Deribit Multicast

Client Development Guide

v1.4.2 - 23 May 2022

Table of Contents

1 Introduction

Introducing multicasting of public events on the Deribit colocation network provides a low latency interface to distribute information that also significantly reduces resource requirements on both client and server side. Migrating clients to use multicasted events instead of event subscriptions on the Websocket API does not only allow them to react faster due to the lower latency and more efficient encoding, but also improves the processing speed of trading actions on the API nodes.

2 Multicast packet encoding

The events are sent in UDP multicast packets using the SBE (Simple Binary Encoding) format.

The multicast system groups events per currency and product (perpetual, options, futures) combinations. These groups can be associated with channels (the actual channel associations are maintained and shared in a separate document) where each channel can be assigned a separate multicast group address and UDP destination port number. Each channel keeps a limited history of 100.000 packets for recovery.

After the UDP protocol header, multicast packets start with a framing header that contains the Packet Length, Channel ID and a Sequence Number in the channel. The Packet Length is the total length of all the SBE messages (after the Sequence Number) in bytes. The main purpose of this header is to allow clients to detect when they miss packets and retrieve them, using the regular websocket/REST API.

Packet Length (uint16)
Channel ID (uint16)
Sequence Number (uint32)
SBE message
SBE message
SBE message
…

Note that all sample messages provided below, as well as the sample packet capture are from a local development system, with instruments defined solely for the purpose of generating data for testing/developing the multicast decoder. The product/instrument parameters used don’t (and are not intended to) match the production products/instruments.

A - SBE message format intro

SBE is an efficient binary encoding with features that allow e.g. extending the protocol with new elements in a backward compatible way. Also it has a fairly widely used/known standard, with a community behind it providing e.g. tooling for code generation.

Each SBE message contains a header, a number of fixed length fields, groups (variable length lists of items) and optionally variable length fields in this strict sequence.

Header (Fixed size)
Fixed length fields (Fixed size)
Groups
Variable length fields

A.1 SBE Header

```
<composite name="messageHeader">
      <type name="blockLength" primitiveType="uint16" />
      <type name="templateId" primitiveType="uint16" />
      <type name="schemaId" primitiveType="uint16" />
      <type name="version" primitiveType="uint16" />
      <type name="numGroups" primitiveType="uint16" />
      <type name="numVarDataFields" primitiveType="uint16" />
</composite>
```

- `blockLength` is the total number of bytes of the fixed fields in the message. The purpose of this field is that if a new fixed field is introduced (always after the existing fields), the client can decode the fields it is aware of and, based on this field can skip the additional items to find the beginning of the eventual group/variable fields, or the end of the message
- `templateId` refers to the ID of the corresponding message (e.g. 1001 for order book change)
- The `schemaId` and `version` fields are for indicating the ID of the XML specification and its version number
- `numGroups` is the number of groups/lists in the message. The purpose of this field is similar to the `blockLength` and allows introducing new groups in a backward compatible way
- `numVarDataFields` serve a similar purpose as `blockLength` and `numGroups`

A.2 Fixed length block

The list of single fixed length fields are in this block. Strictly in the sequence as defined in the XML specification. Optional fields are present with a default value, either explicitly mentioned in the XML message/field template, or as defined in the SBE specification for the particular primitive type.

A.3 Groups

Groups are a list of items. Each message can contain 0 or more groups. Each group can have 0 or more items. The items have a similar structure as SBE messages. They can consist of a fixed length block, (nested) groups and variable length fields. The information about the structure of the items, as well as the number of items (`numInGroup`) is described in the header of each group.

The group header definition:

```
<composite name="groupSizeEncoding">
      <type name="blockLength" primitiveType="uint16"/>
      <type name="numInGroup" primitiveType="uint16"/>
      <type name="numGroups" primitiveType="uint16"/>
      <type name="numVarDataFields" primitiveType="uint16"/>
</composite>
```

Note that if the the group header indicates that the entries do not have nested groups (`numGroups` = 0) or variable length data fields (`numVarDataFields` = 0) then the entries have a fixed length which is equal to blockLength. In this case the total number of bytes in the group can be calculated by : `blockLength * numInGroup`.

A.4 Variable length fields

Variable length fields begin with a length and a length number of bytes. In the current implementation variable fields are used for Instrument names in instrument events/messages.

```
<composite name="varString" description="Variable-length string">
      <type name="length" primitiveType="uint8" />
      <type name="varData" length="0" primitiveType="uint8" />
</composite>
```

3 Multicast events

In the initial implementation the platform multicasts 4 events

- instrument
- order book change
- ticker
- trades

The structure and content of the events is aimed to be the same or very similar to the current Websocket API subscription events.

A - Instrument

The purpose of this event is to enable the clients to get notified when a new instrument/book is created/closed and settled. Also to provide static instrument information in snapshots. Based on this event, the client can start tracking a new book, or stop the tracking of one. This event can/should also be used to maintain the Instrument ID/Name mapping, since it contains both of this information.

When instrument changes happen in a batch (e.g. closing option books) and multiple messages fit in a packet then they can be combined, even for different instruments for a combination of currency/product. When books are closed, then book change events (deleting the remaining levels) and instrument state change (close) events may be combined.

The instrument message is also sent on the snapshot channels as part of the snapshot, providing static instrument information.

Message specification

```
<message name="instrument" id="1000">
      <field name="header" id="1" type="messageHeader" />
      <field name="instrumentId" id ="2" type="uint32" />
      <field name="instrumentState" id="3" type="instrumentState" />
      <field name="kind" id="4" type="instrumentKind" />
      <field name="futureType" id="5" type="futureType" />
      <field name="optionType" id="6" type="optionType" />
      <field name="rfq" id="7" type="yesNo" />
      <field name="settlementPeriod" id="8" type="period" />
      <field name="settlementPeriodCount" id="9" type="uint16" />
      <field name="baseCurrency" id="10" type="string8" />
      <field name="quoteCurrency" id="11" type="string8" />
      <field name="counterCurrency" id="12" type="string8" />
      <field name="settlementCurrency" id="13" type="string8" />
      <field name="sizeCurrency" id="14" type="string8" />
      <field name="creationTimestampMs" id="15" type="uint64" />
      <field name="expirationTimestampMs" id="16" type="uint64" />
      <field name="strikePrice" id="17" type="double" />
```

```
      <field name="contractSize" id="18" type="double" />
      <field name="minTradeAmount" id="19" type="double" />
      <field name="tickSize" id="20" type="double" />
      <field name="makerCommission" id="21" type="double" />
      <field name="takerCommission" id="22" type="double" />
      <field name="blockTradeCommission" id="23" type="double" />
      <field name="maxLiquidationCommission" id="24" type="double" />
      <field name="maxLeverage" id="25" type="double" />
      <data name="instrumentName" id="26" type="varString" />
</message>
```

