**Intro to Account Abstraction**

Enabling the use of smart contract accounts over EOAs.

[Suggest Edits](https://docs.stackup.sh/edit/account-abstraction)

**Contract Accounts vs EOAs**

Account Abstraction is a blockchain technology that allows users to use smart contracts as their accounts.

The default account for most users is an Externally Owned Account, or EOA. EOAs are accounts controlled by an external private key. On most blockchain networks using the Ethereum Virtual Machine (EVM), only an EOA can trigger transactions so this is the default account model for most users.

Unfortunately EOAs require users to know a lot about how a blockchain works to use them safely. We can create much better user experiences using Contract Accounts.

**Why use contract accounts?**

If you've ever tried to build Web3 apps with amazing user experiences you'll eventually hit road blocks that are imposed by the limitations of an EOA. In other words, the UX ceiling in terms of what you can achieve with an EOA is not very high.

Here are some of the things contract accounts can enable that aren't possible with EOAs:

* **🔑 Arbitrary verification logic:** Support single and multi sig verification and any arbitrary signature scheme.
* **💱 Sponsored transactions:** Allow users to pay transaction fees in ERC-20 tokens or build your own fee logic, including sponsoring transaction fees on your app.  
  [block:tutorial-tile]{"backgroundColor":"#ffffff","emoji":"💱","id":"6529d9ab7d4e97005a9c165f","link":"[https://docs.stackup.sh/v0.6/recipes/use-a-paymaster","slug":"use-a-paymaster","title":"Use](https://docs.stackup.sh/recipes/use-a-paymaster%22,%22slug%22:%22use-a-paymaster%22,%22title%22:%22Use) a Paymaster"}[/block]
* **🔒 Account security:** Enable social recovery and security features like time-locks and withdraw limits.
* **⚛️ Atomic multi-operations:** Build flows that better align with your user's intent such as trading in one click rather than approving and swapping separately.  
  [block:tutorial-tile]{"backgroundColor":"#ffffff","emoji":"📨","id":"6529ce4f6b4d8300eab62c1d","link":"[https://docs.stackup.sh/v0.6/recipes/send-a-user-operation","slug":"send-a-user-operation","title":"Send](https://docs.stackup.sh/recipes/send-a-user-operation%22,%22slug%22:%22send-a-user-operation%22,%22title%22:%22Send) a User Operation"}[/block]

Account abstraction sounds great! But there are some down sides to also consider:

* **✍️ Signing issues:** Ideally, all apps would follow EIP-1271 to validate signatures. Unfortunately this is not always the case and those apps would be incompatible with contract accounts.
* **⛽️ Higher gas cost:** On L2s and other scaling solutions this becomes less of a problem. However more research on how to reduce gas cost in this context, especially on Ethereum mainnet, is required.

**The history of account abstraction**

There have been multiple approaches to account abstraction on Ethereum-based blockchains over the years such as [EIP-86](https://eips.ethereum.org/EIPS/eip-86) and [EIP-2938](https://eips.ethereum.org/EIPS/eip-2938). Unfortunately these weren't practical to achieve since they required consensus layer changes.

Luckily we can build account abstraction without consensus layer changes. A new standard, [ERC-4337](https://eips.ethereum.org/EIPS/eip-4337), is a new approach to account abstraction that relies on higher-level infrastructure that avoids the need for consensus layer changes.

This means, unlike past proposals, developers can start building on a universal standard today.

**The challenge for builders**

Although contract accounts offer the ability to build much better user experiences, they also introduce more complexity for developers to manage the interaction between different components.

That's where Stackup comes in.

📘

Did you know?

Stackup is a suite of open source tools for building applications with account abstraction.