

# Predicate

## LimitOrderPredicateBuilder

A limit order can contain one or more predicates that indicate the logic of its validity. There are two types of predicate operators:

### Conditional operators:

Name Description and combine several predicates, return true when all predicates are valid or combine several predicates, return true when one of predicates is valid

### Comparative operators:

All comparative operators have two arguments:

[ comparative ] ( value :

string , calldata :

string ) How the operators works:

- On an operator call, the contract execute the calldata
- on the limit-order-protocol contract and compare a result
- with the value
- .
- In case if you need to call method on third party contract use `arbitraryStaticCall(targetAddress, calldata)`
- .

Name Description `eq` a result must be equal to the value `lt` a result must be less than the value `gt` a result must be greater than the value

### Built-in operators:

#### timestampBelowAndNonceEquals

timestampBelowAndNonceEquals ( timestamp :

number

| bigint , makerNonce :

number

| bigint , makerAddress :

string ) The predicate checks the same as `and(timestampBelow(), nonceEquals())` .

Gasless only!

For usual Limit Order and P2P use [SeriesNonceManagerPredicateBuilder.timestampBelowAndNonceEquals\(series, timestamp, nonce, maker\)](#) together with `arbitraryStaticCall(target, calldata)` . See examples below.

#### arbitraryStaticCall

arbitraryStaticCall ( targetAddress :

string

| Facade , calldata :

string ) Allows to call methods on third party contracts.

Example:

```
const predicate =
```

```
arbitraryStaticCall ( seriesContractAddress ,
```

```
// target third-party contract address seriesNonceManagerPredicateBuilder . nonceEquals ( ... ) ,
```

```
// calldata to execute on it ) ;
```

```
// More convenient with facade const predicate =
```

```
arbitraryStaticCall ( seriesNonceManagerPredicateBuilder . facade ,
```

```
// facade stores contract address as well seriesNonceManagerPredicateBuilder . nonceEquals ( ... ) ,
```

```
// same calldata ) ;
```

## Deprecated built-in operators:

WARNING!

To save gas consider using one of

- [SeriesNonceManagerPredicateBuilder.timestampBelowAndNonceEquals\(...\)](#)
- – for everything but gasless
- [timestampBelowAndNonceEquals\(...\)](#)
- – for gasless only

instead as it more optimal then separate `and(timestampBelow(), nonceEquals())` calls.

- `nonceEquals`
- `nonceEquals`
- (
- `makerAddress`
- :
- `string`
- ,
- `makerNonce`
- :
- `number`
- |
- `bigint`
- )
- Gasless only!
- For usual Limit Order and P2P use [SeriesNonceManagerPredicateBuilder.nonceEquals\(...\)](#)
- together with `arbitraryStaticCall(...)`
- .
- See examples below.
- The predicate checks that the `makerNonce`
- is equal to the nonce of `makerAddress`
- .
- `timestampBelow`
- `timestampBelow`
- (
- `timestamp`
- :
- `number`
- |
- `bigint`
- )
- The predicate checks that `timestamp`
- (seconds integer) is greater than the current time.

## Examples

Preparation code:

```

import Web3 from

'web3' ; import

{ limitOrderProtocolAdresses , seriesNonceManagerContractAddresses , Erc20Facade , LimitOrderBuilder ,
LimitOrderProtocolFacade , LimitOrderPredicateBuilder , NonceSeriesV2 , SeriesNonceManagerFacade ,
Web3ProviderConnector }

from

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

const connector =

new

Web3ProviderConnector ( new

Web3 ( '...' ) ) ; const contractAddress = limitOrderProtocolAdresses [ 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 limitOrderPredicateBuilder =

new

LimitOrderPredicateBuilder ( limitOrderProtocolFacade ) ;

const

{ or , and , timestampBelow , nonceEquals , timestampBelowAndNonceEquals , gt , lt , eq , arbitraryStaticCall , }
= limitOrderPredicateBuilder ;

// Example data const makerAddress =

'0x5fa31604fc5dcebfccac2481f9fa59d174126e5e6' ; const makerAsset =

'0xcc83bc1050244c98ac562f9faff408f069a137d7' ; const minimumAllowedBalance =

10 ; const expiration =

5444440000 ;

// Some moment in future const currentNonce =

4 ; Simple predicate for usual Limit-Order or P2P

// Because timestampBelowAndNonceEquals is method of another contract arbitraryStaticCall() is necessary const
simpleLimitOrderPredicate : LimitOrderPredicateCallData =

arbitraryStaticCall ( seriesNonceManagerPredicateBuilder . facade , seriesNonceManagerPredicateBuilder .
timestampBelowAndNonceEquals ( NonceSeriesV2 . LimitOrderV3 , expiration , nonce , walletAddress , ) , ) ; Complex
predicate for usual Limit-Order or P2P

const predicate =

and ( // Because timestampBelowAndNonceEquals is method of another contract arbitraryStaticCall() is necessary

```

```
arbitraryStaticCall ( seriesNonceManagerPredicateBuilder . facade , seriesNonceManagerPredicateBuilder .  
timestampBelowAndNonceEquals ( NonceSeriesV2 . LimitOrderV3 , expiration , nonce , walletAddress , ) , ) , gt (   
minimumAllowedBalance , arbitraryStaticCall ( makerAsset ,
```

```
// target address. Token contract in this case erc20Facade . balanceOf ( makerAsset , walletAddress , ) ) , ) ; Simple  
predicate for Gasless order
```

```
const simplePredicate : LimitOrderPredicateCallData =
```

```
timestampBelowAndNonceEquals ( expiration , currentNonce , makerAddress , ) ; Complex predicate for Gasless order
```

```
const balanceOfCalldata =
```

```
arbitraryStaticCall ( makerAsset ,
```

```
// target address. Token contract in this case erc20Facade . balanceOf ( makerAsset , walletAddress , ) ) ; const  
complexPredicate : LimitOrderPredicateCallData =
```

```
or ( and ( timestampBelowAndNonceEquals ( expiration , currentNonce , makerAddress , ) , gt ( '10' , makerAsset ,  
balanceOfCalldata ) , ) , or ( timestampBelow ( expiration ) , lt ( '20' , makerAsset , balanceOfCalldata ) , ) , eq ( '30' ,  
makerAsset , balanceOfCalldata ) , ) ; Edit this page Previous Nonce Next SeriesNonceManager predicate
```