UNIVERSAL CHAIN ABSTRACTION FOR WEB3

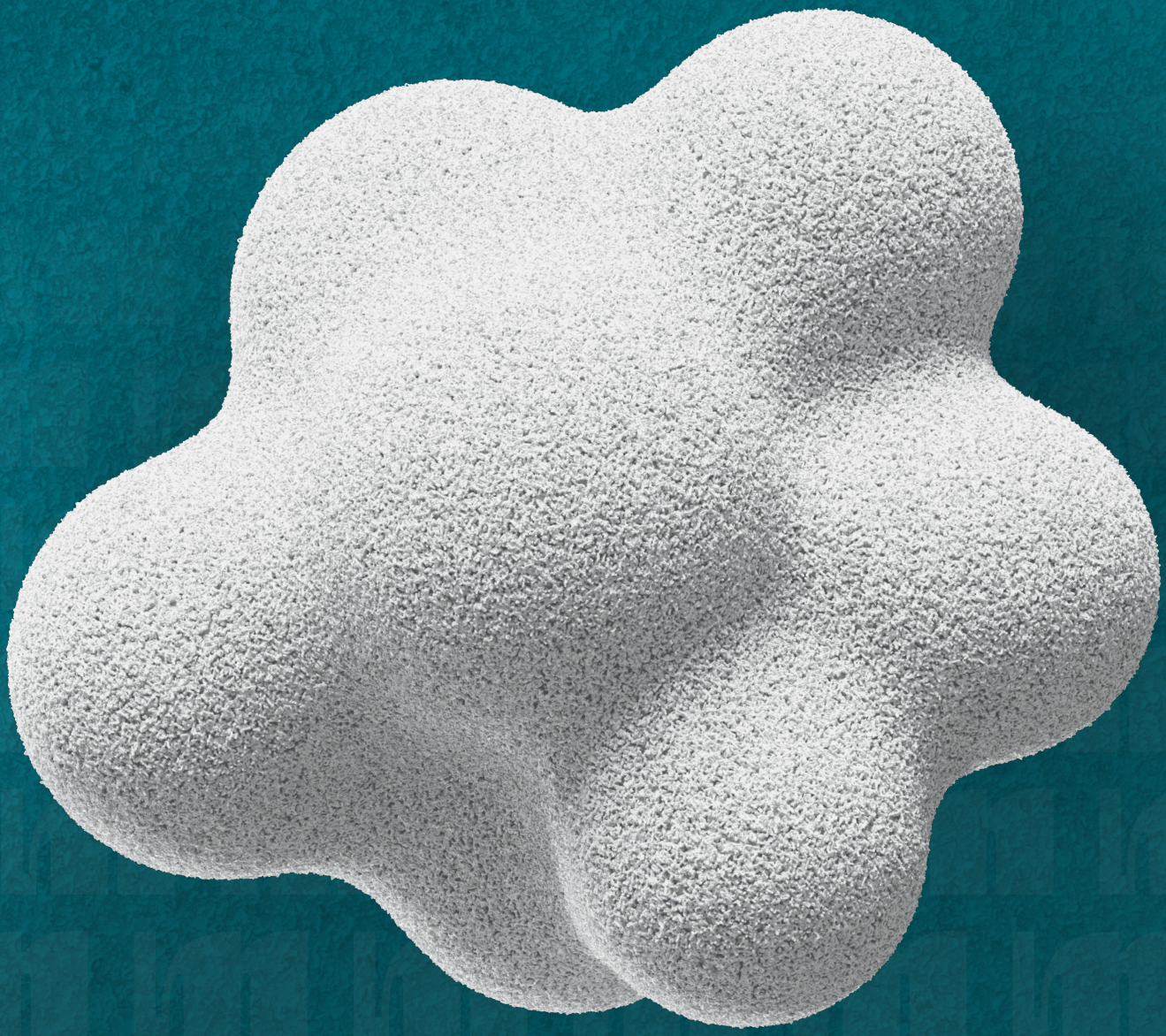


Table of Contents

Abstract

Mono Protocol introduces revolutionary chain abstraction technology that unifies per-token balances across all blockchain networks, enabling instant, guaranteed, MEV-resilient execution through universal accounts and simple APIs. Users can transact anywhere with one account, one balance, and one click.

