



OST

Eastern Switzerland
University of Applied Sciences

Blockchain (BlCh)

Account Abstraction

Thomas Bocek

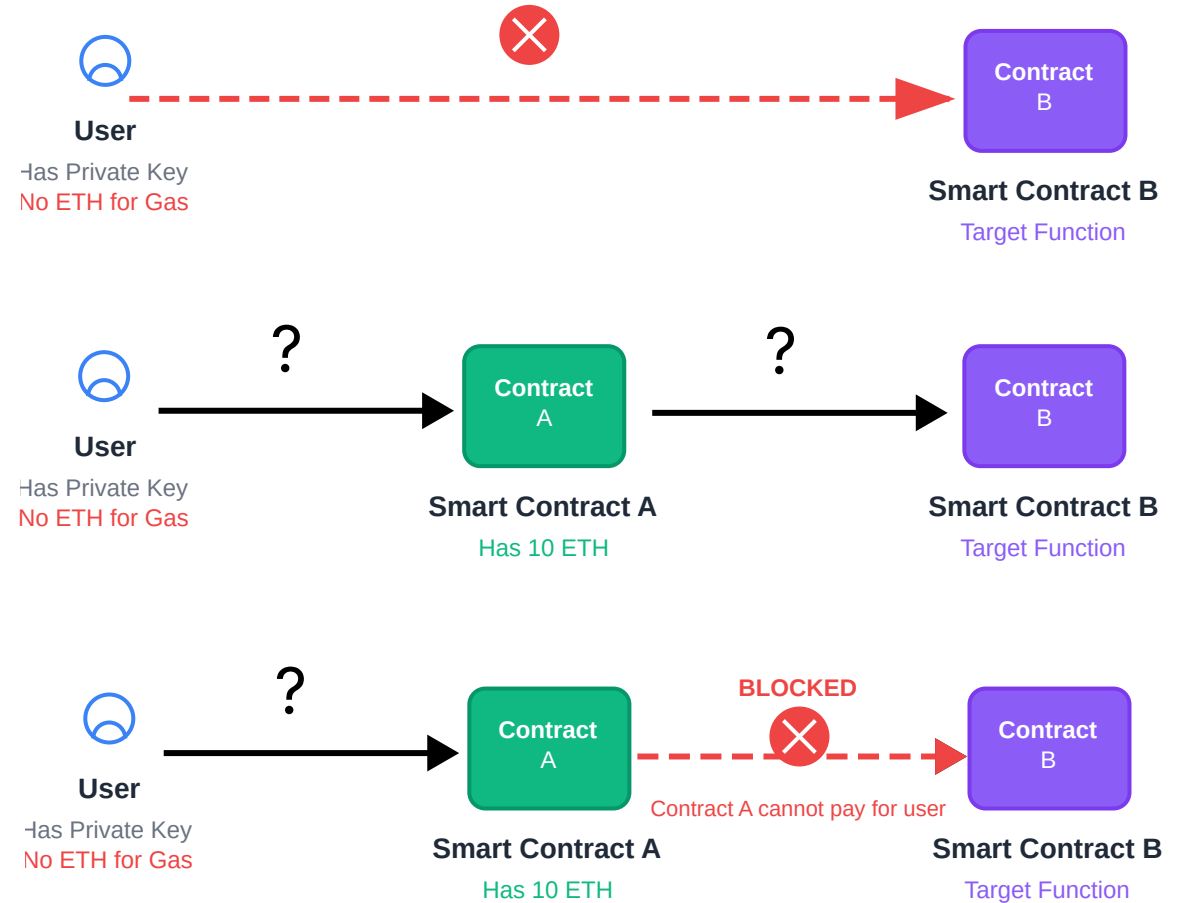
27.09.2025

Learning Goals

- Lecture 3
 - Account Abstraction
 - Single-sentence summary: Allow smart contract wallets (smart accounts) to act as first-class user accounts, unlocking UX, recovery, and new signature schemes
 - Learning goals:
 - Understand components
 - UserOperation
 - Bundler
 - EntryPoint
 - Paymaster
 - Current ecosystem adoption
 - Tradeoffs
 - Next steps

Motivation: UX & security problems with EOAs

- EOAs: user manages private keys and native ETH for gas
 - Bad UX for mainstream users
- Common issues for adoption
 - lost seed phrases
 - confusing gas payment, poor onboarding
- With account abstraction
 - Social login, account recovery
 - Gas sponsorship
 - Session keys
 - Batched flows

