Rhinestone Warp: A Technical Whitepaper on the Multi-Chain Intent Execution Engine

1.0 Introduction

The multi-chain ecosystem, while powerful, has precipitated a user experience crisis. Users and developers alike are confronted with a fragmented landscape of disparate chains, wallets, gas tokens, and bridges. This complexity forces users to manually navigate a labyrinth of low-level infrastructure, becoming a significant barrier to mass adoption. Developers, in turn, must wrestle with integrating a perpetually expanding set of protocols, diverting resources from creating truly differentiated products.

Rhinestone Warp is the sophisticated engineering solution to this crisis. It is a blazing-fast intent routing and execution engine designed to make multi-chain complexity invisible. By leveraging a novel architecture built on foundational standards, Warp orchestrates complex, multi-leg execution paths into a single, atomic user intent with unparalleled speed and reliability.

This whitepaper provides the definitive technical explanation of the Warp architecture. It details the system’s foundational primitives, unpacks the intent-based paradigm it employs, and dissects the core mechanisms that enable fast, atomic, and deterministic cross-chain execution. The following sections will illuminate how Warp achieves its performance through the powerful synergy between modular smart accounts and a competitive solver market.

2.0 Foundational Primitives: Account Abstraction and Modularity

To understand Warp's advanced execution model, one must first understand the two core blockchain standards it leverages: ERC-4337 (Account Abstraction) and ERC-7579 (Modular Smart Accounts). These are not merely complementary technologies; they are the essential on-chain building blocks that provide the necessary programmability and security guarantees for Warp's high-speed, intent-based system.

2.1 ERC-4337: Programmable Accounts

Account Abstraction, as standardized by ERC-4337, is the mechanism that transforms a user's wallet from a simple, externally owned account (EOA) into a fully programmable smart contract. This crucial evolution allows for the implementation of advanced logic and custom features

directly at the account level. Under this standard, transactions are initiated as `UserOperation` objects, which instruct the smart account on how to execute a more complex set of actions, moving beyond simple value transfers.

2.2 ERC-7579: Modular Smart Accounts

ERC-7579, a standard co-authored by Rhinestone, extends the concept of programmable accounts by making them modular. It establishes an open platform architecture where new features, packaged as "modules," can be securely installed and uninstalled from a user's account. This modularity provides the necessary foundation for a truly abstracted user experience.

There are three primary types of modules that can be installed on an ERC-7579 compliant account:

- **Validators:** These modules manage the account's authentication and signature validation logic. A *Passkey Validator*, for example, would allow a user to authorize transactions using their device's biometrics instead of a traditional private key.
- **Executors:** These modules contain custom logic for executing complex actions. A *Social Recovery Executor* could define the rules and trusted parties who are permitted to help a user recover a lost account.
- **Hooks:** These modules are executed before or after a transaction to enforce specific rules or logic. A *Spending Limit Hook* could be used to prevent transactions that exceed a predefined daily limit, adding a layer of security.

This modular architecture is the critical prerequisite for the powerful security model that underpins the Warp engine.

3.0 The Intent-Based Paradigm

The strategic shift from imperative transactions to declarative intents is a cornerstone of the Warp architecture. This paradigm abstracts execution complexity away from the end-user and delegates it to a specialized set of professional actors who are incentivized to find the most efficient path. This allows users to focus on their ultimate goal rather than the intricate steps required to achieve it.

3.1 From Imperative Transactions to Declarative Intents

Traditional blockchain interactions are imperative—the user must specify not only what they want to achieve but also the exact sequence of steps on how to achieve it. Intents introduce a declarative model, where the user simply states their desired final outcome.

Imperative Transaction	Declarative Intent
The user is responsible for defining the *how*. They must manually execute a multi-step process: approve a token, swap it on a DEX, find and use a bridge, and wait for finality.	The user is responsible for defining the *what*. They sign a single, legible message that declares their goal, abstracting away the steps.
Example: A user must execute a series of transactions to move 1500 USDC from Polygon to get 1 ETH on Arbitrum.	**Example:** A user signs a single intent: "Goal: 1 ETH on Arbitrum, starting with 1500 USDC on Polygon."

