Creating an RFQ order

Parameters:

Field Type Description id number is a pass-through, integer identifier starting at 1 wrapEth boolean? when is true, then in case of a limit order withWETH as maker asset, taker will receiveETH instead ofWETH expiresInTimestamp number is the timestamp in seconds when the limit order will no longer be available for execution. For example: 1623166270029 makerAssetAddress string the address of the asset you want to sell (address of a token contract) takerAssetAddress string the address of the asset you want to buy (address of a token contract) makerAddress string an address of the maker (wallet address) takerAddress string? the address of the taker for whom the limit order is being created.This is an optional parameter , if it is not specified, then the limit order will be available for execution for everyone makerAmount string the number of maker asset tokens that you want to sell (in token units). For example: 5 DAI = 5000000000000000000 units takerAmount string the number of taker asset tokens that you want to receive for selling the maker asset (in token units). For example: 5 DAI = 5000000000000000000 units

Creating with a typescript/javascript:

import Web3 from

'web3' ; import

{ LimitOrderBuilder , Web3ProviderConnector , }

from

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

const contractAddress =

'0x7643b8c2457c1f36dc6e3b8f8e112fdf6da7698a' ; const walletAddress =

'0xd337163ef588f2ee7cdd30a3387660019be415c9' ; const chainId =

1 ;

const web3 =

new

Web3 ('...') ; // You can create and use a custom provider connector (for example: ethers) const connector =

new

Web3ProviderConnector (web3) ;

const limitOrderBuilder =

new

LimitOrderBuilder (contractAddress , chainId , connector) ;

const RFQorder = limitOrderBuilder . buildRFQOrder ({ id :

1 , expiresInTimestamp :

1623166102 , makerAssetAddress :

'0x111111111117dc0aa78b770fa6a738034120c302' , takerAssetAddress :

'0xbb4cdb9cbd36b01bd1cbaebf2de08d9173bc095c' , makerAddress : walletAddress , makerAmount :

'1000000000000000000' , takerAmount :

'9000000000000000000' , }) ;

Creating via CLI (with arguments):

npx limit-order-rfq-utils -- \ --operation = create \ --chainId = 56

\ --privateKey = { xxx }

\ --orderId = 1

\ --expiresIn = 300

\ --makerAssetAddress = 0x111111111117dc0aa78b770fa6a738034120c302 \ --takerAssetAddress = 0x1af3f329e8be154074d8769d1ffa4ee058b1dbc3 \ --makerAmount = 1000000000000000000

\ --takerAmount = 4000000000000000000

\ --takerAddress = ""

Creating via CLI (through prompt):

npx limit-order-rfq-utils As result you will receive a structure of RFQ order . Example:

{ "info" :

"29941961886664662331741887180822" , "makerAsset" :

"0x111111111117dc0aa78b770fa6a738034120c302" , "takerAsset" :

"0x1af3f329e8be154074d8769d1ffa4ee058b1dbc3" , "makerAssetData" :

"0x23b872dd00000000...0000" , "takerAssetData" :

"0x23b872dd00000000...0000" } Edit this page Previous RFQ order structure Next Filling a RFQ order