Breaking changes in V3

Limit Order Protocol

itself changes

- Now contract is pubshed to npm as@1inch/limit-order-protocol
- Order
- struct has changed as following:
- struct Order {
- uint256 salt;
- address makerAsset;
- address takerAsset;
- address maker;
- address receiver;
- address allowedSender; // equals to Zero address on public orders
- uint256 makingAmount;
- uint256 takingAmount;
- -
- // Was in v2
- -
- // bytes makerAssetData;
- -
- // bytes takerAssetData;
- -
- // bytes getMakingAmount; // this.staticcall(abi.encodePacked(bytes, swapTakerAmount)) => (swapMakerAmount)
- -
- // bytes getTakingAmount; // this.staticcall(abi.encodePacked(bytes, swapMakerAmount)) => (swapTakerAmount)
- -
- // bytes predicate; // this.staticcall(bytes) => (bool)
- -
- // bytes permit; // On first fill: permit.1.call(abi.encodePacked(permit.selector, permit.2))
- -
- // bytes interaction;
- +
- // Now in v3
- +
- uint256 offsets;
- +
- bytes interactions; // concat(makerAssetData, takerAssetData, getMakingAmount, getTakingAmount, predicate, permit, preIntercation, postInteraction)
- }
- where offset is bytes, where every 32's bytes represents offset of the n'ths interaction.
- Eg: for[2, 4, 6]
- offsets:
- (
- 2n
- <<
- 32n
- *
- 0n
-)
- +
- (
- 4n
- <<
- 32n
- *
- 1n
-)
- +
- (
- 6n
- <<
- 32n

- *
- 2n
-)
- // 0x000000060000000400000002
- SeeLimitOrderBuilder.joinStaticCalls()
- andLimitOrderBuilder.packInteractions()

- utils for help.

- Order.interaction

- is nowOrder.postInteraction
- ,
- as long asOrder.preInteraction

- was added.

- New arguments forfillOrder

- andfillOrderToWithPermit
- methodsfunction fillOrderToWithPermit(
- OrderLib.Order calldata order,
- bytes calldata signature,
- +
- bytes calldata interaction,
- uint256 makingAmount,
- uint256 takingAmount,
- -
- uint256 thresholdAmount,
- +
- uint256 skipPermitAndThresholdAmount,
- address target,
- bytes calldata permit
-)
-
    - interaction
-
    - is pre-interaction in fact.
-
    - skipPermit
-
    - is just 255'th byte ofskipPermitAndThresholdAmount
-
    - , when rest of bytes isthresholdAmount
-
    - SeefillLimitOrder()
-
    - ,fillOrderToWithPermit()
-
    - andpackSkipPermitAndThresholdAmount()
-
    - utils methods and helpers.

- Methodseq

- ,lt
- ,gt
- ,nonceEquals
- no more have address arguments. UsearbitraryStaticCall
- instead in case if you need read value from different smartcontract.

limit-order-protocol-utils

library changes:

- Now contract is pubshed to npm as@1inch/limit-order-protocol-utils

TheLimitOrderProtocolFacade

- newchainId
- argument of LimitOrderProtocolFacadeimport import { ChainId } from '@1inch/limit-order-protocol-utils/model/limit-order-protocol.model';
- new LimitOrderProtocolFacade(
- public readonly contractAddress: string,
- +
- private readonly chainId: ChainId | number,
- public readonly providerConnector: ProviderConnector,
-)
- LimitOrderProtocolFacade.fillLimitOrder
- andfillOrderToWithPermit
- have more abilities now:
- fillOrderToWithPermit({
- order,
- signature,
- -
- makerAmount,
- +
- makingAmount,
- -
- takerAmount,
- +
- takingAmount,
- thresholdAmount,
- targetAddress,
- permit,
- +
- interaction = ZX,
- +
- skipPermit = false,
- })
  - interaction
  - is pre-interaction in fact.
  - skipPermit
  - wether to skip maker's permit evaluation if it was evaluated before.
  - Useful if multiple orders was created with same nonce.Tip: you can just check if allowance exsists and then set it totrue
  - .
- simulateCalls(addresses[], calldatas[]): Promise
- no more avaible.
- simulate(address, calldata): Promise<{ success, rawResult }>
- was introduced insted, so you don't needcatch
- block anymore.

TheLimitOrderBuilder

- LimitOrderBuilder.buildLimitOrder
- have more abilities now:
- buildLimitOrder({
- makerAssetAddress,
- takerAssetAddress,
- makerAddress,
- receiver = ZERO_ADDRESS,
- -

- takerAddress = ZERO_ADDRESS,
- +
- allowedSender = ZERO_ADDRESS,
- -
- makerAmount,
- +
- makingAmount,
- -
- takerAmount,
- +
- takingAmount,
- predicate = ZX,
- permit = ZX,
- +
- getMakingAmount,
- +
- getTakingAmount,
- -
- interaction = ZX,
- +
- postInteraction = ZX,
- +
- preInteraction = ZX,
- +
- salt = this.generateSalt(),

- }: LimitOrderData)

- LimitOrderPredicateBuilder.timestampBelowAndNonceEquals

- was added to reduce gas.-
- const predicate = and(
- -
- nonceEquals(
- -
- walletAddress,
- -
- currentNonce,
- -
-),
- -
- timestampBelow(timestamp),
- -
-);
- +
- const predicate = timestampBelowAndNonceEquals(
- +
- timestamp,
- +
- currentNonce,
- +
- walletAddress,
- +
-);

TheLimitOrderPredicateBuilder

- LimitOrderPredicateBuilder.arbitraryStaticCall
- was added to get values from other smartcontract.
- LimitOrderPredicateBuilder
- memberseq
- ,lt
- ,gt
- ,nonceEquals
- no more have address arguments. UsearbitraryStaticCall
- instead in case if you need read value from different smartcontract.
- Eg:

- eq
- (
- nonce
- ,
- arbitraryStaticCall
- (
- SIDE_NONCE_MANAGER_ADDRESS
- ,
- callData
-)
- ,
-)

TheSeriesNonceManagerFacade

- Was added to help you manage different groups of "cancel all" noncesEdit this page Previous Canceling a limit order Next 1inch limit order protocol