Quick start

Installation

npm install @1inch/limit-order-protocol-utils

or

yarn add @1inch/limit-order-protocol-utils Note

@1inch/limit-order-protocol package is now used for smart contract distribution and no longer contains this library.

Example on how to create and fill a limit order

import Web3 from

'web3' ; import

{ limirOrderProtocolAdresses , seriesNonceManagerContractAddresses , ChainId , Erc20Facade , LimitOrderBuilder , LimitOrderProtocolFacade , LimitOrderPredicateBuilder , NonceSeriesV2 , SeriesNonceManagerFacade , SeriesNonceManagerPredicateBuilder , Web3ProviderConnector }

from

'@1inch/limit-order-protocol-utils' ;

const chainId = ChainId . etherumMainnet ;

// suggested, or use your own number const connector =

new

Web3ProviderConnector (new

Web3 ('...')) ; const contractAddress = limirOrderProtocolAdresses [chainId] ; const seriesContractAddress = seriesNonceManagerContractAddresses [chainId] ;

const limitOrderProtocolFacade =

new

LimitOrderProtocolFacade (contractAddress , chainId , connector) ; const seriesNonceManagerFacade =

new

SeriesNonceManagerFacade (seriesContractAddress , chainId , connector) ;] const seriesNonceManagerPredicateBuilder =

new

SeriesNonceManagerPredicateBuilder (seriesContractAddress , chainId , connector) ; const erc20Facade =

new

Erc20Facade (connector) ;

const expiration =

5444440000 ;

// Some moment in future const nonce = seriesNonceManagerFacade . nonce (NonceSeriesV2 . LimitOrderV3 , walletAddress) ;

// Creates predicate that restricts Limit Order invalidation conditions // Because timestampBelowAndNonceEquals is method of another contract arbitraryStaticCall() is necessary const simpleLimitOrderPredicate : LimitOrderPredicateCallData =

arbitraryStaticCall (seriesNonceManagerPredicateBuilder . facade , seriesNonceManagerPredicateBuilder . timestampBelowAndNonceEquals (NonceSeriesV2 . LimitOrderV3 , expiration , nonce , walletAddress ,) ,) ;

// Create a limit order and it's signature const limitOrder = limitOrderBuilder . buildLimitOrder ({ makerAssetAddress :

'0xbb4cdb9cbd36b01bd1cbaebf2de08d9173bc095c' , takerAssetAddress :

'0x111111111117dc0aa78b770fa6a738034120c302' , makerAddress : walletAddress , makerAmount :

'100' , takerAmount :

'200' , predicate : simpleLimitOrderPredicate , // permit = '0x', // receiver = ZERO_ADDRESS, // allowedSender = ZERO_ADDRESS, // getMakingAmount = ZERO_ADDRESS, // getTakingAmount = ZERO_ADDRESS, // preInteraction = '0x', // postInteraction = '0x', // eg Wrapped to native unwrapper }) ;

const limitOrderTypedData = limitOrderBuilder . buildLimitOrderTypedData (limitOrder ,) ; const limitOrderSignature = limitOrderBuilder . buildOrderSignature (walletAddress , limitOrderTypedData ,) ;

// Create a call data for fill the limit order const callData = limitOrderProtocolFacade . fillLimitOrder ({ order : limitOrder , signature : limitOrderSignature , makingAmount :

'100' , takingAmount :

'0' , thresholdAmount :

'50' }) ;

// Send transaction for the order filling // Must be implemented sendTransaction ({ from : walletAddress , gas :

210_000 ,

// Set your gas limit gasPrice :

40000 ,

// Set your gas price to : contractAddress , data : callData , }) ; Note: you can use any implementation for the provider. Just implement theProviderConnector interface:

class

MyProviderConnector

implements

ProviderConnector

{ //... } Edit this page Previous About Next Create a limit order