Enum Types used in the message

```
<enum name="instrumentState" encodingType="uint8">
      <validValue name="created">0</validValue>
      <validValue name="open">1</validValue>
      <validValue name="closed">2</validValue>
      <validValue name="settled">3</validValue>
</enum>
<enum name="instrumentKind" encodingType="uint8">
      <validValue name="future">0</validValue>
      <validValue name="option">1</validValue>
</enum>
<enum name="optionType" encodingType="uint8">
      <validValue name="not_applicable">0</validValue>
      <validValue name="put">1</validValue>
      <validValue name="call">2</validValue>
</enum>
<enum name="futureType" encodingType="uint8">
      <validValue name="not_applicable">0</validValue>
      <validValue name="reversed">1</validValue>
      <validValue name="linear">2</validValue>
</enum>
<enum name="period" encodingType="uint8">
      <validValue name="perpetual">0</validValue>
      <validValue name="minute">1</validValue>
      <validValue name="hour">2</validValue>
      <validValue name="day">3</validValue>
      <validValue name="week">4</validValue>
      <validValue name="month">5</validValue>
      <validValue name="year">6</validValue>
</enum>
```

Example message

```
Frame 4096: 227 bytes on wire (1816 bits), 227 bytes captured (1816 bits) on
interface lo, id 0
Ethernet II, Src: 00:00:00_00:00:00 (00:00:00:00:00:00), Dst:
00:00:00_00:00:00 (00:00:00:00:00:00)
Internet Protocol Version 4, Src: 127.0.0.1, Dst: 239.111.111.2
User Datagram Protocol, Src Port: 48210, Dst Port: 6100
Deribit SBE
    Framing Header
```

```
        Packet Length: 177
        Channel Id: 2
        Channel Sequence: 1
    Instrument State
        Header
            Root Block Length: 140
            Type: instrument (1000)
            Schema Id: 1
            Version: 1
            Num Groups: 0
            Num Vars: 1
        Instrument Id: 618
        Instrument State: created (0)
        Instrument Kind: option (1)
        Future Type: not applicable (0)
        Option Type: call (1)
        RFQ: 0
        Settlement Period: minute (1)
        Settlement Period Count: 15
        Base Currency: BTC
        Quote Currency: BTC
        Counter Currency: USD
        Settlement Currency: BTC
        Size Currency: BTC
        Creation Timestamp: May 14, 2022 06:35:05.000000000 UTC
        Expiration Timestamp: May 14, 2022 06:45:00.000000000 UTC
        Strike Price: 29200
        Contract Size: 1
        Minimum Trade Amount: 0,01
        Tick Size: 0,0001
        Maker Commission: 0,0001
        Taker Commission: 0,0005
        Block Trade Commission: 0,00015
        Max Liquidation Commission: 0
        Max Leverage: 0
        Instrument Name: BTC-14MAY22_0645-29200-C

0000   00 00 00 00 00 00 00 00 00 00 00 00 08 00 45 00   ..............E.
0010   00 d5 bc fc 40 00 20 11 bf a8 7f 00 00 01 ef 6f   ....@. ........o
0020   6f 02 bc 52 17 d4 00 c1 de 45 b1 00 02 00 01 00   o..R.....E......
0030   00 00 8c 00 e8 03 01 00 01 00 00 00 01 00 6a 02   ..............j.
0040   00 00 00 01 00 01 00 01 0f 00 42 54 43 00 00 00   ..........BTC...
0050   00 00 42 54 43 00 00 00 00 00 55 53 44 00 00 00   ..BTC.....USD...
0060   00 00 42 54 43 00 00 00 00 00 42 54 43 00 00 00   ..BTC.....BTC...
0070   00 00 a8 1d 47 c1 80 01 00 00 e0 31 50 c1 80 01   ....G......1P...
0080   00 00 00 00 00 00 00 84 dc 40 00 00 00 00 00 00   .........@......
0090   f0 3f 7b 14 ae 47 e1 7a 84 3f 2d 43 1c eb e2 36   .?{..G.z.?-C...6
00a0   1a 3f 2d 43 1c eb e2 36 1a 3f fc a9 f1 d2 4d 62   .?-C...6.?....Mb
00b0   40 3f 61 32 55 30 2a a9 23 3f 00 00 00 00 00 00   @?a2U0*.#?......
00c0   00 00 00 00 00 00 00 00 00 00 18 42 54 43 2d 31   ...........BTC-1
00d0   34 4d 41 59 32 32 5f 30 36 34 35 2d 32 39 32 30   4MAY22_0645-2920
00e0   30 2d 43                                          0-C
```

B - Order book change

This event contains changes of the order book levels. The changes can be new levels, change in the amount on the levels or deletion (no amount left), just as in the websocket subscription events.

Message Specification

```
<message name="book" id="1001">
      <field name="header" id="1" type="messageHeader" />
      <field name="instrumentId" id="2" type="uint32" />
      <field name="timestampMs" id="3" type="uint64" />
      <field name="prevChangeId" id="4" type="uint64" />
      <field name="changeId" id="5" type="uint64" />
      <field name="isLast" id="6" type="yesNo" />
      <group name="changesList" id="7" dimensionType="groupSizeEncoding">
            <field name="side" id="1" type="bookSide" />
            <field name="change" id="2" type="bookChange" />
            <!-- Use double (64 bit float) encoding, SBE FIX price/amount
                  decimal encoding makes sense when the
                  decimal point is on a somewhat fixed position, for crypto,
                  it can vary more by instrument -->
            <field name="price" id="3" type="double" />
            <field name="amount" id="4" type="double" />
      </group>
</message>
```