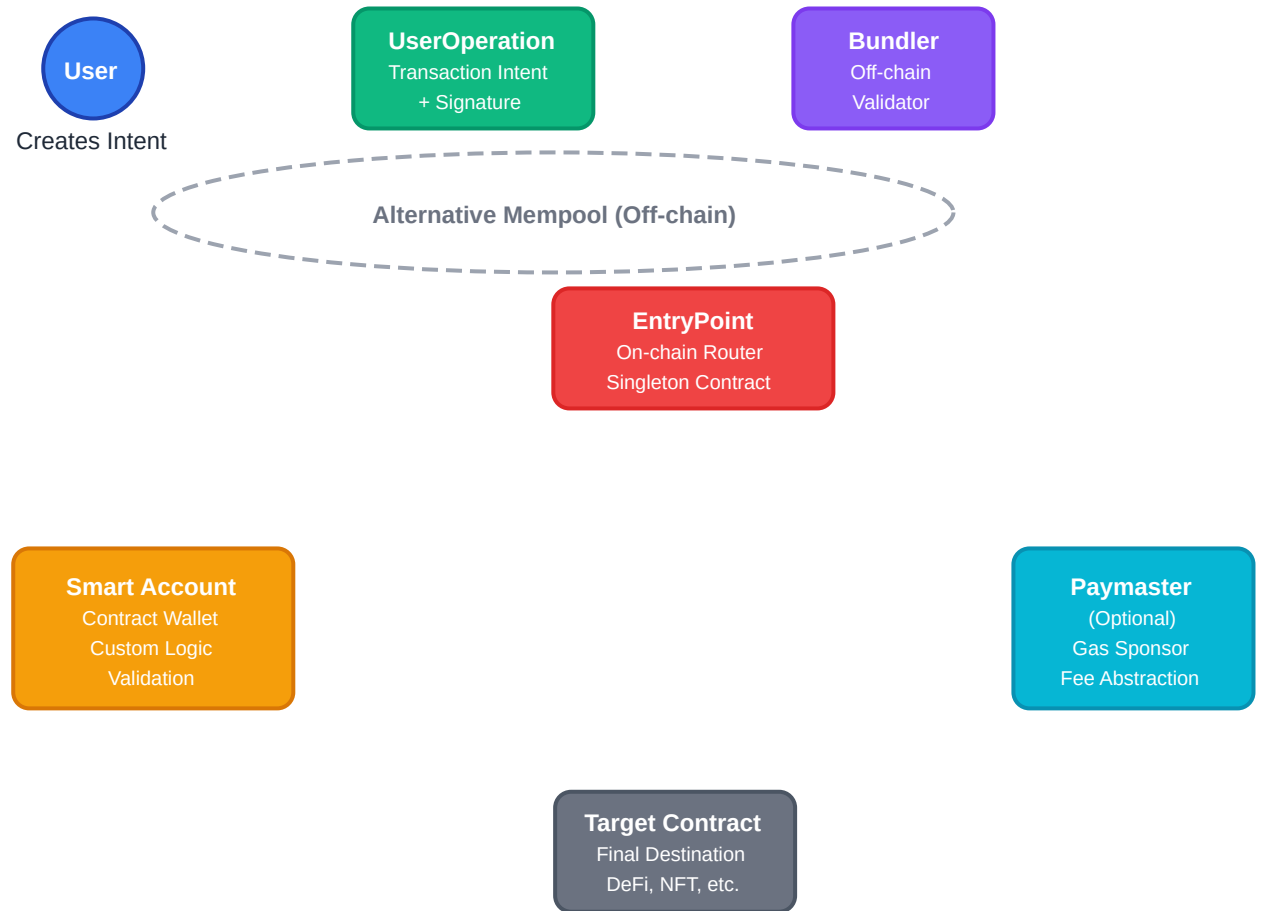


Fusaka Upgrade & EOF Removal

- Fusaka upgrade scheduled: 03.12.2025
 - Originally planned: Nov 2025
 - Focus on PeerDAS and scalability improvements
- Ethereum Object Format (EOF) officially removed from Fusaka
 - Decision made during April 2025
 - Technical uncertainties and community opposition cited
 - EOF was intended as architectural foundation for native AA
- Native Account Abstraction delayed indefinitely
 - EIP-7701 (Native AA) depends on EOF infrastructure
 - Current focus on [ERC-4337](#) implementation
 - Potential inclusion in future Glamsterdam upgrade
- Community consensus challenges
 - Over 100 participants in final decision call
 - Lack of [agreement](#) on EOF variants (Options A-D)
 - Priority given to PeerDAS for scaling improvements
- Impact on AA roadmap
 - ERC-4337 remains primary AA solution for 2025-2026
 - Native protocol integration pushed to post-2026
 - Reliance on external infrastructure: bundlers, paymasters

