Discover the Key Concepts of Account Abstraction and Why ERC-4337 is Important

**Last Updated:**

June 1, 2023

Account Abstraction as defined by **[ERC-4337](https://eips.ethereum.org/EIPS/eip-4337" \t "_blank)**, "allows users to use smart contract wallets containing arbitrary verification logic instead of EOAs as their primary account." [**ERC-4337 introduces many user experience benefits**](https://www.alchemy.com/overviews/smart-contract-wallet-benefits), most notably enabling people to use Smart Contracts as their primary accounts.

ERC-4337 runs on top of the blockchain and does not require any changes to the blockchain itself. This makes it usable today, on Ethereum or any EVM chain, without any significant changes to the underlying blockchain infrastructure.

On March 1st, 2023, Yoav Weiss from the Ethereum Foundation, announced that ERC-4337 was live on Ethereum mainnet when the EntryPoint contract was first deployed.

**What are the key concepts of Account Abstraction?**

The key concepts of Account Abstraction are UserOperation, Bundler, Sender, EntryPoint, Paymaster, and Aggregator. In combination, these concepts work together to enable web3 developers to build [**smart contract wallets**](https://www.alchemy.com/best/smart-contract-wallets) and [**make dapps compatible with SCWs**](https://docs.alchemy.com/docs/how-to-make-your-dapp-compatible-with-smart-contract-wallets).

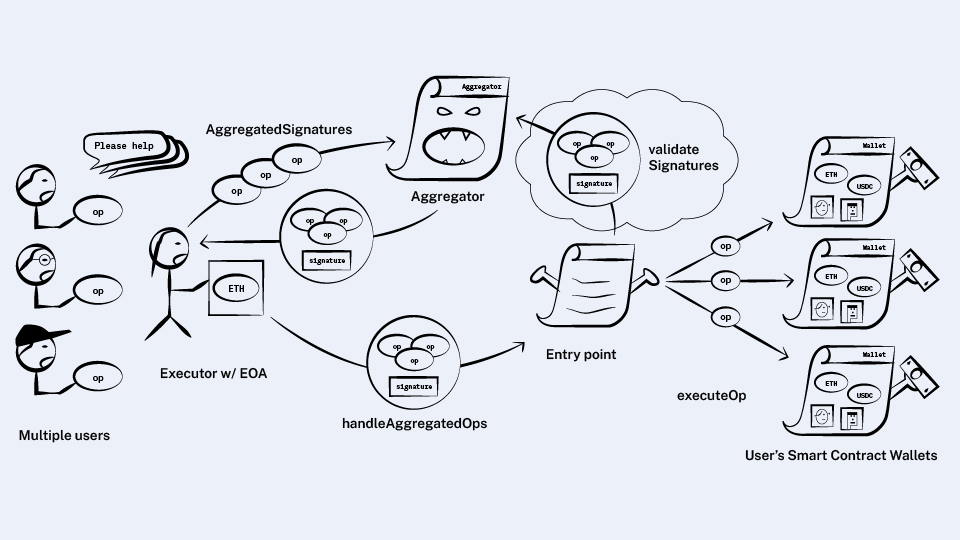


Diagram showing how users define UserOperations which are bundled, sent to the EntryPoint contract, and then executed on behalf of user's Smart Contract Wallets.

**1. UserOperation**

A [**user operation**](http://www.alchemy.com/overviews/user-operations) is a “pseudo-transaction object” representing a user’s transaction intent. Any user operation can contain multiple instructions and additional data to execute smart contract calls initiated by the Smart Contract Account. UserOperations begin the 4337 transaction flow.

