Future visions for Ethereum have included smart contract wallets for some time. Not only do smart contract wallets improve efficiency and user experience, they provide a general way to mitigate cryptographic weaknesses (like ECDSA being vulnerable to quantum computing.)

The future of accounts is a wide open design space. We present a few rough options to migrate existing EOAs to smart contract wallets: forcibly migrate current EOAs, assume a default contract, a new transaction type, and a newly proposed opcode (AUTHUSURP

) plus EIP-3074.

For the purposes of this post, deploying bytecode may refer to actually deploying bytecode in the current sense, or setting a delegate/proxy field on the account in the verkle trie.

Approaches

Forced Deployment

What is it?

Perform an irregular state transition to deploy bytecode into every account that may have been an EOA.

Benefits

Irregular state transitions are a one-time cost, and this change could be performed alongside another state transition (like verkle trees.)

Drawbacks

The first major drawback to this approach is that if you're going to deploy bytecode, you need to have some bytecode to deploy. You'd need to implement, at minimum, a call function and some upgrade functionality.

This approach will also break any system of contracts that relies on SELFDESTRUCT

and CREATE2

, if the account is migrated between the SELFDESTRUCT

and the CREATE2

. There are, however, [plans to remove SELFDESTRUCT

](https://eips.ethereum.org/EIPS/eip-4758) so these contracts may break anyway.

Counterfactual contracts, even without SELFDESTRUCT

, would break as well.

Finally, this approach has a high cost to miners/validators, because every existing EOA has to be touched and modified.

Assume a Default Contract

What is it?

If a transaction originates from an account with no code, pretend that account had some default code which behaves like an EOA.

Benefits

Unlike actually modifying the state above, this approach does not have a one-time cost.

Since the bytecode isn't actually deployed anywhere, it's possible to upgrade it and add features over time.

Counterfactual contract deployments would not be entirely broken.

Drawbacks

While the default bytecode can be upgraded over time, you still need an implementation to execute, which may or may not do everything users need.

Create Transaction Type

What is it?

Introduce a new EIP-2718 transaction type that deploys code at the transaction signer's address.

Benefits

No one-time cost to miners/validators.

No need to create a single contract that would be deployed everywhere, instead users could choose what to deploy.

Drawbacks

The signing account must have a non-negligible ether balance to upgrade.

AUTH

- AUTHUSURP

Leveraging the AUTH

opcode from EIP-3074, create a new opcode AUTHUSURP

that deploys code at the authorized

address.

Benefits

Just like the new transaction type above, this approach has no one-time cost to miners/validators, and users can choose what to deploy.

Also works well with sponsored transactions: the account to be upgraded doesn't need an ether balance.

Drawbacks

Comes with the drawbacks of EIP-3074: invokers potentially have total control over an account, it breaks some rare flash loan protections, and consumes three opcodes that might become deprecated in the future.

Conclusion

As far as the above options go, only three are serious candidates. Deploying bytecode and permanently breaking counterfactual deployments is unacceptable.

Assuming a default contract is reasonable, but takes an opinionated stance on what a smart contract wallet will look like. Allowing users to choose their wallet—either an EOA or smart, either with a new transaction type or with AUTHUSURP

—is more in line with the Ethereum ethos.

At the risk of letting my biases show through, I believe EIP-3074 brings a lot of benefits for users today, and—coupled with the AUTHUSURP

migration path off of EOAs—is a great direction to pursue.

Are there other approaches to migration that aren't listed here? If so, I'd love to know!

Stay tuned for a companion post on how EIP-3074 might work in a post-EOA world!