Enum types used in this message

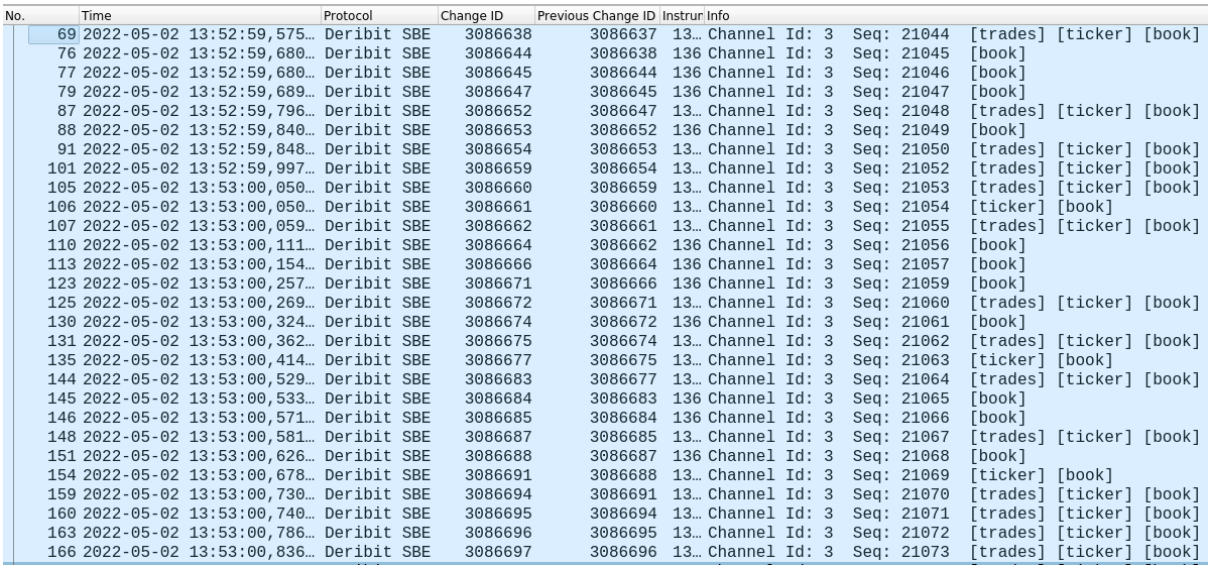
```
<enum name="bookChange" encodingType="uint8">
      <validValue name="created">0</validValue>
      <validValue name="changed">1</validValue>
      <validValue name="deleted">2</validValue>
</enum>

<enum name="bookSide" encodingType="uint8">
      <validValue name="ask">0</validValue>
      <validValue name="bid">1</validValue>
</enum>

<enum name="yesNo" encodingType="uint8">
      <validValue name="no">0</validValue>
      <validValue name="yes">1</validValue>
</enum>
```

The events use the same mechanism to ensure the consistency of the order book as the websocket events.

Each message contains a current `changeId` and `prevChangeId`.

The list of book changes may occasionally result in a message that exceeds the maximum packet size. In this case the platform splits the list of changes into multiple lists and generates a sequence of multiple complete order book change messages. The messages have the same fixed fields (`header`, `timestampMs`, `prevChangeId`, `changeId`) but the `isLast` field is set to the value 0 for all but the last message in the sequence, to indicate if the complete change list is sent.

Example sequence



No.	Time	Protocol	Change ID	Previous Change ID	Instrur	Info
69	2022-05-02 13:52:59,575…	Deribit SBE	3086638	3086637	13…	Channel Id: 3 Seq: 21044 [trades] [ticker] [book]
76	2022-05-02 13:52:59,680…	Deribit SBE	3086644	3086638	136	Channel Id: 3 Seq: 21045 [book]
77	2022-05-02 13:52:59,680…	Deribit SBE	3086645	3086644	136	Channel Id: 3 Seq: 21046 [book]
79	2022-05-02 13:52:59,689…	Deribit SBE	3086647	3086645	136	Channel Id: 3 Seq: 21047 [book]
87	2022-05-02 13:52:59,796…	Deribit SBE	3086652	3086647	13…	Channel Id: 3 Seq: 21048 [trades] [ticker] [book]
88	2022-05-02 13:52:59,840…	Deribit SBE	3086653	3086652	136	Channel Id: 3 Seq: 21049 [book]
91	2022-05-02 13:52:59,848…	Deribit SBE	3086654	3086653	13…	Channel Id: 3 Seq: 21050 [trades] [ticker] [book]
101	2022-05-02 13:52:59,997…	Deribit SBE	3086659	3086654	13…	Channel Id: 3 Seq: 21052 [trades] [ticker] [book]
105	2022-05-02 13:53:00,050…	Deribit SBE	3086660	3086659	13…	Channel Id: 3 Seq: 21053 [trades] [ticker] [book]
106	2022-05-02 13:53:00,050…	Deribit SBE	3086661	3086660	13…	Channel Id: 3 Seq: 21054 [ticker] [book]
107	2022-05-02 13:53:00,059…	Deribit SBE	3086662	3086661	13…	Channel Id: 3 Seq: 21055 [trades] [ticker] [book]
110	2022-05-02 13:53:00,111…	Deribit SBE	3086664	3086662	136	Channel Id: 3 Seq: 21056 [book]
113	2022-05-02 13:53:00,154…	Deribit SBE	3086666	3086664	136	Channel Id: 3 Seq: 21057 [book]
123	2022-05-02 13:53:00,257…	Deribit SBE	3086671	3086666	136	Channel Id: 3 Seq: 21059 [book]
125	2022-05-02 13:53:00,269…	Deribit SBE	3086672	3086671	13…	Channel Id: 3 Seq: 21060 [trades] [ticker] [book]
130	2022-05-02 13:53:00,324…	Deribit SBE	3086674	3086672	136	Channel Id: 3 Seq: 21061 [book]
131	2022-05-02 13:53:00,362…	Deribit SBE	3086675	3086674	13…	Channel Id: 3 Seq: 21062 [trades] [ticker] [book]
135	2022-05-02 13:53:00,414…	Deribit SBE	3086677	3086675	13…	Channel Id: 3 Seq: 21063 [ticker] [book]
144	2022-05-02 13:53:00,529…	Deribit SBE	3086683	3086677	13…	Channel Id: 3 Seq: 21064 [trades] [ticker] [book]
145	2022-05-02 13:53:00,533…	Deribit SBE	3086684	3086683	136	Channel Id: 3 Seq: 21065 [book]
146	2022-05-02 13:53:00,571…	Deribit SBE	3086685	3086684	136	Channel Id: 3 Seq: 21066 [book]
148	2022-05-02 13:53:00,581…	Deribit SBE	3086687	3086685	13…	Channel Id: 3 Seq: 21067 [trades] [ticker] [book]
151	2022-05-02 13:53:00,626…	Deribit SBE	3086688	3086687	136	Channel Id: 3 Seq: 21068 [book]
154	2022-05-02 13:53:00,678…	Deribit SBE	3086691	3086688	13…	Channel Id: 3 Seq: 21069 [ticker] [book]
159	2022-05-02 13:53:00,730…	Deribit SBE	3086694	3086691	13…	Channel Id: 3 Seq: 21070 [trades] [ticker] [book]
160	2022-05-02 13:53:00,740…	Deribit SBE	3086695	3086694	13…	Channel Id: 3 Seq: 21071 [trades] [ticker] [book]
163	2022-05-02 13:53:00,786…	Deribit SBE	3086696	3086695	13…	Channel Id: 3 Seq: 21072 [trades] [ticker] [book]
166	2022-05-02 13:53:00,836…	Deribit SBE	3086697	3086696	13…	Channel Id: 3 Seq: 21073 [trades] [ticker] [book]