Built on cutting-edge Liquidity Lock technology, Mono Protocol eliminates multi-chain complexity while providing developers with powerful tools to build seamless cross-chain applications. The protocol represents a paradigm shift in blockchain interaction, addressing fundamental Web3 fragmentation by making all blockchains feel like a single, cohesive network.

Introduction

What is Mono Protocol

Mono Protocol is a universal chain abstraction protocol that creates a unified execution environment for cross-chain interactions. The protocol combines three revolutionary technologies:

- **Universal Account System** for chain-abstracted identity
- **Liquidity Lock Technology** for guaranteed execution
- **MEV-Resistant Routing** for optimal value protection

Mission

To make Web3 feel like one network by unifying per-token balances across chains and delivering instant, guaranteed, MEV-resilient execution through universal accounts and simple APIs.

Vision

A future where blockchain networks are invisible to users, where multi-chain complexity is completely abstracted away, allowing users and developers to focus on building and using incredible applications.

Problem Statement

Current blockchain infrastructure faces critical challenges:

Multi-Chain Fragmentation

- **Balance Fragmentation:** Users manage separate token balances across different chains
- **Bridge Friction:** Complex bridging processes with long wait times and high feesMEV Exploitation: Users lose value to front-running and sandwich attacks

- **MEV Exploitation:** Users lose value to front-running and sandwich attacks
- **Poor UX:** Constant network switching creates friction
- **Developer Complexity:** Building cross-chain apps requires extensive infrastructure

Market Opportunity

- $200+ billion in multi-chain TVL across ecosystems
- 50+ major blockchain networks requiring interoperability
- Growing demand for chain-abstracted applications
- Universal chain abstraction enables true Web3 mass adoption

Solution Overview

Core Innovations

1. Mono Balance System

- Unified token balances across all supported chains
- Automatic balance aggregation and optimization
- Single interface for multi-chain assets

2. Liquidity Locks

- Advanced execution guarantee mechanisms
- Prevents failed cross-chain transactions
- Enables instant settlement with cryptographic guarantees

3. Chain Abstracted Operations

- Execute transactions on optimal chains automatically
- Support for any blockchain network or token
- Intelligent routing and MEV protection

4. Universal Gas

- Pay gas fees with any token on any chain
- Universal paymaster system eliminates native token requirements

Key Benefits

- **Speed:** Execute cross-chain transactions in seconds
- **Reliability:** Cryptographic guarantees ensure 100% success
- **Cost Efficiency:** Reduce gas costs by up to 40%
- **Simplified UX:** One account, one balance, one click
- **Developer-First:** Simple APIs abstract away complexity

Technical Architecture

Chain Abstraction Layer

Creates unified Web3 environment through intelligent routing and execution, eliminating individual blockchain network concepts from user experience.

Account Model

Universal smart wallets with:

- Programmable logic and custom validation rules
- Session management across all chains
- Social recovery mechanisms
- Cross-chain identity synchronization

Liquidity Lock Technology

Revolutionary mechanism that separates intent fulfillment from settlement:

1. **Verification:** Validates user balances and locked positions
2. **Co-signing:** Prevents double-spending during async execution
3. **Guarantees:** Issues security attestations for immediate delivery
4. **Tracking:** Ensures system integrity through risk management

Routing System

Automatically optimizes:

- Balance distribution analysis across chains
- Optimal execution path considering liquidity and fees
- MEV risk mitigation and timing
- Atomic settlement coordination

Supported Networks

Starting with major EVM chains and expanding:

- Ethereum, Optimism, Arbitrum, Polygon, Base
- Avalanche, BSC, Blast, Berachain
- Solana (Beta), Bitcoin (Planned)