3.2 The Solver Market

Once a user signs a declarative intent, the message is broadcast to a competitive marketplace of professional actors known as "solvers." This unified marketplace, referred to as the **Relayer Market** within the Warp system, consists of sophisticated off-chain agents that compete to find the most efficient and cost-effective execution path to fulfill the user's declared goal.

This model introduces a critical security problem: for a solver to optimistically provide assets on a destination chain, they require a guaranteed repayment, necessitating a new architectural primitive that can only be realized through modularity.

4.0 The Architectural Lynchpin: Resource Locks via Modularity

The core technical thesis of the Warp engine is that its high-speed, optimistic execution model is fundamentally dependent on the unique capabilities afforded by ERC-7579 modular smart accounts. This technical dependency is the central innovation of the system, enabling a level of performance and security that would otherwise be impossible.

4.1 The Solver's Guarantee

The primary challenge for any solver in an intent-based system is managing risk. Before optimistically fronting their own assets to a user on a destination chain, a solver requires an ironclad, cryptographic guarantee that the user's funds on the origin chain are reserved exclusively for them and cannot be double-spent. Without this on-chain guarantee, the optimistic model collapses.

4.2 The "Resource Lock Hook" Module

This guarantee is achieved through a mechanism called a Resource Lock, which is implemented in the Warp architecture as a specialized ERC-7579 module: the **Resource Lock Hook**. When a user signs a Warp intent, this pre-installed hook activates to create an irrevocable on-chain commitment of their funds. The Resource Lock Hook doesn't bounce off the account; it "clicks in" to a designated module slot, becoming an integrated part of the account's security logic. Standard EOAs or non-modular smart accounts cannot support this logic, as they lack the programmability to enforce such a conditional lock. Modularity is therefore a strict technical requirement for the system to function securely.

4.3 The Technical Dependency Chain

The entire Warp system is built upon a clear and sequential chain of technical dependencies, where the high-level user experience is made possible only by the foundational primitives at the lowest level.

1. The **Warp System**, which provides an instant, intent-based UX, requires...
2. **Resource Locks** to enable solvers to perform optimistic fills, which are enforced by...
3. The **Resource Lock Hook**, which is a specialized, installable...
4. **Module**, that can only be installed on a...
5. **Modular Smart Account (ERC-7579)**.

This powerful synergy between the intent-based paradigm and the modular account architecture is the central innovation that enables the performance and security guarantees of the Rhinestone Warp engine.

5.0 The Warp Engine: Architecture and Execution Flow

Warp is the cohesive system that orchestrates the previously discussed concepts into a high-performance engine. It comprises a set of off-chain and on-chain components working in concert to manage the entire lifecycle of an intent. This section details the core architectural components and traces the step-by-step execution flow from creation to settlement.

5.1 System Architecture

The Warp engine is composed of three primary components that manage intent routing, solving, and execution:

- The off-chain **Orchestrator**, which functions as a sophisticated pathfinder. It receives intents, indexes user balances, finds the optimal cross-chain path, and builds the transaction data. It breaks intents down into discrete **"chain-token elements"** and assigns each to the most efficient settlement layer to ensure the best price and speed.

- The on-chain **Routing Engine**, which consists of smart contracts that allow solvers to interact directly with underlying settlement layers.
- The unified **Relayer Market**, a competitive marketplace where solvers receive execution plans from the Orchestrator and compete to fulfill them.

5.2 The Intent Lifecycle: From Signature to Settlement

The end-to-end execution flow of a Warp intent is a carefully choreographed sequence designed to maximize speed for the user while ensuring security for the solver.

1. **Intent Creation & Routing:** A user signs a single, legible EIP-712 intent declaring their goal. The Orchestrator receives this intent, breaks it down into "chain-token elements," finds the optimal execution path, and propagates it to the Relayer Market.
2. **On-Chain Commitment:** On the origin chain, the installed "Resource Lock Hook" on the user's modular smart account activates, creating an irrevocable on-chain commitment of their funds for the solver.
3. **Optimistic Fill:** A solver in the Relayer Market verifies the on-chain lock. Confident in this guarantee, the solver immediately sends the required assets from its own liquidity pool to the user on the destination chain.
4. **User Finality:** This "optimistic fill" achieves finality for the user with remarkable speed. Average pre-confirmation time is just **500ms**. Final transaction inclusion on Ethereum L2s averages **under 1.5 seconds**, even when Ethereum L1 is used as an origin chain.
5. **Background Settlement:** After the origin chain transaction finalizes, the solver is automatically reimbursed by claiming the user's locked assets through the underlying settlement layer. This process is invisible to the user.