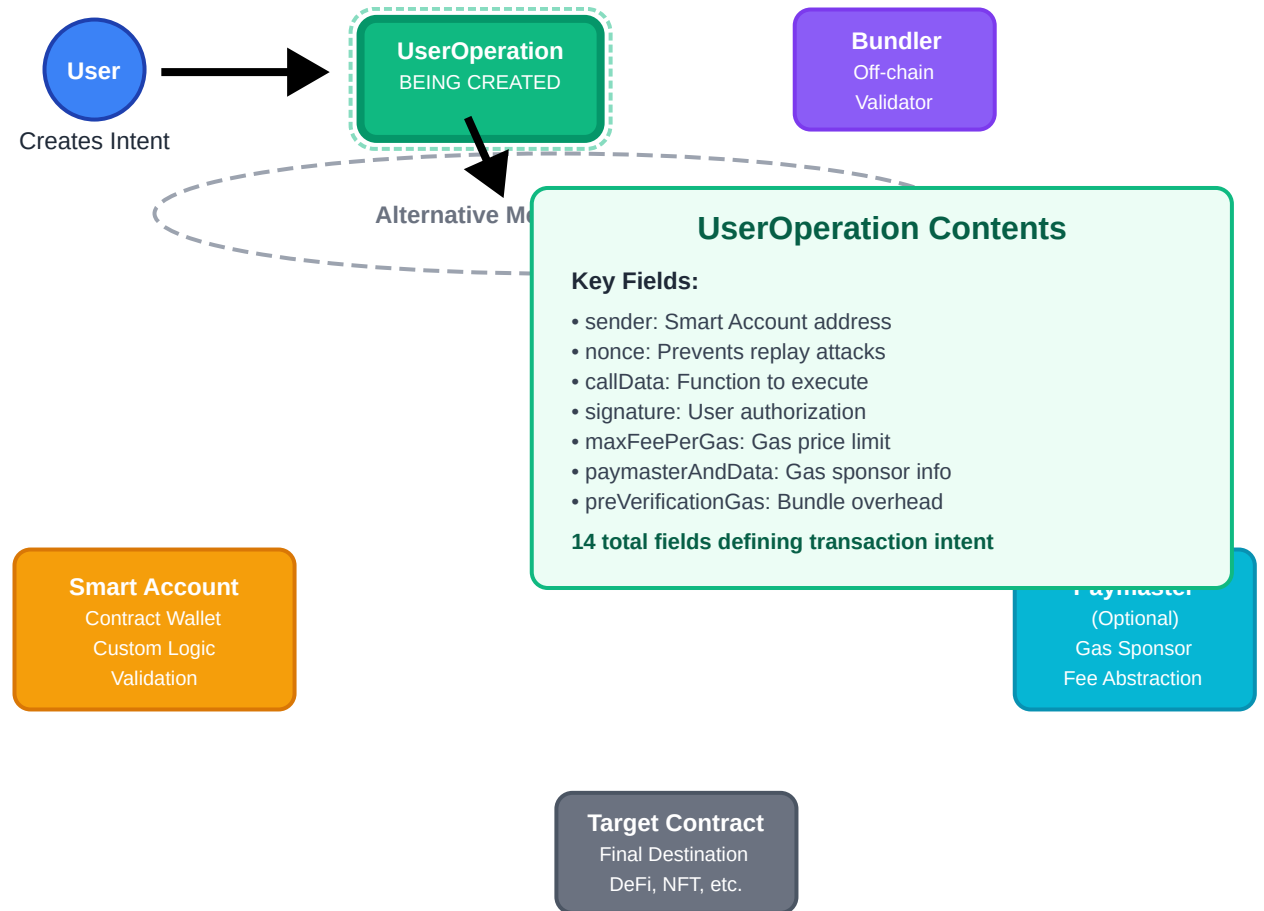
ERC-4337 (1)

