Using the Kit

The following are some examples of the capabilities of theContractKit , assuming it is already connected to a node. If you aren't connected,here is a refresher.

Setting Default Tx Options

kit allows you to set default transaction options:

import

{ CeloContract }

from

"@celo/contractkit" ;

let accounts =

await kit . web3 . eth . getAccounts ( ) ; kit . defaultAccount = accounts [0] ; // paid gas in cUSD await kit . setFeeCurrency (CeloContract . StableToken) ;

SetfeeCurrency

for a transaction

You can set thefeeCurrency for each transaction individually by setting thefeeCurrency field in the.send() method. ThefeeCurrency field accepts contract addresses of whitelisted fee currencies.

let cUSDcontract =

await kit . contracts . getStableToken ( ) ; let cUSDtx =

await cUSDcontract . transfer (someAddress , amount) . send ({

feeCurrency : cUSDcontract . address

}) ;

Getting the Total Balance

This method from thekit will return the CELO, locked CELO, cUSD and total balance of the address

let totalBalance =

await kit . getTotalBalance (myAddress) ;

Deploy a contract

Deploying a contract with the default account already set. Simply send a transaction with noto: field. See more about sending custom transactions below.

You can verify the deployment on theAlfajores block explorer here . Wait for the receipt and log it to get the transaction details.

let bytecode =

"0x608060405234..." ;

// compiled Solidity deployment bytecode

let tx =

await kit . sendTransaction ({ data : bytecode , }) ;

let receipt = tx . waitReceipt ( ) ; console . log (receipt) ;

Sending Custom Transactions

The Celo transaction object is not the same as Ethereum's. There is a new optional field present:

- feeCurrency
- (address of the ERC20 contract to use to pay for gas)

feeCurrency enables transaction fees to be paid in currencies other than CELO. The currently supported fee currencies are CELO, cUSD and cEUR. You can specify the currency by passing the contract address of the currency you would like the transaction fees to be paid in.

Celo accepts original Ethereum type transactions as well, so you can use Ethereum signing tools (like Metamask) as well as Celo specific wallets and tools. You can read more about these transaction formats inCIP 35 .

For a raw transaction:

const tx = kit . sendTransaction ({ from : myAddress , to : someAddress , value : oneGold , }) ; const hash =

await tx . getHash ( ) ; const receipt =

await tx . waitReceipt ( ) ; When interacting with a web3 contract object:

const goldtoken =

await kit . _web3Contracts . getGoldToken ( ) ; const oneGold = kit . web3 . utils . toWei ("1" ,

"ether") ;

const txo =

await goldtoken . methods . transfer (someAddress , oneGold) ; const tx =

await kit . sendTransactionObject (txo ,

{ from : myAddress }) ; const hash =

await tx . getHash ( ) ; const receipt =

await tx . waitReceipt ( ) ;

Interacting with Custom contracts

You can use ContractKit to interact with any deployed smart contract, provided you have the contract address and theABI . To do so, you will initialize a newweb3 Contract instance. Then you can call functions on the contract instance to read state or send transactions to update the contract. You can see some code snippets below. For a more comprehensive example, see theInteracting with Custom Contracts section of the Deploy a Contract code example.

let cUSDcontract =

await kit . contracts . getStableToken ( ) ; let contract =

new

kit . connection . web3 . eth . Contract (ABI , address) ;

// Init a web3.js contract instance let name =

await instance . methods . getName ( ) . call ( ) ;

// Read contract state

// Specifying the 'from' account and 'feeCurrency' is optional // Transactions with an unspecified feeCurrency field will default to paying fees in CELO const tx =

await instance . methods . setName (newName) . send ({ from : account . address , feeCurrency : cUSDcontract . address }) ;

Selling CELO only if the rate is favorable

// This is at lower price I will accept in cUSD for every CELO const favorableAmount =

100 ; const amountToExchange = kit . web3 . utils . toWei ("10" ,

"ether") ; const oneGold = kit . web3 . utils . toWei ("1" ,

"ether") ; const exchange =

await kit . contracts . getExchange ( ) ;

const amountOfcUsd =

await exchange . quoteGoldSell (oneGold) ;

if

(amountOfcUsd

favorableAmount)

{ const goldToken =

await kit . contracts . getGoldToken ( ) ; const approveTx =

await goldToken . approve (exchange . address , amountToExchange) . send ( ) ; const approveReceipt =

await approveTx . waitReceipt ( ) ;

const usdAmount =

await exchange . quoteGoldSell (amountToExchange) ; const sellTx =

await exchange . sellGold (amountToExchange , usdAmount) . send ( ) ; const sellReceipt =

await sellTx . waitReceipt ( ) ; }

Buying all the CELO I can, with the cUSD in my account

const stableToken =

await

this . contracts . getStableToken ( ) ; const exchange =

await

this . contracts . getExchange ( ) ;

const cUsdBalance =

await stableToken . balanceOf (myAddress) ;

const approveTx =

await stableToken . approve (exchange . address , cUsdBalance) . send ( ) ; const approveReceipt =

await approveTx . waitReceipt ( ) ;

const goldAmount =

await exchange . quoteUsdSell (cUsdBalance) ; const sellTx =

await exchange . sellDollar (cUsdBalance , goldAmount) . send ( ) ; const sellReceipt =

await sellTx . waitReceipt ( ) ;Edit this page Previous Celo ContractKit Setup Next Celo Core Contracts (Wrapper/Registry)