Create a limit order

LimitOrderBuilder.buildLimitOrder()

Parameters:

Field Type Description 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? by default contains a zero address, which means that a limit order is available for everyone to fill. If you set a value, then the limit order will be available for execution only for the specified address (private limit order) reciever string? by default contains a zero address, which means that taker asset will be sent to the address of the creator of the limit order. If you set a value, then taker asset will be sent to the specified address makingAmount string the number of maker asset tokens that you want to sell (in token units). For example: 5 DAI = 5000000000000000000 units takingAmount 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 predicate string? a predicate call data. Default:0x . SeePredicate docs permit string? a permit (EIP-2612) call data. Could be built usingutility library . Default:0x interaction string? a call data for InteractiveNotificationReceiver. See moreInteraction receiver docs . Default:0x

Example:

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 limitOrder = limitOrderBuilder . buildLimitOrder ({ makerAssetAddress :

'0xbb4cdb9cbd36b01bd1cbaebf2de08d9173bc095c' , takerAssetAddress :

'0x111111111117dc0aa78b770fa6a738034120c302' , makerAddress :

'0xfb3c7ebccccAA12B5A884d612393969Adddddddd' , makingAmount :

'100' , takingAmount :

'200' , // predicate = '0x', // permit = '0x', // receiver = ZERO_ADDRESS, // allowedSender = ZERO_ADDRESS, // getMakingAmount = ZERO_ADDRESS, // getTakingAmount = ZERO_ADDRESS, // preInteraction = '0x', // postInteraction = '0x', }) ; const limitOrderTypedData = limitOrderBuilder . buildLimitOrderTypedData (limitOrder) ; const limitOrderSignature = limitOrderBuilder . buildOrderSignature (walletAddress , limitOrderTypedData) ; const limitOrderHash = limitOrderBuilder . buildLimitOrderHash (limitOrderTypedData) ; As result you will receive a structure oflimit order . Example:

{ "salt" :

"741670577637" , "makerAsset" :

"0xc02aaa39b223fe8d0a0e5c4f27ead9083c756cc2" , "takerAsset" :

"0x6b175474e89094c44da98b954eedeac495271d0f" , "maker" :

"0xfb3c7ebccccAA12B5A884d612393969Adddddddd" , "receiver" :

"0x0000000000000000000000000000000000000000" , "allowedSender" :

"0x0000000000000000000000000000000000000000" , "makingAmount" :

"100" , "takingAmount" :

"200" , "offsets" :

"3666552747586172848286066858439618331958618072439065963670861217005568" , "interactions" :

"0x20b83f2d000000000000000000000000000000000000000000000000000000000000006400000000000000000000000000000000000000000000000000000000000000c87e2d21830000000000000000(
}

Limit order signature

To fill a limit order, you need a typed data structure signature. You can create a signature following the example above.

But the example usesWeb3ProviderConnector which is designed to work with a wallet. If you need to get the signature on the server side, you can use thePrivateKeyProviderConnector and get the signature using the private key.

const walletAddress =

'0xd337163ef588f2ee7cdd30a3387660019be415c9' ;

const privateKey = 'd8d1f95deb28949ea0ecc4e9a0decf89e98422c2d76ab6e5f736792a388c56c7' ; const limitOrderTypedData : EIP712TypedData =

{ // ... } ;

const web3Provider =

new

Web3 ('...') ; const privateKeyProviderConnector =

new

PrivateKeyProviderConnector (privateKey , web3Provider) ;

```
const signature =
await privateKeyProviderConnector . signTypedData ( walletAddress , limitOrderTypedData ) ;
```

Edit this page Previous Quick start Next Limit order structure