New Token Design

ERC20 tokens are Ethereum-based standard tokens that can be traded and transferred on the Ethereum network. But the essence of ERC20 is based on the EOA wallet design. EOA wallet has no state and code storage, and the smart contract wallet is different.

Almost all ERCs related to tokens are adding functions, our opinion is the opposite, we think the token contract should be simpler, more functions are taken care of by the smart contract wallet.

Our proposal is to design a simpler token asset based on the smart contract wallet,

It aims to achieve the following goals:

1. Keep the asset contract simple, only need to be responsible for the transaction function
2. approve and allowance functions are not managed by the token contract , approve and allowance should be configured at the user level instead of controlled by the asset contract, increasing the user's more playability , while avoiding part of the ERC20 contract risk.
3. Remove the transferForm function, and a better way to call the other party's token assets is to access the other party's own contract instead of directly accessing the token asset contract.
4. Forward compatibility with ERC20 means that all fungible tokens can be compatible with this proposal.

Examples

The third party calls the user's token transaction(transferForm)

,

Judges whether the receiving address is safe (safeTransferForm)

,

Permit extension for signed approvals (ERC-2612,``permit)

Authorizes the distribution of the user's own assets(approve, allowance)

,

Add the transfer hook function. (ERC-777, hook)

The above work should be handled by the user's smart contract wallet, rather than the token contract itself.

New Wallet Design

The smart contract wallet allows the user's own account to have state and code, bringing programmability to the wallet. We think there are more directions to expand. For example, token asset management, functional expansion of token transactions, etc.

It aims to achieve the following goals:

1. Assets are allocated and managed by the wallet itself, such as approve and allowance, which are configured by the user's contract wallet, rather than controlled by the token asset contract, to avoid some existing ERC-20 contract risks.
2. Add the transferFungibleToken function, the transaction initiated by the non-smart wallet itself or will verify the allowance amount
3. Users can choose batch approve and batch transfer. Batch approve can greatly reduce gas. The contract wallet itself manages the authorization status of all assets, and batch approve can be performed without calling multiple asset contracts.
4. Users can choose to add hook function before and after their transferFungibleToken to increase the user's more playability
5. The user can choose to implement the receive hook