Tokenomics

MONO Token Utility

1. Universal Gas & Protocol Fees
   - Paymaster fees for universal gas payments
   - Routing fees for optimal execution
   - API/SDK usage fees
   - Tiered staking discounts

2. Security & Governance

- Network security through operator bonds
- Governance rights for protocol parameters
- Fee distribution to stakers
- Asset listing and upgrade voting

3. Execution Bonds

- Performance guarantees for instant settlement
- Success premiums and failure penalties
- Insurance pool contributions

Token Distribution

Allocation	Percentage	Amount	Vesting
Presale	50%	500M	18 month
Liquidity	10%	100M	6 month
Private Round	5%	50M	24 month
Ecosystem	5%	50M	36 month
Treasury	5%	50M	DAO-controlled
User Rewards	5%	50M	18 month
Strategic	5%	50M	As needed
Team	5%	50M	24 month
Marketing	10%	100M	12 month

Economic model

- **Total Supply:** 1,066,056,144 MONO
- **Initial Circulating:** 141,252,439 MONO (13.3%)
- **Deflationary Mechanisms:** Fee burns, buybacks, staking locks
- **Revenue Distribution:** 40% to stakers, 30% treasury, 20% operations, 10% buyback

Roadmap

Q2 2025 — Foundation

- Core team assembly and architecture finalization
- Technical prototypes and security modeling
- Strategic partnerships with WaaS providers and oracles

Q3 2025 — Presale Launch

- Brand development and marketing materials
- Presale smart contracts and security audits
- Community building and developer preview

Q4 2025 — Beta Release

- Mainnet beta on major EVM L2s and initial Solana support
- Liquidity Locks MVP with solver bonding
- Fee abstraction and MEV protection integration

Q2 2026 — Market Expansion

- Scaled go-to-market strategy and PR campaigns
- Enterprise partnerships and co-marketing initiatives
- Regional expansion and localization efforts

Q1 2026 — Ecosystem Growth

- Mainnet release candidate with comprehensive audits
- Governance v1 and staking implementation
- Chain expansion and insurance pool deployment

Team

Mono Protocol is built by experienced blockchain engineers, cryptographers, and product designers with deep Web3 infrastructure expertise.

Core Competencies

- **Protocol Engineering:** Secure, scalable blockchain infrastructure
- **Cryptography:** Zero-knowledge proofs and consensus mechanisms
- **Product Design:** Multi-chain UX optimization
- **Business Development:** Ecosystem partnerships and integrations

Advisory Network

Leading figures from major blockchain protocols, prominent DeFi projects, and successful Web3 infrastructure companies provide strategic guidance and industry connections.

Advanced Concepts

Chain Abstracted Swap

Build seamless cross-chain swaps powered by **Mono Protocol** and **Privy**
Now that your project is set up with Privy and Mono Protocol, let’s create a user interface for chain-abstracted swaps. This is the part that makes users go “wow”—they can swap tokens across different chains without ever dealing with bridges, gas fees, or network settings.

Why It Works

- **Simplified UX:** Privy handles smooth wallet interactions.
- **Chain Abstraction:** Mono Protocol ensures users don’t need to think about networks, gas, or bridging.

Step 1:

