

SeriesNonceManager predicate

Make sure you have read and understand

- [predicate](#)
- [seriesnonce](#)

Since Limit Order Protocol V3 nonce management are delegated to SeriesNonceManager smart contract in terms of usual Limit Orders and P2P (aka "private limit order"), there are two helpful classes:

- SeriesNonceManagerFacade
- — an interface for interacting with smart contract
- SeriesNonceManagerPredicateBuilder
- — an [LimitOrderPredicateBuilder](#)
- -compatible DSL for building predicate, which relays on SeriesNonceManager

SeriesNonceManagerFacade

See [nonce](#)

SeriesNonceManagerPredicateBuilder

Partially implements [LimitOrderPredicateBuilder](#) for same purpose, but for SeriesNonceManager smart contract.

Incompatible with Gasless!

For Gasless orders use [LimitOrderPredicateBuilder](#) .

Built-in operators

timestampBelowAndNonceEquals

timestampBelowAndNonceEquals (timestamp :

number

| bigint , makerNonce :

number

| bigint , makerAddress :

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

nonceEquals

nonceEquals (makerAddress :

string , makerNonce :

number

| bigint) The predicate checks that the makerNonce is equal to the nonce of makerAddress .

WARNING!

To save gas consider using one of

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

- – for gasless only

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

Examples

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

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

See [LimitOrderPredicateBuilder](#) for more examples on using `LimitOrderPredicateBuilder` and `SeriesNonceManagerPredicateBuilder` together. [Edit this page](#) [Previous Predicate](#) [Next Limit order remaining](#)