L78)

- • Each validator has aMsgProposedOperations(opens in a new tab)
- , which is one validator’s view of the operations queue
- • In the context of short-term orders, the block proposer should eventually be gossiped the short-term order and have it in their MsgProposedOperations
- • The block proposer would then optimistically place the short-term order inCheckTx(opens in a new tab)
- • Matches that short term orders were included in block during MsgProposedOperations would be included in block for all the validators in the network duringDeliverTx(opens in a new tab) Long-lived orders which may execute far in the future. These orders should not be lost during a validator restart (placed in the block after the order was received). In the event a validator restarts, all stateful orders areplaced back onto the in-memory orderbook(opens in a new tab). Likely to be used primarily by retail traders. The front end would be sending stateful orders for all order types other than market orders.

Two types of stateful orders:

1. Long-Term Orders • meant to be added to the orderbook as soon as possible. Due to certain technical limitations, long-term orders are placed in the block after they are written to state. E.g. ifMsgPlaceOrder is included in block N, taker order matching would occur for the long-term order in block N+1. • Order types requiring immediate execution such as fill-or-kill / immediate-or-cancel are disallowed as these should be placed as short term orders; Long-term FoK/IoC orders would never be maker orders, so there is no benefit to writing them to state.

2. Conditional Orders • execute when the oracle price becomes either LTE or GTE to specified trigger price, depending on the type of conditional order (e.g. stop loss sell = LTE, take profit buy = GTE) • orders are placed in the block after their condition is met and they become triggered • it is possible for a conditional order to become triggered in the same block they are initially written to state in. Conditional orders are placed in block ≥ N+1. placement message MsgPlaceOrder MsgPlaceOrder , long term or conditional order flag enabled onMsgPlaceOrder.Order.OrderId.OrderFlags

• valid OrderFlags values are 32 (conditional) and 64 (long-term) for stateful orders cancellation message MsgCancelOrder

Short term cancellations are handled best-effort, meaning they are only gossiped and not included in MsgProposedOperations MsgCancelOrder , long term or conditional order flag enabled onMsgCancelOrder.OrderId.OrderFlags expirations Good-Till-Block (GTB)

Short term orders have a maximum GTB of current block height +ShortBlockWindow(opens in a new tab) . Currently this value is 20 blocks, or about 30 seconds. Short term orders can only be GTB because in the interest of being resilient to chain halts or slowdowns. Good-Till-Block-Time (GTBT)

Stateful orders have a maximum GTBT of current block time +StatefulOrderTimeWindow(opens in a new tab) . Currently this value is 95 days.

GTBT is used instead of GTB to give a more meaningful expiration time for stateful orders. inclusion in block OperationRaw_ShortTermOrderPlacement insideMsgProposedOperations.OperationsQueue which is an app-injected message in the proposal. Included if and only if the short term order is included in a match. Normal cosmos transaction. The original Tx which included theMsgPlaceOrder orMsgCancelOrder would be included directly in the block. signature verification Short-term orders must undergo custom signature verification because they are included in an app-injected transaction.

The memclob stores each short term order placement’s raw transaction bytes in the memclob. When the order is included in a match, anOperationRaw_ShortTermOrderPlacement operation is included inMsgProposedOperations which contains these bytes.

DuringDeliverTx , we decode the raw transaction bytes into a transaction object and pass the transaction through the app’s antehandler which executes signature verification. If signature verification fails, theMsgProposedOperations execution returns an error and none of the operations are persisted to state. Operations for a given block is all-or-nothing, meaning all operations execute or none of them execute. Normal cosmos transaction signature verification, executed by the app’s antehandler. replay prevention Keep orders in state until after Good-Till-Block aka expiry (even if fully-filled or cancelled) Cosmos SDK sequence numbers, verified to be strictly increasing in the app’s antehandler.

Note that their use of sequence numbers requires stateful orders to be received in order otherwise they would fail. If placing multiple stateful orders they should be sent to the same validator to prevent issues. time placed (matching logic) CheckTx , immediately after placement transaction is received by the validator.

Short term orders are only included in a block when matched. See “time added to state” below. long-term: Block N+1 inPrepareCheckState(opens in a new tab) whereMsgPlaceOrder was included in block N conditional: Block N+1PrepareCheckState where the order was triggered inEndBlocker of block N what is stored in state OrderAmountFilledKeyPrefix : • key = OrderId • value = OrderFillAmount & PrunableBlockHeight

BlockHeightToPotentiallyPrunableOrdersPrefix : • key = block height • value = list of potentially prunable OrderIds

PrunableBlockHeight holds the block height at which we can safely remove this order from state. BlockHeightToPotentiallyPrunableOrders stores a list of order ids which we can prune for a certain block height. These are used in conjunction for replay prevention of short term orders StatefulOrderPlacementKeyPrefix :

• key = OrderId • value = Order

StatefulOrdersTimeSlice : • key = time • value = list of OrderIds expiring at this GTBT

OrderAmountFilledKeyPrefix : • key = OrderId • value = OrderFillAmount & PrunableBlockHeight (prunable block height unused for stateful orders) time added to state DeliverTx when part of a match included inMsgProposedOperations StatefulOrderPlacementKeyPrefix andStatefulOrdersTimeSlice :DeliverTx , theMsgPlaceOrder(opens in a new tab) is executed forMsgPlaceOrder msgs included in the block. The handler performs stateful validation, a collateralization check, and writes the order to state.Stateful orders are also written to the checkState in CheckTx for spam mitigation purposes.

OrderAmountFilledKeyPrefix : DeliverTx, when part of a match included inMsgProposedOperations time removed from state Always inEndBlocker(opens in a new tab) based off of prunable block height • cancelled by user: removed from state inDeliverTx forMsgCancelOrder

• forcefully-cancelled by protocol: removed from state inDeliverTx when processingOperationRaw_OrderRemoval operation. This operation type is included by the proposer inMsgProposedOperations when a stateful order is no longer valid. Removal reasons listedhere(opens in a new tab)

• fully-filled: removed from state inDeliverTx in the block in which they become fully filled. The order is added toRemovedStatefulOrderIds ofprocessProposerMatchesEvents to be used inEndBlocker to remove from the in-memory orderbook. • expired: pruned during EndBlocker based off of GTBT

also removed from state in CheckTx for cancellations. This is for spam mitigation purposes. time added to in-memory orderbook When placed inCheckTx , if not fully-matched When placed inPrepareCheckState , if not fully-matched time removed from in-memory orderbook • when fully-filled: removed inPrepareCheckState where invalid memclob state is purged via fully filled orders present inOrderIdsFilledInLastBlock

• when cancelled: (CheckTx) • when expired: PrepareCheckState, removed using memclob.openOrders.blockExpirationsForOrders data structure which stores expiration times for short term orders based off of GTB • when fully-filled: removed inPrepareCheckState where we purge invalid memclob state based off ofRemovedStatefulOrderIds

• when cancelled: removed inPrepareCheckState based off ofPlacedStatefulCancellations

• when expired: removed inPrepareCheckState byPurgeInvalidMemclobState , using the list ofExpiredStatefulOrderIds produced inEndBlocker