We’ll start by creating type definitions for swap quotes and operations.
Add a new file at:

```ts
// src/lib/types/quote.ts
export interface Account {
  sessionAddress: string;
  adminAddress: string;
  accountAddress: string;
}

export interface TokenInfo {
  aggregatedAssetId: string;
  amount: string;
  assetType: string | string[];
  fiatValue: never;
}

export interface ChainOperation {
  userOp: {
    sender: string;
    nonce: string;
    callData: string;
    callGasLimit: string;
    verificationGasLimit: string;
    preVerificationGas: string;
    maxFeePerGas: string;
    maxPriorityFeePerGas: string;
    paymaster: string;
    paymasterVerificationGasLimit: string;
    paymasterPostOpGasLimit: string;
    paymasterData: string;
    signature: string;
  };
  typedDataToSign: {
    domain: unknown;
    types: unknown;
    primaryType: string;
    message: unknown;
  };
  assetType: string;
  amount: string;
}

export interface Quote {
  id: string;
  account: Account;
  originToken: TokenInfo;
  destinationToken: TokenInfo;
  expirationTimestamp: string;
  tamperProofSignature: string;
  originChainsOperations: ChainOperation[];
  destinationChainOperation?: ChainOperation;
}

export interface QuoteStatus {
  quoteId: string;
  status: {
    status: 'PENDING' | 'COMPLETED' | 'FAILED' | 'IN_PROGRESS' | 'REFUNDED';
  };
  user: string;
  recipientAccountId: string;
  originChainOperations: {
    hash: string;
    chainId: number;
    explorerUrl: string;
  }[];
  destinationChainOperations: {
    hash: string;
    chainId: number;
    explorerUrl: string;
  }[];
}
```

Executing a Swap

When a user initiates a swap, they'll be prompted to sign the transaction with their **Privy wallet.**

Using API Functions

Our chain-abstracted swap flow relies on the API. These are the three core endpoints you'll be working with:

- **GetQuote** — fetches a quote for cross-chain swaps or transfers.
- **ExecuteQuote** — submits a signed quote for execution.
- **CheckStatus** — polls the API to track transaction status.

Building the Dashboard Page

Next, let's build a dashboard that allows users to swap USDC ↔ ETH with automatic transaction status updates.

Chain-Abstracted Swap Flow: Key Features

Bidirectional Swapping

- Users can swap **USDC → ETH** or **ETH → USDC.**
- The UI updates dynamically based on direction.

Real-Time Quote Estimation

- Entering an amount triggers a quote request.
- Users see the estimated output instantly for transparency.

Balance Display

- Aggregates USDC and ETH balances across all supported chains.
- Balances refresh automatically after successful swaps.

Rmpust Transaction Polling

- Polls transaction status until completion.
- Handles **pending**, **completed**, and **failed** states.
- Stops polling once the transaction finalizes.

Error Handling & Recovery

- Displays clear error messages.
- Cleans up polling intervals to prevent resource leaks.

Enhanced User Experience

- Loading indicators during swap operations.
- Visual feedback for transaction states.
- Links to block explorers for full transparency.

Multichain Asset Visibility

A core strength of this implementation is how it abstracts away multichain complexity:

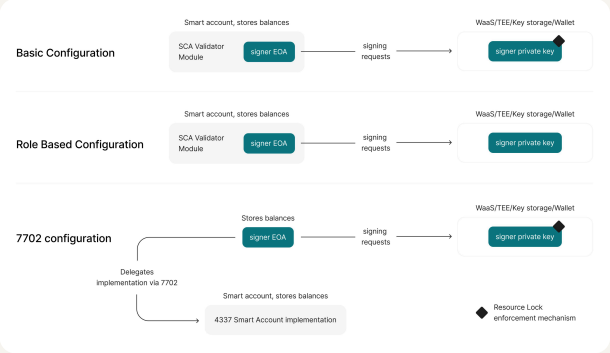
- Users see **one aggregated balance** for each token across chains.
- They choose assets (USDC, ETH) without worrying about networks.
- The system handles cross-chain interactions under the hood.

From the user's perspective, tokens exist in a **unified asset space**, making the experience as simple as swapping within a single network.

Account Model

Mono Protocol implements a revolutionary account model that abstracts away chain-specific details while maintaining security and user control.

Configuration	Validator	Version	Deployment	Different Options
Basic	ECDSAValidator	Kernel 3.1	ERC-4337	Can be enabled
EIP-7702	ECDSAValidator	Kernel 3.3	EIP-7702	Can be enabled
Role-Based	RoleBased Validator	Kernel 3.1	ERC-4337	Always enabled



Getting Started with Account Models

Follow the Getting Started guide, which uses Role-Based configuration.

