API Kit

TheAPI Kit(opens in a new tab) facilitates the interaction with theSafe Transaction Service API(opens in a new tab) , allowing to propose and share transactions with the other signers of a Safe, sending the signatures to the service to collect them, getting information about a Safe (like reading the transaction history, pending transactions, enabled Modules and Guards, etc.), among other features.

.mui-style-tn3bmg{box-sizing:border-box;margin:0;-webkit-flex-direction:row;-ms-flex-direction:row;flex-direction:row;margin-top:24px;}

.mui-style-x3r9fo{overflow:hidden;-webkit-transition:all 0.2s ease-in-out;transition:all 0.2s ease-in-out;border:1px solid;border-color:transparent;background-color:transparent;height:100%;width:100%;}.mui-style-x3r9fo:hover{border-color:var(--mui-palette-secondary-light);} .mui-style-1vkna86{background-color:var(--mui-palette-background-paper);color:var(--mui-palette-text-primary);-webkit-transition:box-shadow 300ms cubic-bezier(0.4, 0, 0.2, 1) 0ms;transition:box-shadow 300ms cubic-bezier(0.4, 0, 0.2, 1) 0ms;border-radius:6px;box-shadow:var(--mui-shadows-1);background-image:var(--mui-overlays-1);overflow:hidden;-webkit-transition:all 0.2s ease-in-out;transition:all 0.2s ease-in-out;border:1px solid;border-color:transparent;background-color:transparent;height:100%;width:100%;}.mui-style-1vkna86:hover{border-color:var(--mui-palette-secondary-light);} .mui-style-2ea2lo{box-sizing:border-box;display:-webkit-box;display:-webkit-flex;display:-ms-flexbox;display:flex;-webkit-box-flex-wrap:wrap;-webkit-flex-wrap:wrap;-ms-flex-wrap:wrap;flex-wrap:wrap;width:100%;margin:0;-webkit-flex-direction:row;-ms-flex-direction:row;flex-direction:row;-webkit-flex-direction:column;-ms-flex-direction:column;flex-direction:column;-webkit-box-pack:center;-ms-flex-pack:center;-webkit-justify-content:center;justify-content:center;-webkit-align-items:center;-webkit-box-align:center;-ms-flex-align:center;align-items:center;} .mui-style-1stacw4{margin:0;font-family:DM Sans,sans-serif;font-weight:400;font-size:20px;line-height:30px;font-weight:500;margin-top:8px;margin-bottom:8px;}@media (min-width:600px){.mui-style-1stacw4{font-size:24px;line-height:32px;}} #### @safe-global/api-kit .mui-style-30ww2h{margin:0;font-family:DM Sans,sans-serif;font-weight:400;font-size:14px;line-height:24px;color:var(--mui-palette-text-secondary);}

Quickstart

In this guide we will see how to propose transactions to the service and collect the signatures from the owners so they become executable.

For more detailed information, see the guideIntegrating the Protocol Kit and API Kit(opens in a new tab) and theAPI Kit Reference .

Prerequisites

1. Node.js and npm(opens in a new tab)
2. A Safe with several signers

Install dependencies

To add the API Kit to your project, run:

yarn

add

@safe-global/api-kit

Instantiate an EthAdapter

First of all, we need to create anEthAdapter , which contains all the required utilities for the SDKs to interact with the blockchain. It acts as a wrapper forweb3.js(opens in a new tab) orethers.js(opens in a new tab) Ethereum libraries.

Depending on the library used by the dapp, there are two options:

- Create anEthersAdapter instance(opens in a new tab)
- Create aWeb3Adapter instance(opens in a new tab)

Once the instance ofEthersAdapter orWeb3Adapter is created, it can be used in the initialization of the API Kit.

import { EthersAdapter } from

'@safe-global/protocol-kit'

const

provider

=

new

ethers .JsonRpcProvider (config . RPC_URL) const

signer

=

new

ethers .Wallet (config . SIGNER_ADDRESS_PRIVATE_KEY , provider)

const

ethAdapter

=

new

EthersAdapter ({ ethers , signerOrProvider : signer })

Initialize the API Kit

We need to create an instance of the API Kit. In chains where Safe provides a Transaction Service, it's enough to specify thechainId . You can set your own service using the optionaltxServiceUrl parameter.

import SafeApiKit from

'@safe-global/api-kit'

const

apiKit

=

new

SafeApiKit ({ chainId :

1 n })

// or using a custom service const

apiKit

=

new

SafeApiKit ({ chainId :

1 n ,

// set the correct chainId txServiceUrl :

'https://url-to-your-custom-service' })

Propose a transaction to the service

Before a transaction can be executed, any of the Safe signers needs to initiate the process by creating a proposal of a transaction. We send this transaction to the service to make it accessible by the other owners so they can give their approval and sign the transaction as well.

import Safe from

'@safe-global/protocol-kit'

```
// Create Safe instance const
protocolKit
=
await
Safe .create ({ ethAdapter , safeAddress :
config . SAFE_ADDRESS })
// Create transaction const
safeTransactionData :
MetaTransactionData
= { to :
'0x' , value :
'1' ,
// 1 wei data :
'0x' , operation :
OperationType .Call }
const
safeTransaction
=
await
protocolKit .createTransaction ({ transactions : [safeTransactionData] })
const
senderAddress
=
await
signer .getAddress () const
safeTxHash
=
await
protocolKit .getTransactionHash (safeTransaction) const
signature
=
await
protocolKit .signHash (safeTxHash)
// Propose transaction to the service await
apiKit .proposeTransaction ({ safeAddress :
await
protocolKit .getAddress () , safeTransactionData :
```

safeTransaction .data , safeTxHash , senderAddress , senderSignature :

signature .data })

Retrieve the pending transactions

Different methods in the API Kit are available to retrieve pending transactions depending on the situation. To retrieve a transaction given the Safe transaction hash use the method that's not commented.

const

transaction

=

await

service .getTransaction ("") // const transactions = await service.getPendingTransactions() // const transactions = await service.getIncomingTransactions() // const transactions = await service.getMultisigTransactions() // const transactions = await service.getModuleTransactions() // const transactions = await service.getAllTransactions()

Confirm the transaction

In this step we need to sign the transaction with the Protocol Kit and submit the signature to the Safe Transaction Service using theconfirmTransaction method.

const

safeTxHash

=

transaction .transactionHash const

signature

=

await

protocolKit .signHash (safeTxHash)

// Confirm the Safe transaction const

signatureResponse

=

await

apiKit .confirmTransaction (safeTxHash ,

signature .data)

The Safe transaction is now ready to be executed. This can be done using the Safe{Wallet} web interface, the Protocol Kit or any other tool that's available.

Migrating to V2 Reference