**What is the difference between a UserOperation and traditional transaction?**

Three key differences between a UserOperation and regular transaction are:

* **Additional fields** - UserOperations include new fields in the transaction structure (e.g., EntryPoint, Bundler and Aggregator)
* **Alternate mempool** - UserOperations are sent to a separate mempool, where bundlers can package them into transactions which get included in a block
* **Authentication** - for a transaction, authentication is always done via a signature from a single private key that can never change for a given sender. In a user op, authentication is programmable.**‍**

**2. Bundler**

A bundler monitors an alternative mempool specifically built for user operations. The bundler bundles multiple user operations into a single transaction and submits that transaction to the entry point contract. Bundlers are compensated for doing this by taking a portion of the gas fees.

Bundlers are a critical piece of [**infrastructure to actualize ERC-4337**](https://www.alchemy.com/overviews/what-is-a-bundler) because all Ethereum transactions need to be initiated by an Externally Owned Account (EOA). Bundlers have EOAs, and in an account abstracted ecosystem they are the only participants that need EOAs. One of the main goals of ERC-4337 is to abstract away the need for everyone in web3 to have their own EOA wallet.

**3. EntryPoint**

The EntryPoint is a singleton smart contract that receives transactions from Bundlers, then verifies and executes UserOperations.

**How does the EntryPoint verification process work?**

The smart contract account gets to define its own verification and therefore its own authentication. During the verification process the EntryPoint contract checks to see if the wallet has enough funds to pay the maximum amount of gas it might possibly use, which is based on the gas fields in the UserOperation. If the wallet does not have enough funds, the EntryPoint contract rejects the transaction.

**How does the EntryPoint execution process work?**

During the execution process the EntryPoint contract executes the user operation by calling the account using the **calldata** specified in the user operation, and taking money from the Smart Contract Account to reimburse the Bundler with the right amount of ETH to pay for the gas.

**4. Paymaster**

The Paymaster is an ERC-4337 defined smart contract that handles the [**implementation of gas payment policies**](https://www.alchemy.com/overviews/what-is-a-paymaster). These gas policies create flexibility for how gas is paid (e.g. in what currency) and by whom, which removes the prerequisite for users to hold native blockchain tokens to interact with the blockchain.

For example, the native blockchain token for Ethereum is ETH and the native blockchain token for Polygon is MATIC. Therefore, instead of paying gas for Ethereum transactions in the native token (e.g. ETH), users can pay for gas fees with any ERC20 token like USD Coin (USDC) or Tether (USDT).

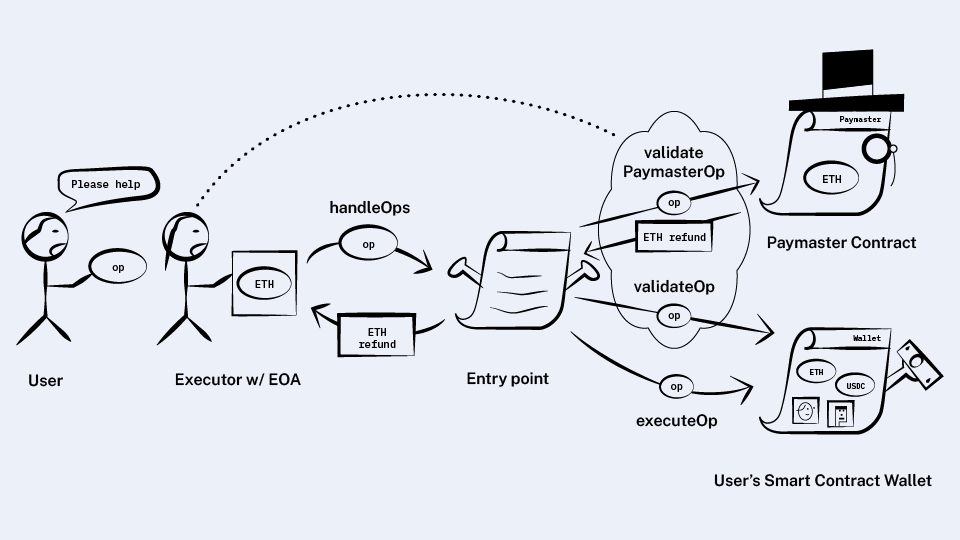


Diagram showing how the EntryPoint contract can communicate with the Paymaster contract to execute flexible gas policies.