Example message

```
Frame 220: 135 bytes on wire (1080 bits), 135 bytes captured (1080 bits) on
interface lo, id 0
Ethernet II, Src: 00:00:00_00:00:00 (00:00:00:00:00:00), Dst:
00:00:00_00:00:00 (00:00:00:00:00:00)
Internet Protocol Version 4, Src: 127.0.0.1, Dst: 239.111.111.3
User Datagram Protocol, Src Port: 51012, Dst Port: 6100
Deribit SBE
    Framing Header
        Packet Length: 85
        Channel Id: 3
        Channel Sequence: 21093
    Book Change
        Header
            Root Block Length: 29
            Type: book (1001)
            Schema Id: 1
            Version: 1
            Num Groups: 1
            Num Vars: 0
        Instrument Id: 136
        Timestamp: May  2, 2022 11:53:01.475000000 UTC
        Previous Change ID: 3086730
        Change ID: 3086733
        Is Last Part: last (1)
        Changes
            Group Header
                Group Block Length: 18
                Num In Group: 2
```

```
                Group Num Groups: 0
                Group Num Vars: 0
            Change List
                delete bid - price : 35171,99 amount : 0
                new bid - price : 36930,58 amount : 40

0000   00 00 00 00 00 00 00 00 00 00 00 00 08 00 45 00   ..............E.
0010   00 79 7b 98 40 00 20 11 01 68 7f 00 00 01 ef 6f   .y{.@. ..h.....o
0020   6f 03 c7 44 17 d4 00 65 dd ea 55 00 03 00 65 52   o..D...e..U...eR
0030   00 00 1d 00 e9 03 01 00 01 00 01 00 00 00 88 00   ................
0040   00 00 23 e3 9d 84 80 01 00 00 8a 19 2f 00 00 00   ..#........./...
0050   00 00 8d 19 2f 00 00 00 00 00 01 12 00 02 00 00   ..../...........
0060   00 00 00 01 02 e1 7a 14 ae 7f 2c e1 40 00 00 00   ......z...,.@...
0070   00 00 00 00 00 01 00 f6 28 5c 8f 52 08 e2 40 00   ........(\.R..@.
0080   00 00 00 00 00 44 40                              .....D@
```

C - Trades

In case an order results in one or more trades, these are sent in a trades event containing one or more trades related to the same instrument and order.

The meaning/purpose of the fields are the same as the corresponding websocket API event.

Message specification

```
<message name="trades" id="1002">
      <field name="header" id="1" type="messageHeader" />
      <field name="instrumentId" id="2" type="uint32" />
      <group name="tradesList" id="3" dimensionType="groupSizeEncoding">
            <field name="direction" id="1" type="direction" />
            <field name="price" id="2" type="double" />
            <field name="amount" id="3" type="double" />
            <field name="timestampMs" id="4" type="uint64" />
            <field name="markPrice" id="5" type="double" />
            <field name="indexPrice" id="6" type="double" />
            <field name="tradeSeq" id="7" type="uint64" />
            <field name="tradeId" id="8" type="uint64" />
            <field name="tickDirection" id="9" type="tickDirection" />
            <field name="liquidation" id="10" type="liquidation" />
            <field name="iv" id="11" type="double"
                  presence="optional" nullValue="0" />
            <field name="blockTradeId" id="12" type="uint64"
                  presence="optional" nullValue="0" />
            <field name="comboTradeId" id="13" type="uint64"
                  presence="optional" nullValue="0" />
      </group>
</message>
```

Enum types used in the message

```
<enum name="tickDirection" encodingType="uint8">
      <validValue name="plus">0</validValue>
      <validValue name="zeroplus">1</validValue>
      <validValue name="minus">2</validValue>
      <validValue name="zerominus">3</validValue>
</enum>

<enum name="direction" encodingType="uint8">
      <validValue name="buy">0</validValue>
      <validValue name="sell">1</validValue>
</enum>

<enum name="liquidation" encodingType="uint8">
      <validValue name="none">0</validValue>
      <validValue name="maker">1</validValue>
      <validValue name="taker">2</validValue>
      <validValue name="both">3</validValue>
</enum>
```

The event system combines the events related to the same transaction on an instrument. If these events fit in a single packet, the related `book`, `trades` and `ticker` messages will be combined in a packet.

