Multicalls

Multicall is a key advantage of Account abstraction that enables you to aggregate multiple transactions into one.

Think about the Approve + Swap feature present on most DEXes. What if you could aggregate both calls into one? That would create a huge impact on dApps UX right? Welcome to Multicalls.

To understand how to aggregate calls, we need to first take a look at theexecute method on theaccount object.

Theexecute method

Theexecute method executes one or multiple calls using the account contract. If there is only one call,transactions will be an object that contains the parameters below. If there are multiple calls,transactions will be an array oftransactions objects:

Thetransactions object structure:

- contractPayload.contractAddress
- - the address of the contract
- contractPayload.entrypoint
- - the entrypoint of the contract
- contractPayload.calldata
- - (defaults to []) the calldata
- contractPayload.signature
- - (defaults to []) the signature
-

abi - (optional) the abi of the contract for better displaying

ThetransactionsDetail object may include any of:

- transactionsDetail.maxFee
- - Max Fee that will be used to execute the call(s)
- transactionsDetail.nonce
- - Nonce for the transaction
- transactionsDetail.version
- - Version for the transaction (default is 1)
-

Single vs Multicalls

In this section, we are going to take a look at how you perform single vs multiple calls using theexecute method.

Single call:

```

Copy constcall=awaitaccount.execute( {
contractAddress:'0x049d36570d4e46f48e99674bd3fcc84644ddd6b96f7c741b1562b82f9e004dc7',// ETH contract address
entrypoint:'approve', calldata:starknet.stark.compileCalldata( {
spender:"0x15e90f807a00a01df845460324fbcd33986f2df3cc9d981e9e8b5005b7f595e", amount:{ type:'struct', low:'1',// 1
wei high:'0', } } ), }, undefined, { nonce:'10', } );

```

multicalls:

```

Copy constmultiCall=awaitaccount.execute( [ {
contractAddress:'0x049d36570d4e46f48e99674bd3fcc84644ddd6b96f7c741b1562b82f9e004dc7',// ETH contract address
entrypoint:'approve', calldata:starknet.stark.compileCalldata( {
spender:"0x15e90f807a00a01df845460324fbcd33986f2df3cc9d981e9e8b5005b7f595e", amount:{ type:'struct', low:'1',// 1
wei high:'0', } } ), }, { contractAddress:'0x15e90f807a00a01df845460324fbcd33986f2df3cc9d981e9e8b5005b7f595e',
```

entrypoint:'transfer_ether', calldata:['1','0'],// 1 wei }], undefined, { nonce:'10', });

```

Previous AA use cases Next Cosigners - Guardians and 2FA

Last updated3 months ago
```