- ERC-4337: introduce UserOperation objects + an alternative (off-chain) mempool instead of changing consensus rules.
- Components
 - Smart Account (contract wallet)
 - UserOperation
 - Bundler
 - EntryPoint (on-chain router/validator)
 - Paymaster (optional gas sponsor)



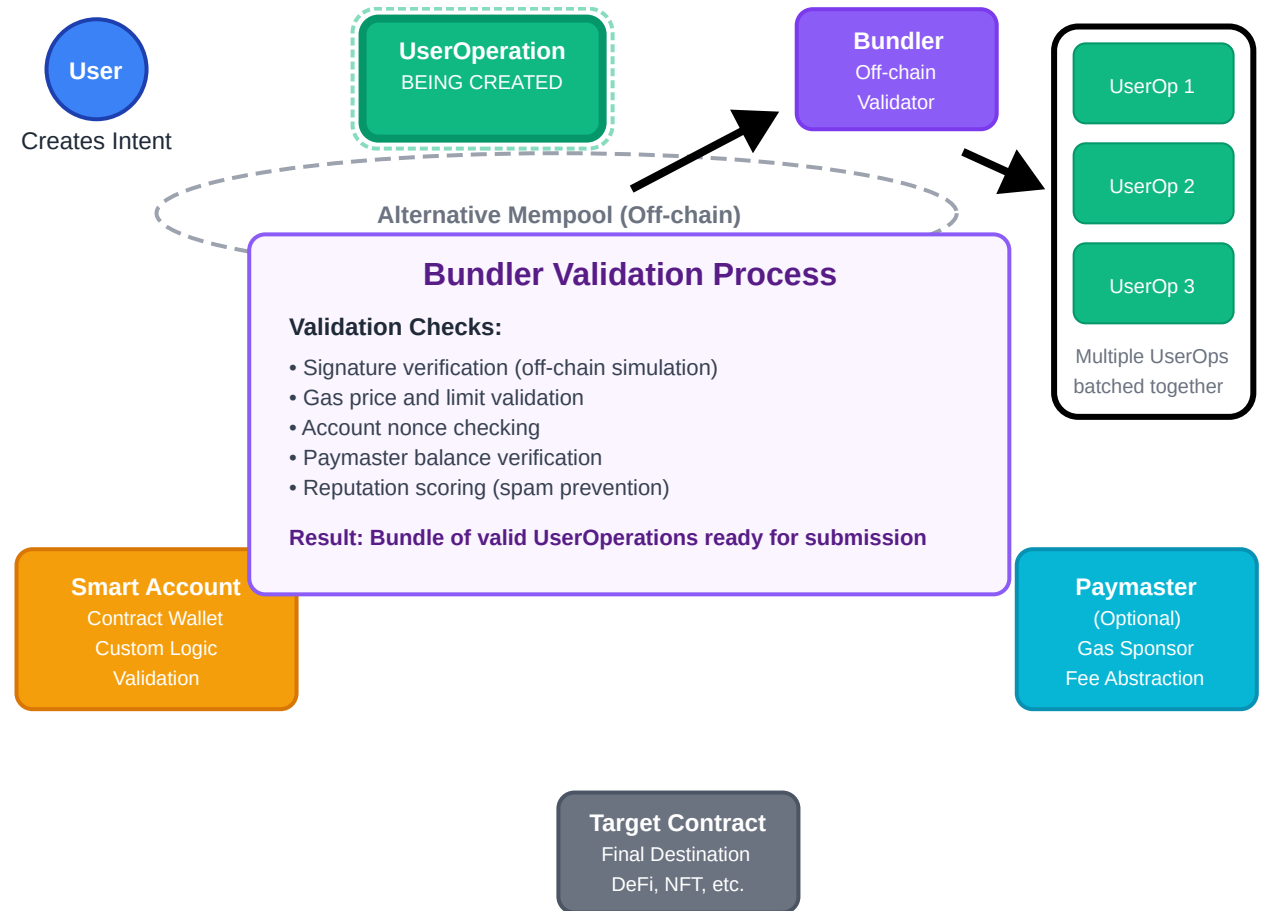
ERC-4337 (2)