Signer Compatibility

The Mono Protocol Toolkit is designed to work with various signer providers:

Signer	Account	Standard Path (no RL)	Enabling Lock
turnkey	Basic	● Available	● Available
	EIP-7702	● Available	● Available
	Role-Based	RL can't be disabled	● Available
privy	Basic	● Available	● Available
	EIP-7702	● Available	In design
	Role-Based	RL can't be disabled	● Available
	solana	Early Q3	In design
Other signers	Other accounts	On demand, case by case	On demand

Fees & Monetizations

Mono Protocol provides multiple revenue streams and monetization options for applications building on the platform.

Category	Fee Type	Who Pays	Description
Gas Fees	Cross-Chain Source or same chain execution	Application ⇒ Paymaster	The app pays for the inclusion to the paymaster service in USD
Gas Fees	Cross-Chain Destination	User⇒Solver	This fee is included in the quote the solver provides for execution
Monetization	Swaps & Transfers	User⇒App	App configures the fees and collects them to a dedicated address. The fee is paid in the input token and delivered with every user transaction
Monetization	Function Calls	User⇒App	App calculates the fee it is willing to take and encodes this into the desired calldata to be executed as a transfer

Aggregated Assets

Aggregated assets are unified token representation. Instead of managing USDC separately on Ethereum, Polygon, and Arbitrum, you work with a single identifier — e.g., **mp:usdc** — that represents all of your USDC holdings across supported chains.

Think of this as your **universal wallet view:** one asset ID that automatically abstracts away cross-chain complexity.

How Aggregated Assets Work

When you interact with an aggregated asset like mp:usdc, Mono Protocol automatically:

- **Aggregates balances** across every chain where you hold the token.
- **Optimizes routing** to use the most efficient chain for spending
- **Unifies pricing** with consistent fiat value calculations.

Supported Aggregated Assets

Each aggregated asset includes:

- **Unique ID** (e.g., mp:usdc, mp:eth)
- **Symbol & Name** (human-readable identifiers)
- **Decimals** (precision for the asset)
- **Underlying assets** (the chain-specific tokens that make up the aggregate)

You can also:

- **List supported assets** — fetch all aggregated assets available.
- **View underlying tokens** — see which chain-specific assets compose them.

Example: USDT Aggregated Asset

```
{
  "aggregatedAssetId": "mp:usdt",
  "symbol": "USDT",
  "name": "Tether USD",
  "decimals": 6,
  "aggregatedEntities": [
    {
      "assetType": "eip155:1/erc20:0xdAC17F958D2ee523a2206206994597C13D831ec7",
      "decimals": 6,
      "name": "Tether USD",
      "symbol": "USDT"
    },
    {
      "assetType": "eip155:42161/erc20:0xFd086bC7CD5C481DCC9C85ebE478A1C0b69FCbb9",
      "decimals": 6,
      "name": "Tether USD",
      "symbol": "USDT"
    }
  ]
}
```

Building with LLMs

We provide two continuously updated files designed for AI ingestion:

- **llms.txt** — A concise list of top-level pages for quick context.
- **llms-full.txt** — The complete documentation for full-context indexing.

Use these URLs in your custom GPTs or LLM-powered apps to get accurate, Mono Protocol-specific answers.

Plain Text Docs

Any documentation page can be accessed as Markdown by appending .md to the URL.

For example: /ai/building-with-llms.md

This format makes it easier for AI tools to consume content by providing:

- Fewer formatting tokens
- Complete content (including hidden tabs)
- Proper markdown hierarchy

Code Editor Integration

Cursor Setup

1. Go to **Cursor Settings > Indexing & Docs**
2. Select “**Add Doc**” and paste:

Then use:

@docs → Mono Protocol

to reference docs directly in your code.