**Paymasters allow application developers to:**

* Sponsor gas fees for their users
* Enable gas payments in stablecoins
* Enable gas payments in other ERC-20 tokens

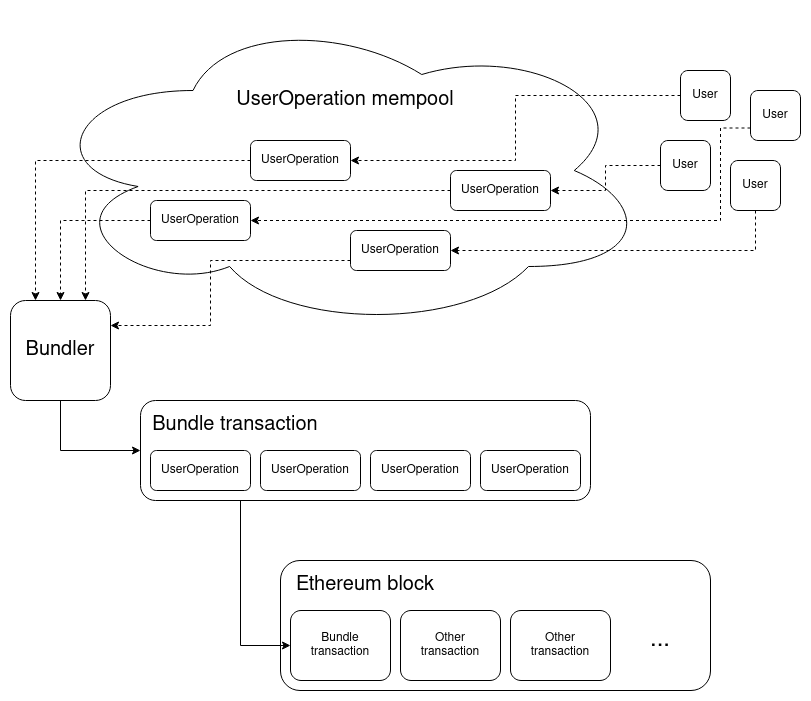
Alchemy’s [**Gas Manager API**](https://www.alchemy.com/account-abstraction) realizes the benefits of ERC-4337 defined Paymasters by completely abstracting away gas payments from users, while doing it in a robust way that lets the application decide what that abstraction should be (e.g. sponsored transactions, pay gas with stablecoins, etc).

**5. Aggregator**

An Aggregator is a smart contract that implements a signature scheme that supports aggregation (i.e. a contract that can verify aggregated signatures.

If multiple messages are signed with different keys, then a single combined signature can be generated that verifies the combined signature, and implies that all the constituent signatures are also valid.

By [**combining multiple signatures into a single signature**](https://www.alchemy.com/blog/account-abstraction-aggregate-signatures), aggregators help save on **calldata** costs, with multiple bundled UserOperations validated in a single step.

**[[](https://notes.ethereum.org/@vbuterin/account_abstraction_roadmap)](https://notes.ethereum.org/@vbuterin/account_abstraction_roadmap" \t "_blank)**

*Diagram showing how ERC-4337 uses an alternative mempool for UserOperations and a bundler to combine a bundle of UserOperations into a single transaction that is included in a block on Ethereum.*

**How did EIP-2938 and EIP-3074 inform the design of ERC-4337?**

ERC-4337 builds upon previous Ethereum Improvement Proposals (EIPs) 2938 and 3074. EIP-2938 introduced the idea for enabling smart contracts to operate as a "top-level account that pays fees and starts transaction execution," and EIP-3074 introduced the idea of delegating "control of the externally owned account (EOA) to a smart contract."

EIP-4337 used the ideas introduced by EIP-2938 and EIP-3074 and combined them with the idea for using an alternative mempool and an implementation route that did not require any change in the consensus layer.