- Step 1: User Creates UserOperation
 - User wallet creates a UserOperation object containing transaction intent
 - Similar to a regular transaction but with 14 specialized AA fields:
 - sender, nonce, callData, signature (standard fields)
 - paymasterAndData, maxFeePerGas, preVerificationGas (AA-specific)
 - initCode, verificationGasLimit, callGasLimit (execution parameters)
 - Contains all necessary information for validation and execution
 - Signed by user but not yet submitted to Ethereum
 - Ready for bundler from alternative mempool



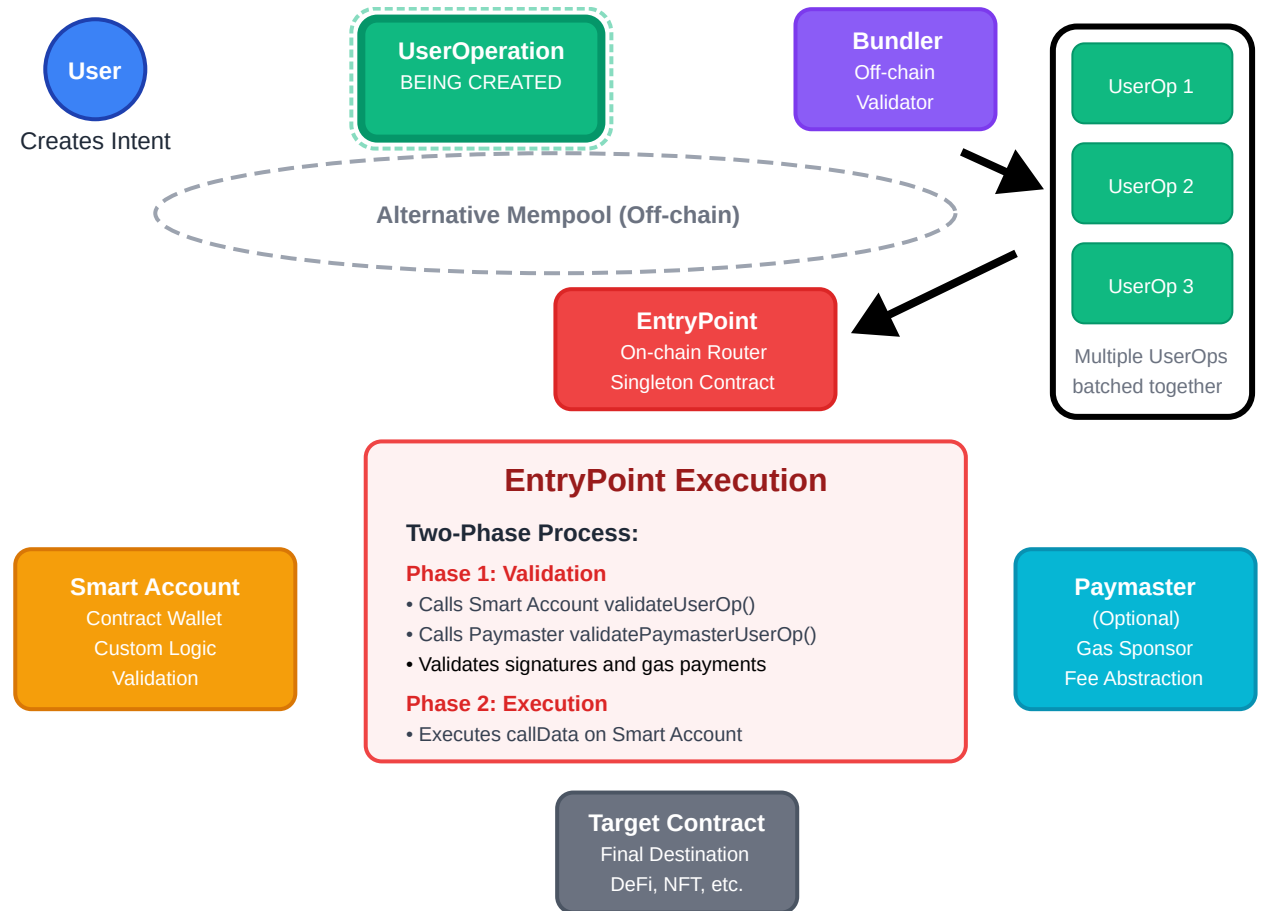
ERC-4337 (3)