Example message

```
Frame 181: 470 bytes on wire (3760 bits), 470 bytes captured (3760 bits) on
interface lo, id 0
Ethernet II, Src: 00:00:00_00:00:00 (00:00:00:00:00:00), Dst:
00:00:00_00:00:00 (00:00:00:00:00:00)
Internet Protocol Version 4, Src: 127.0.0.1, Dst: 239.111.111.1
User Datagram Protocol, Src Port: 55022, Dst Port: 6100
Deribit SBE
    Framing Header
        Packet Length: 420
        Channel Id: 1
        Channel Sequence: 10477
    Trades
        Header
            Root Block Length: 4
            Type: trades (1002)
            Schema Id: 1
            Version: 1
            Num Groups: 1
            Num Vars: 0
        Instrument Id: 1
        Trades
            Group Header
                Group Block Length: 83
                Num In Group: 2
```

```
            Group Num Groups: 0
            Group Num Vars: 0
        Trade list
            Trade
                Trade Direction: sell (1)
                Price: 39344,25
                Amount: 10
                Timestamp: May  2, 2022 11:53:01.000000000 UTC
                Mark Price: 38815,96
                Index Price: 38603,64
                Trade Sequence: 392167
                Trade Id: 1297362
                Tick Direction: zerominus (3)
                Liquidation: no liquidation (0)
                Implied Volatility: 0
                Block Trade Id: 0
                Combo Trade Id: 0
            Trade
                Trade Direction: sell (1)
                Price: 39316,72
                Amount: 10
                Timestamp: May  2, 2022 11:53:01.000000000 UTC
                Mark Price: 38815,96
                Index Price: 38603,64
                Trade Sequence: 392168
                Trade Id: 1297363
                Tick Direction: minus (2)
                Liquidation: no liquidation (0)
                Implied Volatility: 0
                Block Trade Id: 0
                Combo Trade Id: 0
Ticker
    Header
        Root Block Length: 133
        Type: ticker (1003)
        Schema Id: 1
        Version: 1
        Num Groups: 0
        Num Vars: 0
    Instrument Id: 1
    Instrument State: open (1)
    Timestamp: May  2, 2022 11:53:01.000000000 UTC
    Open Interest: 60
    Min Price: 38980,56
    Max Price: 39375,71
    Last Price: 39316,72
    Index Price: 38603,64
    Mark Price: 38815,96
    Best Bid Price: 39316,31
    Best Bid Amount: 30
    Best Ask Price: 39375,72
    Best Ask Amount: 10
    Current Funding: 0,005
```

```
        Funding 8h: 0,00751341
        Estimated Delivery Price: 38603,64
        Delivery Price: 0
        Settlement Price: 39229,82
    Book Change
        Header
            Root Block Length: 29
            Type: book (1001)
            Schema Id: 1
            Version: 1
            Num Groups: 1
            Num Vars: 0
        Instrument Id: 1
        Timestamp: May  2, 2022 11:53:01.000000000 UTC
        Previous Change ID: 3086708
        Change ID: 3086709
        Is Last Part: last (1)
        Changes
            Group Header
                Group Block Length: 18
                Num In Group: 2
                Group Num Groups: 0
                Group Num Vars: 0
            Change List
                delete bid - price : 39316,72 amount : 0
                delete bid - price : 39344,25 amount : 0
```

```
0000   00 00 00 00 00 00 00 00 00 00 00 00 08 00 45 00   ..............E.
0010   01 c8 bd 2f 40 00 20 11 be 83 7f 00 00 01 ef 6f   .../@. ........o
0020   6f 01 d6 ee 17 d4 01 b4 df 37 a4 01 01 00 ed 28   o........7.....(
0030   00 00 04 00 ea 03 01 00 01 00 01 00 00 00 01 00   ................
0040   00 00 53 00 02 00 00 00 00 00 01 00 00 00 00 08   ..S.............
0050   36 e3 40 00 00 00 00 00 00 24 40 48 e1 9d 84 80   6.@......$@H....
0060   01 00 00 85 eb 51 b8 fe f3 e2 40 ae 47 e1 7a 74   .....Q....@.G.zt
0070   d9 e2 40 e7 fb 05 00 00 00 00 00 d2 cb 13 00 00   ..@.............
0080   00 00 00 03 00 00 00 00 00 00 00 00 00 00 00 00   ................
0090   00 00 00 00 00 00 00 00 00 00 00 00 00 01 a4 70   ...............p
00a0   3d 0a 97 32 e3 40 00 00 00 00 00 00 24 40 48 e1   =..2.@......$@H.
00b0   9d 84 80 01 00 00 85 eb 51 b8 fe f3 e2 40 ae 47   ........Q....@.G
00c0   e1 7a 74 d9 e2 40 e8 fb 05 00 00 00 00 00 d3 cb   .zt..@..........
00d0   13 00 00 00 00 00 02 00 00 00 00 00 00 00 00 00   ................
00e0   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
00f0   85 00 eb 03 01 00 01 00 00 00 00 00 01 00 00 00   ................
0100   01 48 e1 9d 84 80 01 00 00 00 00 00 00 00 00 4e   .H.............N
0110   40 b8 1e 85 eb 91 08 e3 40 85 eb 51 b8 f6 39 e3   @.......@..Q..9.
0120   40 a4 70 3d 0a 97 32 e3 40 ae 47 e1 7a 74 d9 e2   @.p=..2.@.G.zt..
0130   40 85 eb 51 b8 fe f3 e2 40 b8 1e 85 eb 89 32 e3   @..Q....@.....2.
0140   40 00 00 00 00 00 00 3e 40 a4 70 3d 0a f7 39 e3   @......>@.p=..9.
0150   40 00 00 00 00 00 00 24 40 7b 14 ae 47 e1 7a 74   @......$@{..G.zt
0160   3f c2 f9 b3 a3 61 c6 7e 3f ae 47 e1 7a 74 d9 e2   ?....a.~?.G.zt..
0170   40 00 00 00 00 00 00 00 00 d7 a3 70 3d ba 27 e3   @..........p=.'.
0180   40 1d 00 e9 03 01 00 01 00 01 00 00 00 01 00 00   @...............
0190   00 48 e1 9d 84 80 01 00 00 74 19 2f 00 00 00 00   .H.......t./....
```

```
01a0   00 75 19 2f 00 00 00 00 00 01 12 00 02 00 00 00   .u./............
01b0   00 00 01 02 a4 70 3d 0a 97 32 e3 40 00 00 00 00   .....p=..2.@....
01c0   00 00 00 00 01 02 00 00 00 00 08 36 e3 40 00 00   ...........6.@..
01d0   00 00 00 00 00 00                                 ......
```

D - Ticker

The ticker event is sent periodically to indicate changes in the book data that are not strictly level (e.g. index price). The event can be related to a book level change (e.g. best bid/ask changes) in which case it is sent together with the corresponding book change. It can also be sent as a standalone event.

The ticker message structure is also used on the snapshot channels

Message specification

```
<message name="ticker" id="1003">
      <!-- according the the SBE spec, optional floats use
            the quietNaN null value 0xffffffffffffffff -->
      <field name="header" id="1" type="messageHeader" />
      <field name="instrumentId" id="2" type="uint32" />
      <field name="instrumentState" id="3" type="instrumentState" />
      <field name="timestampMs" id="4" type="uint64" />
      <field name="openInterest" id="5" type="double" />
      <field name="minSellPrice" id="6" type="double" />
      <field name="maxBuyPrice" id="7" type="double" />
      <field name="lastPrice" id="8" type="double"
            presence="optional"/>
      <field name="indexPrice" id="9" type="double" />
      <field name="markPrice" id="10" type="double" />
      <field name="bestBidPrice" id="11" type="double" />
      <field name="bestBidAmount" id="12" type="double" />
      <field name="bestAskPrice" id="13" type="double" />
      <field name="bestAskAmount" id="14" type="double" />
      <field name="currentFunding" id="15" type="double"
            presence="optional" />
      <field name="funding8h" id="16" type="double"
            presence="optional" />
      <field name="estimatedDeliveryPrice" id="17" type="double"
            presence="optional"/>
      <field name="deliveryPrice" id="18" type="double"
            presence="optional" />
      <field name="settlementPrice" id="19" type="double"
            presence="optional" />
</message>
```

Example message

```
Frame 106: 195 bytes on wire (1560 bits), 195 bytes captured (1560 bits) on
interface lo, id 0
Ethernet II, Src: 00:00:00_00:00:00 (00:00:00:00:00:00), Dst:
00:00:00_00:00:00 (00:00:00:00:00:00)
Internet Protocol Version 4, Src: 127.0.0.1, Dst: 239.111.111.4
User Datagram Protocol, Src Port: 37630, Dst Port: 6100
Deribit SBE
    Framing Header
        Packet Length: 145
        Channel Id: 4
        Channel Sequence: 4218
    Ticker
        Header
            Root Block Length: 133
            Type: ticker (1003)
            Schema Id: 1
            Version: 1
            Num Groups: 0
            Num Vars: 0
        Instrument Id: 2
        Instrument State: open (1)
        Timestamp: May  3, 2022 13:17:24.827000000 UTC
        Open Interest: 10
        Min Price: 2837,56
        Max Price: 2866,08
        Last Price: 2865,61
        Index Price: 2834,85
        Mark Price: 2850,44
        Best Bid Price: 2866,09
        Best Bid Amount: 20
        Best Ask Price: 2866,1
        Best Ask Amount: 10
        Current Funding: 0,004999
        Funding 8h: -0,003006
        Estimated Delivery Price: 2834,85
        Delivery Price: 0
        Settlement Price: 2837,27
0000   00 00 00 00 00 00 00 00 00 00 00 00 08 00 45 00   ..............E.
0010   00 b5 9e 37 40 00 20 11 de 8b 7f 00 00 01 ef 6f   ...7@. ........o
0020   6f 04 92 fe 17 d4 00 a1 de 27 91 00 04 00 7a 10   o........'....z.
0030   00 00 85 00 eb 03 01 00 01 00 00 00 00 00 02 00   ................
0040   00 00 01 db 81 11 8a 80 01 00 00 00 00 00 00 00   ................
0050   00 24 40 85 eb 51 b8 1e 2b a6 40 5c 8f c2 f5 28   .$@..Q..+.@\...(
0060   64 a6 40 1f 85 eb 51 38 63 a6 40 33 33 33 33 b3   d.@...Q8c.@3333.
0070   25 a6 40 7b 14 ae 47 e1 44 a6 40 48 e1 7a 14 2e   %.@{..G.D.@H.z..
0080   64 a6 40 00 00 00 00 00 00 34 40 33 33 33 33 33   d.@......4@33333
0090   64 a6 40 00 00 00 00 00 00 24 40 1c 09 34 d8 d4   d.@......$@..4..
00a0   79 74 3f 6c 07 23 f6 09 a0 68 bf 33 33 33 33 b3   yt?l.#...h.3333.
00b0   25 a6 40 00 00 00 00 00 00 00 00 d7 a3 70 3d 8a   %.@..........p=.
00c0   2a a6 40                                         *.@
```