5.3 The Compact: The Resource Locking Primitive

The resource locking method that enables this high-speed flow is formally known as **The Compact**. This primitive, developed by **Uniswap Labs** in collaboration with **Rhinestone** and **LI.FI**, provides the atomic and deterministic properties of Warp's execution, guaranteeing that every intent is either fulfilled completely or not at all.

This tightly integrated architecture enables a suite of powerful capabilities for developers.

6.0 Core Features and Developer Capabilities

The Warp architecture translates into a powerful set of tangible features that redefine the developer and end-user experience for multi-chain applications.

6.1 Unparalleled Performance and Reliability

- **Speed:** With average pre-confirmation times of **500ms** and L2 transaction inclusion in **under 1.5 seconds**, Warp enables user experiences that move at web2 speed.

- **Deterministic Execution:** Because the relayer supplies the output token, the outcome is guaranteed. This model protects users from **MEV**, eliminates **price slippage**, and ensures the exact amount quoted is the amount received.

6.2 Advanced Transaction Capabilities

- **Fee Abstraction:** Warp is a **drop-in replacement for ERC-4337**. Relayers act as a **bundler-paymaster**, putting the transaction on-chain for the user and funding the gas. This enables developers to offer gas payments in any ERC-20 token or to fully subsidize fees for a gasless experience.
- **Single-Click Multi-Chain Operations:** The system supports arbitrary origin and destination chain operations within a single intent. A critical benefit is the preservation of `msg.sender` on the destination chain, which is a **huge quality of life improvement** for teams that are stringing together crosschain operations against DeFi protocols that rely on this property.
- **Legible Multi-Chain Signatures:** Warp utilizes a purpose-built, legible **EIP-712** signature structure. This avoids forcing users to sign opaque hashes, instead presenting a human-readable representation of every authorized action.
- **Transaction Automation:** A custom session key implementation, fully compatible with **The Compact**, enables developers to build sophisticated multi-chain automations without compromising user self-custody.

Warp's modular design ensures it integrates smoothly into the broader blockchain ecosystem.

7.0 Ecosystem Integration and Composability

Warp is designed with modularity at its core, enabling it to integrate with a wide range of protocols, settlement layers, and account standards to ensure maximum flexibility for developers.

7.1 Aggregated Settlement Layers

Warp aggregates multiple settlement layers via its unified Relayer Market, routing each part of an intent to the most efficient provider. The layers integrated at launch each provide unique value:

- **Across Protocol:** A trustless, permissionless settlement layer **secured by UMA Protocol**. This integration allows Warp to **inherit its trustless execution properties**.
- **Relay Protocol:** Provides extensive chain and token coverage with deep liquidity, enabling **rapid rollout to over 70 chains**.
- **Eco:** Offers **best-in-class stablecoin flows** and is **powered by Hyperlane**, ensuring access to fast and cheap stablecoin liquidity.

7.2 Account Compatibility

To ensure the broadest possible reach, Warp is designed to be a unified engine for all types of on-chain accounts. It offers first-class features for **ERC-7579** accounts, is compatible with **EIP-7702** for EOA upgrades, and supports **vanilla EOAs** through a standard "connect-approve-warp" flow.

This broad compatibility makes Warp a universal engine capable of serving the full spectrum of on-chain accounts and interoperability needs.

8.0 Conclusion

The Rhinestone Warp engine represents a significant step forward in solving the multi-chain user experience crisis. Its primary innovation is the synergistic coupling of modular smart accounts (ERC-7579) with on-chain Resource Locks, which provide the cryptographic guarantee necessary to empower a competitive solver market. This architecture, in turn, delivers an instant, atomic, and deterministic intent-based experience to the user.

Warp is a fully integrated system that makes the immense complexity of chains, bridges, and gas tokens invisible. It provides developers with a single, unified SDK and API to build powerful cross-chain applications, freeing them from the burden of managing low-level infrastructure. By abstracting complexity and prioritizing the user experience, Warp enables a new generation of applications built for a truly interoperable world.

Stop building for chains. Start building for users.