- Step 2: Bundler Validates and Bundles UserOperations
 - Bundler collects multiple UserOperations from alternative mempool
 - Performs off-chain validation checks:
 - Signature verification through simulation
 - Gas price and limit validation
 - Account nonce checking for replay protection
 - Paymaster / smart account balance verification
 - Reputation scoring for spam prevention
 - Creates optimized bundle to minimize gas costs
 - Prepares bundle for submission to EntryPoint contract



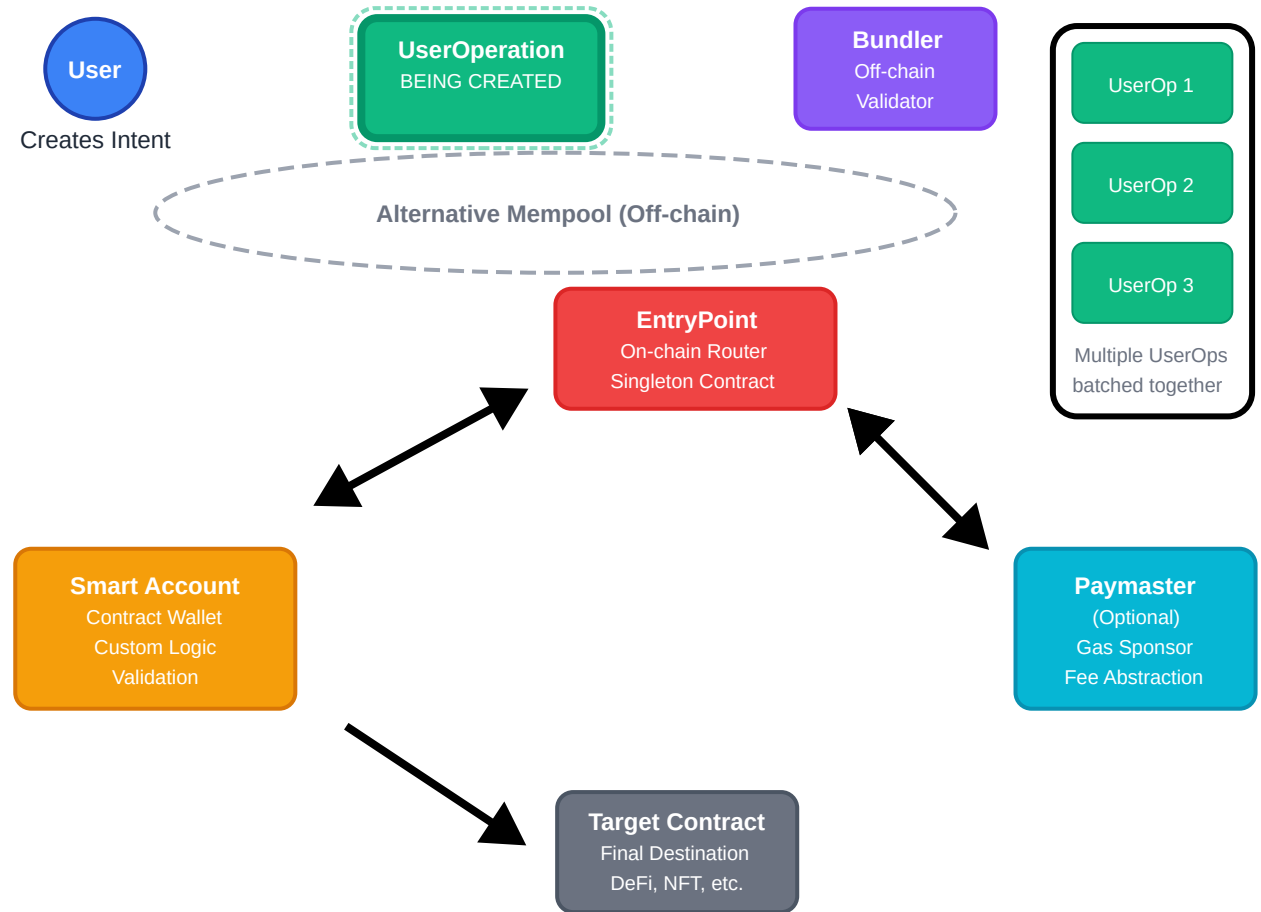
ERC-4337 (4)