E - Snapshots

Next to the events above, regular (each 1 minute by default) snapshots of order books are also multicasted. Snapshots are also grouped by currency/product (like events) and can be assigned to separate channels (see Deribit Multicast Channels document).

Snapshots are a sequence of `instrument/ticker/snapshot` messages that contain the static instrument data, the ticker information in the current state of the book, the order book levels, as well as the last change ID with the last change timestamp.

The book levels on both sides (asks/bids) are combined in a single list using a zigzag method (bid1, ask1, bid2, ask2 ….) .

Snapshots are generally complete, however if they have more than 10000 levels, they will be limited to 10000.

To indicate whether the snapshot is complete (includes all levels) or not, the message contains an `isComplete` flag.

If a complete snapshot does not fit in a packet, the level list is split (similarly to order book changes). The `isLast` flag indicates whether the message contains a part of the level list and the next message contains follow up (value = 0) or the message is the last in the sequence and the sending of the levels is done, or the message has all the levels sent in the snapshot (value = 1).

If multiple instrument/ticker/snapshot message sequences (for different instruments on a currency/product pair) fit in a single packet, they will be combined. Book level snapshot messages may be split across packets.

Message specification

```
<message name="snapshot" id="1004">
      <field name="header" id="1" type="messageHeader" />
      <field name="instrumentId" id="2" type="uint32" />
      <field name="timestampMs" id="3" type="uint64" />
      <field name="changeId" id="4" type="uint64" />
      <field name="isBookComplete" id="5" type="yesNo" />
      <field name="isLastInBook" id="6" type="yesNo" />
      <group name="levelsList" id="8" dimensionType="groupSizeEncoding">
            <field name="side" id="1" type="bookSide" />
            <field name="price" id="2" type="double" />
            <field name="amount" id="3" type="double" />
      </group>
</message>
```

Enum types used in the message

```
<enum name="yesNo" encodingType="uint8">
```

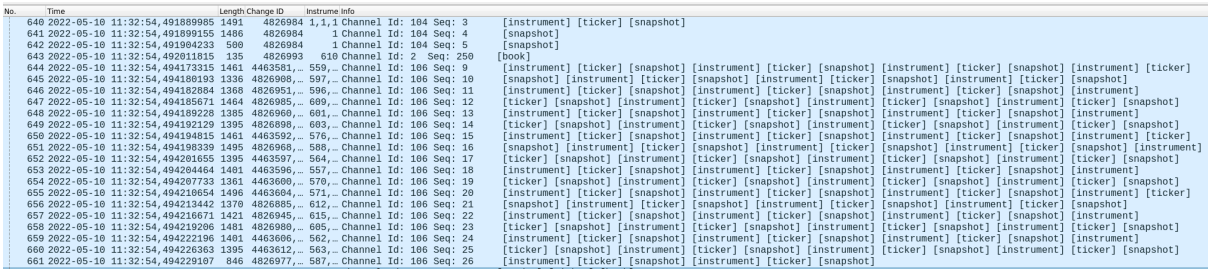
```
      <validValue name="no">0</validValue>
      <validValue name="yes">1</validValue>
</enum>

<enum name="bookSide" encodingType="uint8">
      <validValue name="ask">0</validValue>
      <validValue name="bid">1</validValue>
</enum>
```

Example message

```
Frame 13142: 347 bytes on wire (2776 bits), 347 bytes captured (2776 bits) on
interface lo, id 0
Ethernet II, Src: 00:00:00_00:00:00 (00:00:00:00:00:00), Dst:
00:00:00_00:00:00 (00:00:00:00:00:00)
Internet Protocol Version 4, Src: 127.0.0.1, Dst: 239.111.111.5
User Datagram Protocol, Src Port: 33646, Dst Port: 6101
Deribit SBE
    Framing Header
        Packet Length: 297
        Channel Id: 105
        Channel Sequence: 95
    Book Snapshot
        Header
            Root Block Length: 22
            Type: snapshot (1004)
            Schema Id: 1
            Version: 1
            Num Groups: 1
            Num Vars: 0
        Instrument Id: 486
        Timestamp: May  9, 2022 14:04:23.702000000 UTC
        Change ID: 4434022
        Is Book Complete: complete (1)
        Is Last Part: last (1)
        Levels
            Group Header
                Group Block Length: 17
                Num In Group: 15
                Group Num Groups: 0
                Group Num Vars: 0
            Level List
                bid - price : 32030,22 amount : 10
                bid - price : 32018,26 amount : 10
                bid - price : 32017,43 amount : 10
                bid - price : 32016,94 amount : 20
                bid - price : 32013,36 amount : 10
                bid - price : 32011,99 amount : 10
                bid - price : 31728,66 amount : 10
                bid - price : 31724,04 amount : 30
                bid - price : 31712,1 amount : 10
                bid - price : 31711,62 amount : 20
                bid - price : 31709,33 amount : 20
```

```
                bid - price : 31694,14 amount : 20
                bid - price : 31684,67 amount : 10
                bid - price : 31683,33 amount : 20
                bid - price : 31079,33 amount : 30
0000    00 00 00 00 00 00 00 00 00 00 00 00 08 00 45 00    ..............E.
0010    01 4d c1 01 40 00 20 11 bb 28 7f 00 00 01 ef 6f    .M..@. ..(.....o
0020    6f 05 83 6e 17 d5 01 39 de c0 29 01 69 00 5f 00    o..n...9..).i._.
0030    00 00 16 00 ec 03 01 00 01 00 01 00 00 00 e6 01    ................
0040    00 00 16 ad 22 a9 80 01 00 00 66 a8 43 00 00 00    ....".....f.C...
0050    00 00 01 01 11 00 0f 00 00 00 00 00 01 48 e1 7a    .............H.z
0060    14 8e 47 df 40 00 00 00 00 00 00 24 40 01 3d 0a    ..G.@......$@.=.
0070    d7 a3 90 44 df 40 00 00 00 00 00 00 24 40 01 52    ...D.@......$@.R
0080    b8 1e 85 5b 44 df 40 00 00 00 00 00 00 24 40 01    ...[D.@......$@.
0090    8f c2 f5 28 3c 44 df 40 00 00 00 00 00 00 34 40    ...(<D.@......4@
00a0    01 a4 70 3d 0a 57 43 df 40 00 00 00 00 00 00 24    ..p=.WC.@......$
00b0    40 01 c3 f5 28 5c ff 42 df 40 00 00 00 00 00 00    @...(\.B.@......
00c0    24 40 01 d7 a3 70 3d 2a fc de 40 00 00 00 00 00    $@...p=*..@.....
00d0    00 24 40 01 f6 28 5c 8f 02 fb de 40 00 00 00 00    .$@..(\....@....
00e0    00 00 3e 40 01 66 66 66 66 06 f8 de 40 00 00 00    ..>@.ffff...@...
00f0    00 00 00 24 40 01 e1 7a 14 ae e7 f7 de 40 00 00    ...$@..z.....@..
0100    00 00 00 00 34 40 01 ec 51 b8 1e 55 f7 de 40 00    ....4@..Q..U..@.
0110    00 00 00 00 00 34 40 01 5c 8f c2 f5 88 f3 de 40    .....4@.\......@
0120    00 00 00 00 00 00 34 40 01 14 ae 47 e1 2a f1 de    ......4@...G.*..
0130    40 00 00 00 00 00 00 00 24 40 01 ec 51 b8 1e d5 f0 @.......$@..Q....
0140    de 40 00 00 00 00 00 00 34 40 01 ec 51 b8 1e d5    .@......4@..Q...
0150    59 de 40 00 00 00 00 00 00 3e 40                   Y.@......>@
```

