Safe Factory reference

create

Returns an instance of the Safe Factory.

import { SafeFactory } from

'@safe-global/protocol-kit'

const

safeFactory

=

await

SafeFactory .create ({ ethAdapter })

- TheisL1SafeSingleton
- flag
- Two versions of the Safe contracts are available:Safe.sol(opens in a new tab)
- that doesn't trigger events to save gas andSafeL2.sol(opens in a new tab)
- that does, which is more appropriate for L2 networks.
- By default,Safe.sol
- will only be used on Ethereum Mainnet. For the rest of the networks where the Safe contracts are already deployed, theSafeL2.sol
- contract will be used unless you add theisL1SafeSingleton
- flag to force using theSafe.sol
- contract.
- const
- safeFactory
- =
- await
- SafeFactory
- .create
- ({ ethAdapter
- ,
- isL1SafeSingleton
- :
- true
- })
- ThecontractNetworks
- property
- If the Safe contracts aren't deployed to your current network, thecontractNetworks
- property will be required to point to the addresses of the Safe contracts previously deployed by you.
- import
- { ContractNetworksConfig }
- from
- '@safe-global/protocol-kit'
- const
- chainId
- =
- await
- ethAdapter
- .getChainId
- ()
- const
- contractNetworks
- :
- ContractNetworksConfig
- =
- {
- [chainId]
- :
- {
- safeSingletonAddress

- :
- "
- ,
- safeProxyFactoryAddress
- :
- "
- ,
- multiSendAddress
- :
- "
- ,
- multiSendCallOnlyAddress
- :
- "
- ,
- fallbackHandlerAddress
- :
- "
- ,
- signMessageLibAddress
- :
- "
- ,
- createCallAddress
- :
- "
- ,
- simulateTxAccessorAddress
- :
- "
- ,
- safeSingletonAbi
- :
- "
- ,
- // Optional. Only needed with web3.js
- safeProxyFactoryAbi
- :
- "
- ,
- // Optional. Only needed with web3.js
- multiSendAbi
- :
- "
- ,
- // Optional. Only needed with web3.js
- multiSendCallOnlyAbi
- :
- "
- ,
- // Optional. Only needed with web3.js
- fallbackHandlerAbi
- :
- "
- ,
- // Optional. Only needed with web3.js
- signMessageLibAbi
- :
- "
- ,
- // Optional. Only needed with web3.js
- createCallAbi
- :
- "
- ,
- // Optional. Only needed with web3.js
- simulateTxAccessorAbi
- :

- "
- // Optional. Only needed with web3.js
- }
- }
- const
- safeFactory
- =
- await
- SafeFactory
- .create
- ({ ethAdapter
- ,
- contractNetworks })
- ThesafeVersion
- property
- TheSafeFactory
- constructor also accepts thesafeVersion
- property to specify the Safe contract version that will be deployed. This string can take the values1.0.0
- ,1.1.1
- ,1.2.0
- ,1.3.0
- or1.4.1
- . If not specified, theDEFAULT_SAFE_VERSION
- value will be used.
- const
- safeVersion
- =
- 'X.Y.Z'
- const
- safeFactory
- =
- await
- SafeFactory
- .create
- ({ ethAdapter
- ,
- safeVersion })

deploySafe

Deploys a new Safe and returns an instance of the Protocol Kit connected to the deployed Safe. The address of the singleton, Safe contract version, and the contract (Safe.sol orSafeL2.sol) of the deployed Safe will depend on the initialization of thesafeFactory instance.

const

safeAccountConfig :

SafeAccountConfig

= { owners , threshold , to ,

// Optional data ,

// Optional fallbackHandler ,

// Optional paymentToken ,

// Optional payment ,

// Optional paymentReceiver // Optional }

const

protocolKit

=

await

safeFactory .deploySafe ({ safeAccountConfig })

This method can optionally receive thesaltNonce parameter.

const

safeAccountConfig :

SafeAccountConfig

= { owners , threshold , to ,

// Optional data ,

// Optional fallbackHandler ,

// Optional paymentToken ,

// Optional payment ,

// Optional paymentReceiver // Optional }

const

saltNonce

=

"

const

protocolKit

=

await

safeFactory .deploySafe ({ safeAccountConfig , saltNonce })

Optionally, some properties can be passed as execution options:

const

options :

Web3TransactionOptions

= { from ,

// Optional gas ,

// Optional gasPrice ,

// Optional maxFeePerGas ,

// Optional maxPriorityFeePerGas // Optional nonce // Optional }

const

options :

EthersTransactionOptions

= { from ,

// Optional gasLimit ,

// Optional gasPrice ,

// Optional maxFeePerGas ,

// Optional maxPriorityFeePerGas // Optional nonce // Optional }

const

protocolKit

=

await

safeFactory .deploySafe ({ safeAccountConfig , safeDeploymentConfig , options })

It can also take an optional callback, which receives thetxHash of the Safe deployment transaction before returning a new instance of the Protocol Kit:

const

callback

= (txHash :

string) :

void

=> { console .log ({ txHash }) }

const

protocolKit

=

await

safeFactory .deploySafe ({ safeAccountConfig , callback })

Reference Safe

Was this page helpful?

Report issue