MCP Server Integration

Hosted MCP (Recommended)

Connect directly to Mono Protocol via our hosted MCP server. Just add it to your AI tool (Claude, Cursor, etc.) for instant access to documentation search and live Mono Protocol API functions.

Self-Hosted MCP (Advanced)

For advanced users who prefer local setup:

npx mint-mcp add Mono Protocol-d7e5d4d0

Both options provide:

- Documentation search
- Live Mono Protocol API functions
- Seamless integration with AI-powered dev tools

MCP Server Setup

The Model Context Protocol (MCP) server connects Mono Protocol's documentation and API directly to your AI tools. You can choose:

- **Hosted MCP (Recommended):** Use a ready-to-go server hosted by Mono Protocol—quickest setup.
- **Self-Hosted MCP (Advanced):** Install and run the MCP server locally for flexibility and fine-tuning.

Hosted MCP Setup

For Claude

1. Navigate to **Connectors** in Claude's settings.
2. Choose **Add custom connector.**
3. Set the name to: **Mono Protocol MCP**

Use this URL:

1. https://docs.monoprotocol.com/mcp (you may want to adapt the domain to Mono Protocol).
2. Once added, you can access Mono Protocol's docs and API functions directly within Claude.

For Cursor

1. Press **Cmd + Shift + P (Mac)** or **Ctrl + Shift + P (Windows)** to open MCP settings.
2. Choose **Add custom MCP** to edit mcp.json.

Insert the following configuration:

```
{
  "mcpServers": {
    "Monorotocol-mcp": {
      "url": "https://docs.monoprotocol.com/mcp"
    }
  }
}
```

1. (Note: adjust the URL domain as needed.)
2. Then ask in Cursor chat: "What tools do you have available?" You should see Mono Protocol documentation search and API tools enabled.

Self-Hosted MCP (Advanced)

Ideal for users who want complete control via local installation:

Run:

npx mint-mcp add MonoProtocol-d7e5d4d0

1. You'll see a list of available tools:

- **search** – Documentation search
- **predict-smart-account-address** – Predict smart account addresses
- **list-supported-aggregated-assets** – View supported assets
- **get-aggregated-balance** – Fetch account balances
- **list-supported-chains** – View supported networks
- **get-quote-status** – Check quote execution status
- **get-transaction-history** – Retrieve transaction history
- **get-quote** – Request transfer/swap quotes
- **prepare-call-quote** – Prepare contract calls
- **get-call-quote** – Get contract call quotes
- **execute-quote** – Execute quotes
- You'll be prompted to enter your API key (or use the limited public test key).

Then start your server:

node ~/.mcp/MonoProtocol-d7e5d4d0/src/index.js

1. Use it seamlessly across clients like Cursor or Claude Desktop.

Available Functionality

Both hosted and self-hosted options include:

- **Documentation search** across Mono Protocol's docs.
- **API tools** for:
  - Predicting smart account addresses
  - Aggregated balances and asset queries
  - Chain support listing
  - Quote generation, retrieval, and execution
  - Transaction history and status tracking
  - Contract call preparation and execution

Example Usage Prompts

You can use prompts like:

- "What is my aggregated balance across all chains for address X?"
- "List supported chains."
- "Predict a smart account address for these signers..."
- These examples are typical usage, though primarily found in the MCP Prompts guide.

MCP Example Prompts

Once you've connected to **Mono Protocol MCP** (using your API key and either hosted or self-hosted setup), try these prompts tailored for Mono Protocol's capabilities:

Basic Prompts

Account & Balance Queries

- "What is my aggregated balance across all chains for address 0x..."
- "Show me all supported chains on Mono Protocol"
- "What aggregated assets are available on Mono Protocol?"
- "Predict a smart account address for session signer 0x... and admin signer 0x..."

Transaction Status & History

- "Check the status of quote ID '...'"
- "Show me the transaction history for address 0x..."
- "What's the latest transaction for my account?"