- Step 3: EntryPoint Executes Bundle
 - EntryPoint receives bundle as regular Ethereum transaction from bundler
 - Processes each UserOperation in atomic two-phase execution:
 - Validation Phase: Calls validateUserOp() on Smart Account (state-modifying)
 - Execution Phase: Calls execute() with callData (must follow validation)
 - Acts as single on-chain coordinator for all Account Abstraction operations
 - Manages gas accounting and refunds bundler immediately
 - If any UserOperation fails, entire bundle transaction reverts



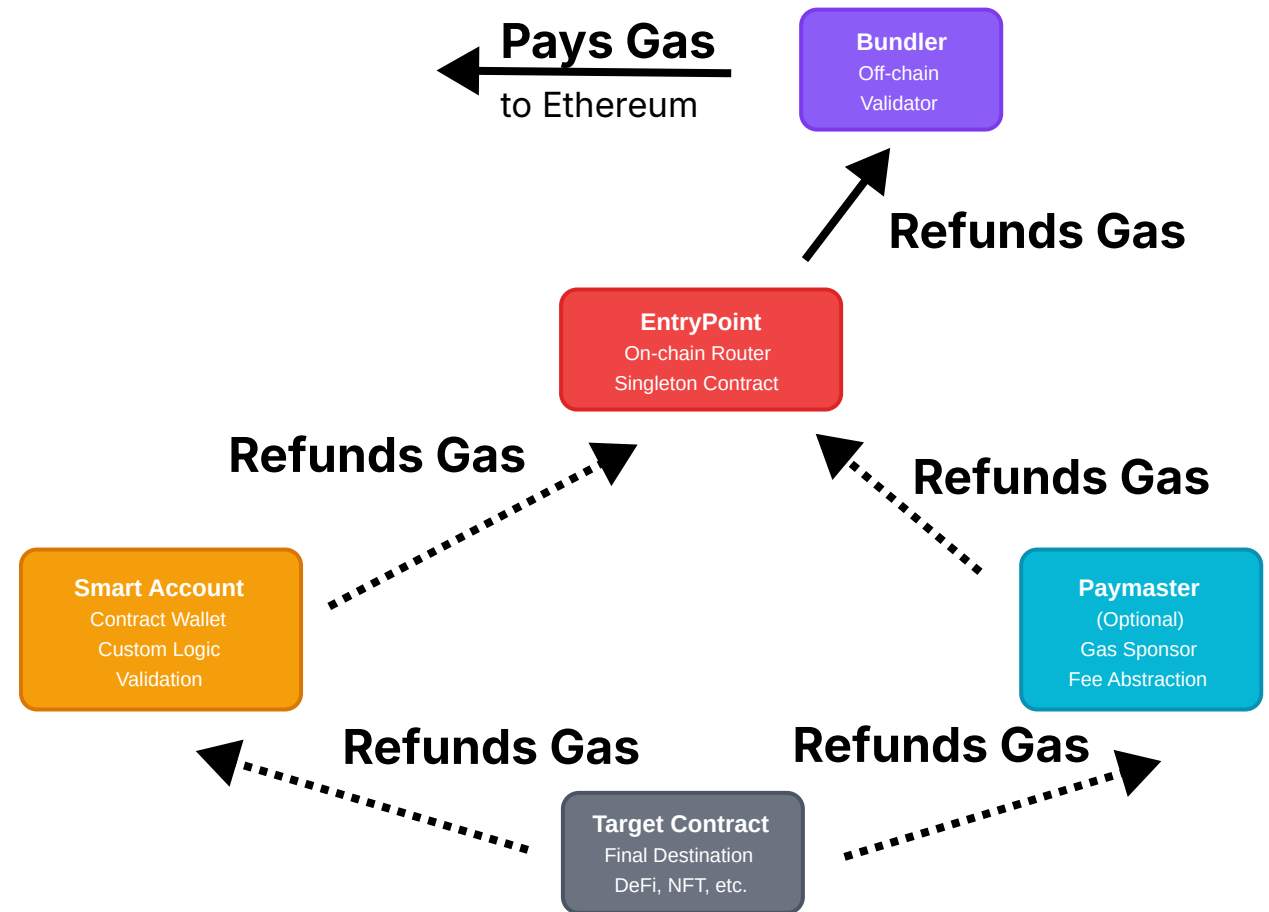
ERC-4337 (4)

- Step 4: Smart Account Validation & Execution
 - EntryPoint calls validateUserOp() on Smart Account contract
 - Smart Account executes custom validation logic:
 - Signature verification (ECDSA, multisig, biometric, etc.)
 - Nonce validation for replay protection
 - Custom authorization rules (time limits, spending caps)
 - Session key validation for pre-approved operations
 - If Paymaster present, EntryPoint validates gas sponsorship and deposit balance
 - Validation can modify state (increment nonce, update counters)
 - If validation succeeds, EntryPoint calls execute() on Smart Account
 - Smart Account forwards call to target contract (DeFi, NFT, etc.)



