Nonce

Nonce - this is the so-calledseries of limit orders. The nonce is useful when you need to create a bunch of limit orders with the ability to cancel them all later.

Methods Return type Purpose LimitOrderProtocolFacade.nonce(makerAddress)

SeriesNonceManagerFacade.nonce(series, makerAddress) Promise Retrieves series nonce for specific address and exact series (order group) LimitOrderProtocolFacade.increaseNonce()

SeriesNonceManagerFacade.increaseNonce(series) string (calldata) Increasing nonce allows to cancel all previous limit-orders within increased series. Seepredicate for more details. LimitOrderProtocolFacade.advanceNonce(count)

SeriesNonceManagerFacade.advanceNonce(series, count) string (calldata) Same as above, but increments nonce by specifiedcount . Mostly for developers purpose.

Difference betweenLimitOrderProtocolFacade

andSeriesNonceManagerFacade

Usual Limit-Order (default) and P2P Should useSeriesNonceManagerFacade for nonce management. Gasless orders Should useLimitOrderProtocolFacade for nonce management.

For Limit Orders and P2P

WARNING!

Since limit-order-protocol V3 you shouldn't use LimitOrderProtocolFacade forNonce when creating usual Limit-Order or P2P.

UseSeriesNonceManagerFacade as it store thoseNonces now. UseNonceSeriesV2 to get possible series values.

Following change is necessary in order to use nonces in v3:

- const nonce = limitOrderProtocolFacade.nonce(walletAddress) - const predicate = limitOrderPredicateBuilder.nonceEquals(- walletAddress, - nonce, -);

+ const nonce = seriesNonceManagerFacade.nonce(NonceSeriesV2.LimitOrderV3, walletAddress); + const predicate = seriesNonceManagerPredicateBuilder.timestampBelowAndNonceEquals(+ NonceSeriesV2.LimitOrderV3, + expiration, + nonce, + walletAddress, +);

Example:

import Web3 from

'web3' ; import

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

from

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

const walletAddress =

'0xhhh...' ; const chainId =

1 ; const contractAddress = limirOrderProtocolAdresses [chainId] ; const seriesContractAddress = seriesNonceManagerContractAddresses [chainId] ;

const connector =

new

Web3ProviderConnector (new

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

new

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

new

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

new

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

new

LimitOrderPredicateBuilder (limitOrderProtocolFacade) ; const limitOrderBuilder =

new

LimitOrderBuilder (contractAddress , chainId , connector) ;

// Get the current nonce const nonce =

await seriesNonceManagerFacade . getNonce (contractAddress) ;

const

{ arbitraryStaticCall , and , timestampBelow }

= limitOrderPredicateBuilder ;

// Create a limit order with timestampBelow and nonceEquals predicate // Because timestampBelowAndNonceEquals is method of another contract arbitraryStaticCall() is necessary const predicate =

arbitraryStaticCall (seriesNonceManagerPredicateBuilder . facade , seriesNonceManagerPredicateBuilder . timestampBelowAndNonceEquals (NonceSeriesV2 . LimitOrderV3 , expiration , nonce , walletAddress ,) ,) ; const limitOrder = limitOrderBuilder . buildLimitOrder ({ ... , predicate }) ; Cancellation example:

// Cancel all v3 limit-orders by advance nonce, but not P2P, Gasless or anything else const cancelAllOrdersCallData = seriesNonceManagerFacade . increaseNonce (NonceSeriesV2 . LimitOrderV3 ,) ; sendTransaction ({ from : walletAddress , gas :

210_000 ,

// Set your gas limit gasPrice :

40000 ,

// Set your gas price to : contractAddress , data : cancelAllOrdersCallData , }) ;

For Gasless

Note

To make Gasless orders use less gas their own gas manager are hosted on same contract. UseLimitOrderProtocolFacade andLimitOrderPredicateBuilder

Example:

import Web3 from

'web3' ; import

{ limirOrderProtocolAdresses , LimitOrderProtocolFacade , LimitOrderPredicateBuilder , NonceSeriesV2 , Web3ProviderConnector }

from

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

const walletAddress =

'0xhhh...' ; const chainId =

1 ; const contractAddress = limirOrderProtocolAdresses [chainId] ;

const connector =

new

Web3ProviderConnector (new

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

new

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

new

LimitOrderPredicateBuilder (limitOrderProtocolFacade) ; const limitOrderBuilder =

new

LimitOrderBuilder (contractAddress , chainId , connector) ;

// Get the current nonce const nonce =

await limitOrderProtocolFacade . nonce (contractAddress) ; const timestamp = Math . floor (Date . now ( )

/

1000)

+ expireTimeSeconds ;

// Create a limit order with timestampBelowAndNonceEquals predicate // This is gas optimised predicate, so you dont need to construct and(timestampBelow(), nonceEquals()) const predicate = limitOrderPredicateBuilder . timestampBelowAndNonceEquals (timestamp , nonce , walletAddress ,) ; const limitOrder = limitOrderBuilder . buildLimitOrder ({ ... , predicate }) ; Cancellation example:

// Cancel all Galsess orders by advance nonce, but not Limit-Orders, P2P or anything else const cancelAllOrdersCallData = limitOrderProtocolFacade . increaseNonce ( ) ; sendTransaction ({ from : walletAddress , gas :

210_000 ,

// Set your gas limit gasPrice :

40000 ,

// Set your gas price to : contractAddress , data : cancelAllOrdersCallData , }) Edit this page Previous Cancel all limit orders Next Predicate