xChange Python API Reference

Important Functions

`place_order(` *`self, asset_code, order_type, order_side, qty, px`* **`)`**

> Places an order on the exchange
>
> **Arguments**:
> `asset_code` (`str`) The code of the asset to place an order for
> `order_type` (`pb.OrderSpecType`) The type of order (e.g. limit order, market order, etc.)
> `order_side` (`pb.OrderSpecSide`) The side of the order
> `qty` (`int`) The # of lots of the asset to buy/sell
> `px` (`Optional[float]`) The price to buy/sell at. Not required if this is a market order.
>
> **Returns** (`pb.PlaceOrderResponse`): The response from the exchange

`modify_order(` *`self, order_id, asset_code, order_type, order_side, qty, px`* **`)`**

> Modify an order that you've already placed. If no order corresponding to the specified `order_id` exists, this will place the new order anyway. This makes this function particularly useful for maintaining quotes on a book
>
> `order_id` (`str`) The ID of the order to replace
> `asset_code` (`str`) The code of the asset to modify
> `order_type` (`pb.OrderSpecType`) The type of order (e.g. limit order, market order, etc.)
> `order_side` (`pb.OrderSpecSide`) The side of the order
> `qty` (`int`) The # of lots of the asset to buy/sell
> `px` (`Optional[float]`) The price to buy/sell at. Not required if this is a market order.
>
> **Returns** (`pb.ModifyOrderResponse`): The response from the exchange

`cancel_order(` *`self, order_id`* **`)`**

> Cancel the order with the specified id
>
> `order_id` (`str`) The ID of the order to cancel
>
> **Returns** (`pb.CancelOrderResponse`): The response from the exchange

Important Classes

`class pb.PlaceOrderResponse`

> A response to a request to place an order
>
> **Fields**:
> `ok` (bool) Whether the request was successful
>
> - If `True`, the order has been scheduled for creation. It may still be rejected with a `RequestFailedMessage'
> - if `False`, request failed and no order will be placed
>
> `order_id` (str) If `ok=True`, this is the ID of the order that was scheduled for creation
> `message` (str) Message about why the request to place an order may have failed

`class pb.ModifyOrderResponse`

> A response to a request to modify an order
>
> **Fields**:
> `ok` (bool) Whether the request was successful
>
> - If `True`, the order has been scheduled for modification. It may still be rejected with a `RequestFailedMessage'
> - if `False`, request failed and no order will be modified
>
> `order_id` (str) If `ok=True`, this is the ID of the order that was scheduled for creation
> `message` (str) Message about why the request to modify an order may have failed

`class pb.CancelOrderResponse`

> A response to a request to cancel an order
>
> **Fields**:
> `ok` (bool) Whether the request was successful
>
> - If `True`, the order has been scheduled for cancellation. It may still be rejected with a `RequestFailedMessage'
> - if `False`, request failed and no order will be cancelled
>
> `message` (str) Message about why the request to cancel the order may have failed

`class pb.FeedMessage`

> A message sent from the exchange to the competitor that can be processed in the `handle_exchange_update` method of your bot. Exactly one of the following fields will actually be

populated. To find out which one, you can call

```
kind, _ = betterproto.which_one_of(feed_message, "msg")
```

After this, kind will be a string containing the name of the populated field (e.g. `kind=="pnl_msg"` if and only if `feed_message.pnl_msg` is populated). See the example bots for more details.

Fields:
`request_failed_msg` (pb.RequestFailedMessage) A message that tells you that your request to place/modify/cancel an order has failed
`pnl_msg` (pb.PnLMessage) An update containing PnL information for the competitor
`trade_msg` (pb.TradeMessage) A message containing info about a recent trade that occurred
`fill_msg` (pb.FillMessage) A message that tells you about an order of yours that filled.
`market_snapshot_msg` (pb.MarketSnapshotMessage) A message containing a snapshot of the order books for every asset
`liquidation_msg` (pb.LiquidationMessage) (*irrelevant for this year's competition*)
`generic_msg` (pb.GenericMessage) A miscellaneous message sent through the update stream. Watch this for important information updates from the exchange about the case.
br/>

class pb.**RequestFailedMessage**

Response sent when a request to place an order has failed. If this message is received, then the request corresponding to the provided order IDs could not be completed

Fields:
`type` (`pb.RequestFailedMessageType`) The type of the failed request that was sent (i.e. whether it was to place/modify/cancel an order)
`place_order_id` (`str`) The ID of the order that was unsuccessfully placed (or used to replace an old order), if any<`cancel_order_id` (`str`) The ID of the order that was unsuccessfully cancelled (or replaced), if any
`message` (`str`) The message associated with the failed request
`asset` (`str`) Asset code that the request was sent for
`timestamp` (`str`) Timestamp that the failure was noted

enum pb.**RequestFailedMessageType**

Represents the type of request that failed in a `pb.RequestFailedMessage`.

Members: `PLACE`, `MODIFY`, `CANCEL`

class pb.**PnlMessage**

An update containing PnL information for the competitor

Fields:
`realized_pnl` (`str`) The PnL of the competitor
`m2m_pnl` (`str`) Marked to Market PnL (a more accurate measure of your performance)
`timestamp` (`str`) The timestamp when this update was created

class pb.TradeMessage

A message containing info about a recent trade that occurred

Fields:
`asset` (`str`) The asset that the trade occurred in
`price` (`str`) The price that the trade occurred at
`qty` (`int`) The quantity of the trade
`timestamp` (`str`) The timestamp at which this trade occurred

class pb.FillMessage

An update containing info about a recent order fill that occurred

Fields:
`order_id` (str) The ID of the order that was filled
`asset` (str) The asset that was filled
`order_side` (pb.FillMessageSide) The side that the competitor was on.

- If `order_side==BUY`, then this fill resulted in you buying the asset
- If `order_side==SELL`, then this fill resulted in you selling the asset

`price` (str) The price level that was filled at
`filled_qty` (int) The quantity that was filled
`remaining_qty` (int) The remaining quantity in the order
`timestamp` / str: The timestamp at which the fill was processed

enum pb.FillMessageSide

Contains information about the side that fill occurred on

Members: `BUY`, `SELL`

class pb.MarketSnapshotMessage

Update containing information on books for every asset

Fields:
`books` (Dict[str, pb.MarketSnapshotMessageBook]) map from asset code to a snapshot of the order

book associated with that asset
`timestamp` (str) The time at which the market info was found

class pb.**MarketSnapshotMessageBook**

Information for individual asset within whole book update

Fields:
`asset` (str) The asset associated with the book
`bids` (List[pb.MarketSnapshotMessageBookPriceLevel]) The bid price levels
`asks` (List[pb.MarketSnapshotMessageBookPriceLevel]) The ask price levels containing qty and px

class pb.**MarketSnapshotMessageBookPriceLevel**

Quantities at each Price Level

Fields:
`px` (str) The price associated with this level. Note that this stored as a string in order to preserve precision
`qty` (int) The total quantity associated with this price level on the book

class pb.**GenericMessage**

A misc. event sent through the update stream. The `event_type` can be used to interpret the context surrounding the `message` text

Fields:
`event_type` (pb.GenericMessageType) The type of exchange event that was sent through the feed
`message` (str) The message text associated with that event

enum pb.**GenericMessageType**

Represents the type of a generic message sent over the data feed

Members: `MESSAGE`, `MESSAGE`, `INTERNAL_ERROR`, `COMPETITOR_DEACTIVATED`, `CHANNEL_REPLACED`, `ROUND_ENDED`, `RISK_LIMIT_BROKEN`

class pb.**LiquidationMessage** *not relevant for this year's competition*