ERC-4337 (5)

- ETH Flow
 - Bundler pays gets, gets refunded by the EntryPoint
 - Only executed if Bundler can be refunded
 - Refunding very flexible
 - Paymaster (typical case) refunds
 - Smart Account (typical case) refunds
 - Or via target contract
 - Refunds via Smart Account
 - Refunds via Paymaster
 - can be implemented theoretically...
- Native Account Abstraction could make this a lot easier...



Core UX Features Unlocked by Account Abstraction

- Gasless Transactions / Fee Sponsorship
 - Paymasters enable apps or protocols to sponsor user gas fees
 - Users can pay in ERC-20 tokens instead of ETH
 - Fiat on-ramps possible without ETH requirements
- Social Recovery & Key Management
 - Guardian-based account recovery (friends, family, institutions)
 - Multisig validation with customizable thresholds
 - Eliminates catastrophic seed phrase loss scenarios
- Session Keys / Limited Delegation
 - Pre-approved operations within defined parameters
 - Gaming: auto-approve in-game transactions up to X tokens
 - DeFi: automated trading within risk limits
- Batched Transactions / Meta-Transactions
 - Single-click multi-step workflows (approve + swap + stake)
 - Atomic operations that succeed or fail together
 - Improved efficiency and user experience

Who Builds the Infrastructure? (Ecosystem Roles)

- Wallet Teams: Smart Account Implementations
 - [SimpleAccount](#) (reference implementation), [Safe](#) (Gnosis Safe variants)
 - [Argent](#), [Soul Wallet](#), [Biconomy](#), [ZeroDev](#) Kernel
 - Custom implementations with specialized validation logic
- Bundlers: Permissionless Transaction Aggregators
 - Mainnet: [Stackup](#), [Alchemy](#), [Pimlico](#), [Etherspot](#), [Skandha](#)
 - Testnets: Stackup testnet bundlers, Alchemy dev infrastructure
 - Anyone can run a bundler - competitive marketplace for efficiency
- Paymasters: Gas Sponsorship Providers
 - DApps sponsoring their users' transactions
 - Token projects enabling ERC-20 gas payments
 - Enterprise paymasters for B2B blockchain applications
- Tooling & Infrastructure Providers
 - SDKs: [Alchemy AA SDK](#), [Biconomy SDK](#), [ZeroDev SDK](#)
 - Node providers: Alchemy, [Infura](#) with AA support
 - Testing frameworks: [Hardhat](#) plugins, [Foundry](#) integrations

Current Adoption & Ecosystem Status

- ERC-4337 Production Ready: Live on Ethereum mainnet since March 2023
 - Growing Infrastructure: 26+ million smart accounts deployed, 170+ million UserOperations processed
 - Wallet Ecosystem: Safe (41.6M accounts), Biconomy (3.5M users), ZeroDev Kernel (4M accounts), Argent on StarkNet
 - Bundler Market: Stackup (dominant on Ethereum mainnet), Pimlico (leading overall bundler), Alchemy Bundler, Etherspot Skandha
 - Real-world Usage: 99.2% of UserOperations use paymasters, \$430K+ in gas sponsorships
 - Developer Tools: Mature SDKs from Alchemy, Biconomy, ZeroDev; comprehensive testing frameworks
- Challenges & Open Problems
 - Performance Overhead: 10-15% gas cost increase vs EOA for individual UserOperations; bundling multiple operations can reduce costs through amortized EntryPoint overhead
 - Infrastructure Maturity: Bundler reputation systems and mempool standardization still evolving
 - Centralization Concerns: Market concentration in bundlers/paymasters creates potential bottlenecks
 - Security Complexity: Smart contract wallets expand attack surface requiring extensive audits
 - User Experience: 1.5x-11.7x slower transaction propagation compared to EOA flows
 - Standards Fragmentation: Multiple wallet implementations create integration complexity for dApps

Demo / Next steps / Resources

- Next week...