Quote Generation / Transfers

- "Get a quote to swap 10 USDC to ETH using my aggregated Mono Protocol"
- "How much would it cost to transfer 50 USDT to address 0x...?"
- "Show me a quote for swapping $500 worth of aggregated assets to WBTC"

Advanced Integration Prompts

Multi-Chain Operations

- "Create an interactive dashboard showing my aggregated balances, recent transactions, and pending quotes in real time."
- "Generate a quote comparison table for swapping 1000 USDC to ETH vs WBTC vs SOL—show best rates and execution costs across chains."
- "Analyze my transaction history over the last 30 days by chain and type, including total volume and fees."

Contract Interactions

- "Prepare a staking quote to stake 100 ETH on Ethereum Mainnet using my aggregated Mono Protocol, including gas estimates."
- "Get a quote for depositing 500 USDC into AAVE on Base via my cross-chain balance."
- "Create a batch transaction: swap 200 USDC → ETH, transfer 50 USDT to address 0x..., execute a contract call on Arbitrum—show combined quote and execution plan."

Portfolio Analysis

- “Build a portfolio dashboard: total balance in USD, breakdown by asset allocation, chain distribution, and 7-day performance, displayed interactively.”

Developer Integration Prompts

SDK & Tooling

- “Generate a TypeScript SDK wrapper for Mono Protocol’s API: include types, error handling, and example usage for all endpoints.”
- “Create a React hook library for Mono Protocol offering easy hooks for balance queries, transaction history, and quote generation.”

Testing & Monitoring

- “Build a test suite for Mono Protocol integration: cover all endpoints, error scenarios, and edge cases.”
- “Create a monitoring dashboard tracking Mono Protocol API performance, success rates, quota usage, with alerts for anomalies.”

Tips for Better Results

- **Be Specific:** include actual addresses, amounts, asset types to get accurate responses.
- **Ask for Explanations:** append “explain the process” to prompts to understand mechanics.
- **Request Multiple Formats:** ask for tables, charts, or interactive formats for better outcomes.

Risk Factors

Technical Risks

- Smart contract vulnerabilities and potential exploits
- Cross-chain bridge security dependencies
- Scalability limitations during high network congestion
- Integration complexities with new blockchain networks

Market Risks

- Regulatory uncertainty in multiple jurisdictions
- Competition from existing and emerging protocols
- Market volatility affecting token value and adoption
- Dependency on broader Web3 ecosystem growth

Operational Risks

- Key personnel dependency and team scaling challenges
- Third-party service provider reliability

- Community adoption and developer ecosystem growth
- Potential technical standard changes affecting compatibility

Financial Risks

- Token price volatility and liquidity concerns
- Funding requirements for continued development
- Revenue model execution and monetization challenges
- Economic attack vectors on protocol incentives

Disclaimer

IMPORTANT NOTICE: This whitepaper is for informational purposes only and does not constitute investment advice, financial advice, trading advice, or any other sort of advice. Nothing in this whitepaper should be taken as a recommendation to purchase, sell, or hold any token, digital asset, or investment.

MONO tokens may be subject to regulatory restrictions in certain jurisdictions. Potential purchasers should consult with legal advisors regarding the regulatory status of MONO tokens in their jurisdiction before participating in any token sale or distribution.

This whitepaper contains forward-looking statements based on current expectations and assumptions. Actual results may differ materially from those projected. The development roadmap, technical specifications, and business plans described herein are subject to change without notice.

Blockchain technology and smart contracts are relatively new and experimental. The protocol may contain bugs, vulnerabilities, or design flaws that could result in loss of funds or failure to operate as intended.

No representations or warranties are made regarding the future performance, adoption, or success of Mono Protocol. The protocol is under development and may never achieve the goals outlined in this whitepaper.

Purchasing MONO tokens involves substantial risk of loss. Tokens may have no value and purchasers may lose their entire investment. Only participate with funds you can afford to lose entirely.

By accessing this whitepaper, you acknowledge that you have read, understood, and accepted the risks outlined above and agree to conduct your own research and due diligence before making any investment decisions.