Example sequence of snapshots



No.	Time	Length	Change ID	Instrume	Info
640	2022-05-10 11:32:54,491889985	1491	4826984	1,1,1	Channel Id: 104 Seq: 3 [instrument] [ticker] [snapshot]
641	2022-05-10 11:32:54,491899155	1486	4826984	1	Channel Id: 104 Seq: 4 [snapshot]
642	2022-05-10 11:32:54,491904233	500	4826984	1	Channel Id: 104 Seq: 5 [snapshot]
643	2022-05-10 11:32:54,492011815	135	4826993	610	Channel Id: 2 Seq: 250 [book]
644	2022-05-10 11:32:54,494173315	1461	4463581,…	559,…	Channel Id: 106 Seq: 9 [instrument] [ticker] [snapshot] [instrument] [ticker] [snapshot] [instrument] [ticker] [snapshot] [instrument] [ticker]
645	2022-05-10 11:32:54,494180193	1336	4826908,…	597,…	Channel Id: 106 Seq: 10 [snapshot] [instrument] [ticker] [snapshot] [instrument] [ticker] [snapshot] [instrument] [ticker] [snapshot]
646	2022-05-10 11:32:54,494182884	1368	4826951,…	596,…	Channel Id: 106 Seq: 11 [instrument] [ticker] [snapshot] [instrument] [ticker] [snapshot] [instrument] [ticker] [snapshot] [instrument]
647	2022-05-10 11:32:54,494185671	1464	4826985,…	609,…	Channel Id: 106 Seq: 12 [ticker] [snapshot] [instrument] [ticker] [snapshot] [instrument] [ticker] [snapshot] [instrument] [ticker] [snapshot]
648	2022-05-10 11:32:54,494189228	1385	4826960,…	601,…	Channel Id: 106 Seq: 13 [instrument] [ticker] [snapshot] [instrument] [ticker] [snapshot] [instrument] [ticker] [snapshot] [instrument]
649	2022-05-10 11:32:54,494192129	1395	4826898,…	603,…	Channel Id: 106 Seq: 14 [ticker] [snapshot] [instrument] [ticker] [snapshot] [instrument] [ticker] [snapshot] [instrument] [ticker] [snapshot]
650	2022-05-10 11:32:54,494194815	1461	4463592,…	576,…	Channel Id: 106 Seq: 15 [instrument] [ticker] [snapshot] [instrument] [ticker] [snapshot] [instrument] [ticker] [snapshot] [instrument] [ticker]
651	2022-05-10 11:32:54,494198339	1495	4826968,…	588,…	Channel Id: 106 Seq: 16 [snapshot] [instrument] [ticker] [snapshot] [instrument] [ticker] [snapshot] [instrument] [ticker] [snapshot] [instrument]
652	2022-05-10 11:32:54,494201655	1395	4463597,…	564,…	Channel Id: 106 Seq: 17 [ticker] [snapshot] [instrument] [ticker] [snapshot] [instrument] [ticker] [snapshot] [instrument] [ticker] [snapshot]
653	2022-05-10 11:32:54,494204464	1401	4463596,…	557,…	Channel Id: 106 Seq: 18 [instrument] [ticker] [snapshot] [instrument] [ticker] [snapshot] [instrument] [ticker] [snapshot] [instrument]
654	2022-05-10 11:32:54,494207733	1361	4463600,…	570,…	Channel Id: 106 Seq: 19 [ticker] [snapshot] [instrument] [ticker] [snapshot] [instrument] [ticker] [snapshot] [instrument] [ticker] [snapshot]
655	2022-05-10 11:32:54,494210654	1496	4463604,…	571,…	Channel Id: 106 Seq: 20 [instrument] [ticker] [snapshot] [instrument] [ticker] [snapshot] [instrument] [ticker] [snapshot] [instrument] [ticker]
656	2022-05-10 11:32:54,494213442	1370	4826885,…	612,…	Channel Id: 106 Seq: 21 [snapshot] [instrument] [ticker] [snapshot] [instrument] [ticker] [snapshot] [instrument] [ticker] [snapshot]
657	2022-05-10 11:32:54,494216671	1421	4826945,…	615,…	Channel Id: 106 Seq: 22 [instrument] [ticker] [snapshot] [instrument] [ticker] [snapshot] [instrument] [ticker] [snapshot] [instrument]
658	2022-05-10 11:32:54,494219206	1481	4826980,…	605,…	Channel Id: 106 Seq: 23 [ticker] [snapshot] [instrument] [ticker] [snapshot] [instrument] [ticker] [snapshot] [instrument] [ticker] [snapshot]
659	2022-05-10 11:32:54,494222196	1401	4463606,…	562,…	Channel Id: 106 Seq: 24 [instrument] [ticker] [snapshot] [instrument] [ticker] [snapshot] [instrument] [ticker] [snapshot] [instrument]
660	2022-05-10 11:32:54,494226363	1395	4463612,…	563,…	Channel Id: 106 Seq: 25 [ticker] [snapshot] [instrument] [ticker] [snapshot] [instrument] [ticker] [snapshot] [instrument] [ticker] [snapshot]
661	2022-05-10 11:32:54,494229107	846	4826977,…	587,…	Channel Id: 106 Seq: 26 [instrument] [ticker] [snapshot] [instrument] [ticker] [snapshot]

4 Basic mechanisms

A - Client start using the API

When a client starts up, it should start listening to multicast events and queue the received messages to make sure that it does not miss changes while building initial data with the following steps.

