_SDK(
    agent_name="SalesforceIntegrator",
    company_name="Acme Corp - Salesforce Org",
    secret_code=222222
)

# Pay ServiceNow agent for support ticket data
result = sf_agent.call_agent(
    task="fetch_support_tickets",
    amount=25.00,
    intent="Sync support tickets for customer 360 view",
    chain="BASE",
    metadata={"customer_id": "CUST-12345"}
)
```

Use Case 3: AI Agent Marketplace

Scenario: Specialist agents selling services to other agents

```
# Translation agent sells service
translation_agent = UAIP_Enterprise_SDK(
    agent_name="TranslationSpecialist",
    company_name="LangChain Services",
    secret_code=333333
)

# Another agent buys translation
buyer_agent = UAIP_Enterprise_SDK(
    agent_name="ContentBot",
    company_name="Media Corp",
    secret_code=444444
)

result = buyer_agent.call_agent(
    task="translate_article",
    amount=15.00,
    intent="Translate blog post from English to Spanish",
    chain="BASE"
)
```

Use Case 4: High-Value Enterprise Transaction

Scenario: Large payment requiring human approval

```
# Enterprise agent
enterprise_agent = UAIP_Enterprise_SDK(
    agent_name="TreasuryBot",
    company_name="BigCorp Inc",
    secret_code=555555
)

# Large payment triggers human review
result = enterprise_agent.call_agent(
    task="enterprise_software_license",
    amount=50000.00,  # ≥$1,000 requires approval
    intent="Annual enterprise license renewal",
    chain="ETHEREUM",  # Use most secure chain for large amounts
    wait_for_approval=True  # SDK will poll until admin approves
)

if result['status'] == 'PENDING_APPROVAL':
    print("Waiting for admin approval in dashboard...")
    # Admin goes to http://localhost:8000 and clicks "Approve"
```

Next Steps

For Developers

1. Read the API Reference

- Complete endpoint documentation
- All SDK methods explained
- Error handling patterns

2. Explore the Code

```
# Look at the implementation
gateway.py       # Main server
sdk.py          # Client integration
compliance.py   # Audit engine
settlement.py   # Financial processing
privacy.py      # Zero-Knowledge proofs
```

3. Customize for Your Use Case

- Modify fee structure
- Add custom compliance rules
- Integrate with your blockchain
- Build your own dashboard

For Enterprise Users

1. Deploy to Production

- Use PostgreSQL instead of SQLite
- Set up load balancer
- Configure SSL/TLS
- Enable monitoring

2. Configure Compliance

- Add company-specific policies
- Integrate with existing compliance tools
- Set up alert notifications
- Configure human approval workflows

3. Multi-Chain Strategy

- Decide which chains to support
- Set up wallet infrastructure
- Configure gas management
- Plan for chain failures

For Researchers

1. Study the Cryptography

- Zero-Knowledge proof implementation
- Ed25519 signature scheme
- Schnorr protocol details
- Fiat-Shamir heuristic

2. Analyze Security

- Threat modeling
- Attack surface analysis
- Timing attack resistance
- Formal verification opportunities

3. Extend the Protocol

- Add new consensus mechanisms
- Implement additional ZK schemes
- Create formal specifications
- Propose protocol improvements

Troubleshooting

Common Issues

Problem: "ADMIN_KEY must be set"

```
Solution: Export environment variable before starting:
export ADMIN_KEY="your-32-character-minimum-admin-key"
```

Problem: "Port 8000 already in use"

```
Solution: Kill existing process:
# Linux/Mac
lsof -ti:8000 | xargs kill -9

# Windows
netstat -ano | findstr :8000
taskkill /PID <PID> /F
```

Problem: "Module not found: fastapi"

```
Solution: Install dependencies:
pip install fastapi uvicorn pynacl requests
```

Problem: "Database is locked"

```
Solution: Only one gateway instance should run at a time.
Stop other instances or use a different DB_PATH.
```

Problem: "Rate limit exceeded"

```
Solution: Wait 60 seconds or increase rate limit in gateway.py:
RATE_LIMIT_MAX_REQUESTS = 200  # Default is 100
```

Getting Help

Documentation

- **API Reference:** See API-Reference.pdf
- **Architecture Guide:** See Architecture-Deep-Dive.pdf
- **Security Guide:** See Security-Compliance-Guide.pdf

Community

- **GitHub Issues:** https://github.com/jahanzaibahmad112-dotcom/UAIP-Protocol/issues
- **Twitter:** @UAIPProtocol
- **Email:** [your-email]

Support

For enterprise support, security questions, or partnership inquiries:

- Email: [your-email]
- LinkedIn: [your-linkedin]

Quick Reference

Start Gateway

```
export ADMIN_KEY="your-secure-key-32-chars-minimum"
python gateway.py
```

Create Agent

```
from sdk import UAIP_Enterprise_SDK
agent = UAIP_Enterprise_SDK(
    agent_name="MyBot",
    company_name="My Company",
    secret_code=12345678
)
```

Execute Transaction

```
result = agent.call_agent(
    task="task_description",
    amount=50.00,
    intent="why_you're_doing_this",
    chain="BASE"
)
```

Check Fee

```
from settlement import UAIPFinancialEngine
engine = UAIPFinancialEngine()
fee_info = engine.calculate_projected_fee(100.00)
```

View Dashboard

```
Open browser: http://localhost:8000
```

License

UAIP Protocol is licensed under **FSL-1.1-Apache-2.0** (Functional Source License).

What this means:

- ✅ **Free for:** Personal use, internal business use, development/testing
- ✅ **Free for:** Open-source projects
- ❌ **Requires license for:** Managed services, SaaS offerings
- 🔄 **Becomes Apache 2.0 in:** 2 years (fully open source)

Full license: https://fsl.software/

What's Next?

Now that you've completed the Quick Start:

1. 📖 Read **API Reference** for complete endpoint documentation
2. 🏗️ Read **Architecture Deep Dive** to understand internals
3. 🔒 Read **Security & Compliance Guide** for production deployment
4. ⭐ Star the GitHub repo: https://github.com/jahanzaibahmad112-dotcom/UAIP-Protocol
5. 🐦 Follow on Twitter: @UAIPProtocol

Ready to build the autonomous economy! 🚀

Document Version: 1.0.0
Last Updated: January 15, 2025
UAIP Protocol Project