1. Retrieve the current complete instrument dictionary for all the open instruments with the new API call `multicast/get_instrument_dictionary`

The result of the call is a mapping between instrument ID and Instrument name
e.g. :

```
{"jsonrpc":"2.0","result":{"ETH-PERPETUAL":2,"BTC-PERPETUAL":3,"BTC-15APR22":
4}}
```

The mapping can also be retrieved for the given currency/product from the regular API using the `public/get_instruments` call.
The instrument data now contains the numeric instrument ID. The client can use this to build an instrument ID/Name dictionary.

2. Retrieve the status of the order books for each instrument using the `public/get_order_book` (or `public/get_order_book_by_instrument_id`) API call.

Apply the eventually missed order book changes, based on the change ID in the retrieved books and in the received multicast events.

B - Client start using snapshot multicasts

The client can also build the initial state of the books based on the regular snapshots. See details of the snapshot messages (Instrument/ticker/snapshot) are described above.

When using snapshots, it is still recommended to queue changes, as they may happen while the platform builds/sends the snapshots. From the queued book level changes, if there are any with higher Change ID than the one indicated in the snapshot, they should be applied. Obviously changes before the snapshot Change ID are not relevant and should be discarded.

Since snapshots contain both the instrument ID and Instrument name, as well as other static instrument data, probably there is no need to retrieve the dictionary from the API.

Note that while the `multicast/get_instrument_dictionary` API call contains only the list of active instruments, the snapshot stream includes closed/settled instruments until they are archived (usually after a day).

When joining a snapshot channel in the middle of a snapshot batch (i.e. packets received immediately when joining), it is recommended to ignore those and wait for the next batch, to make sure that a complete snapshot batch, with all instruments on the channel is received/processed.

C - New instruments or closed instruments

When a new instrument is added to the system. An `instrument` event (see above) is generated which contains the ID and the name of the new instrument. The new book will also be part of the snapshots.

When instruments are closed/settled there are also `instrument` events generated that can be used by the client to stop maintaining the book.

D - Channel packet recovery

A client can detect a missed packet, based on a jump in sequence numbers. Note that the sequence number is a 32 bit integer and it will turn around to 0 once in a while, which the clients should be able to recognize and not treat it as an error.
Sequence numbers can also reset to 0 after a system maintenance which may require the clients to rebuild their initial data (e.g. as described above).

D.1 API call

When the client detects missed packets, it can retrieve them using the `multicast/get_packets` call.

The parameters of the call

channel_id	The integer ID of the multicast channel where the packet was missed
start	Start sequence number
end	End sequence number

The response contains the list of requested packets excluding the framing header (only the SBE messages) in base64 encoded binary format, that can be decoded the same way as a multicast packet (after base64 decoding).

Example

Request

```
api/v2/multicast/get_packets?channel_id=15&end=2&start=1
```

Response

```
{
	"jsonrpc":"2.0",
	"result":[
		{
			"seq":1,
			"packet":"6QMBAAEAGQABAAAAAQAAABuxRACAAQAAAAAAAAAAAAAFAAAA
		AAAAAAESAAEAAAAAAAAAzczMzBxy5kAAAAAAAAAkQOsDAQABACgAAAAAAAEAAAAb
		sUQAgAEAAAAAAAAAAAAAAAAAAAAAAAADNzMzMHHLmQAAAAAAAACRA"
		},
		{
			"seq":2,
			"packet":"6QMBAAEAGQABAAAAAQAAAIWxRACAAQAABQAAAAAAAAAHAAAA
		AAAAAAESAAEAAAAAAAAEA7FG4Hr185UAAAAAAAAA0QOsDAQABACgAAAAAAAEAAACF
		sUQAgAEAAOxRuB69fOVAAAAAAAAAANEDNzMzMHHLmQAAAAAAAACRA"
		}
	],
	"usIn":1649273556304472,
	"usOut":1649273556304731,
	"usDiff":259,
	"testnet":false
}
```

5 Developer information

Next to this document, the following files are provided to aid client development.

A - Sample captures

These are taken from a development test setup and can be used as a reference to inspect with Wireshark (using the plugin we created), or replay packets with tools like tcpreplay.

Replay sample capture

tcpreplay is available on most linux distributions as a package. After installation the pcap file can be replayed e.g.

```
sudo tcpreplay -p 1 -i lo sample_capture_v1_4_2.pcapng.gz
```

With this command the packets in the capture file are replayed on the local (loop) interface with 1 packet per second speed. The client should also listen on the event/snapshot channel ports (6100, 6101) and join the multicast group address(es) in the capture.

sample_capture_v1_4_2.pcapng	multicast snapshot and message samples

B - Wireshark dissector plugin

A LUA wireshark dissector plugin **deribit_sbe.lua** has been created that allows Wireshark packet capture/analysis tool to display the content of the multicast packets.

The dissector has been tested with Wireshark version 3.4.8 on linux (ubuntu 20.04).

To add the dissector to Wireshark, copy the file to the directory of **'Personal LUA plugins'** or **'Global LUA plugins'** which can be found in Wireshark under **'Help' -> 'About Wireshark' -> 'Folders'**.

The loading of the plugin can be verified by clicking on the **'Plugins'** tab in the same window.

After the plugin is loaded, and the packets are not decoded, then Wireshark should be told to assign the decoder to the port numbers. **Right click on any of the packets** and select **'Decode As…'** from the context menu. In the dialog select the UDP destination port number (e.g. 6100) and assign the **'DERIBIT SBE'** decoder to it.

The LUA file can also be used as a sample for understanding the message decoding.

C - SBE XML definition

The **deribit_multicast.xml** XML file contains the definitions of messages in the SBE XML format (see SBE specification) and can be used as a reference for creating the decoder, or as input for the code generation for different languages.

D - Code generation tools

The tool that can be used for code generation (called **sbe-all-1.25.1.jar** requires java to be installed) is created by Real Logic. It is only bundled for convenience, and it is not supported by us.
For any inquiry (not related to eventual issues with the XML definition) or issues, the project github repository is the best resource (https://github.com/real-logic/simple-binary-encoding).
The tool can generate library code from the XML specification in different programming languages (c, c++, java, go, rust). The tool is open source under the Apache 2.0 license.

Please note that XML definition is only provided to assist client development, it is not used in the platform (and not extensively tested) so feedback about any issues found in XML is welcome.
Sample commands are provided as scripts (e.g. **generate_c.sh**) as an example to generate libraries.

6 Change history

Date	Version	Summary
6 April 2022	1.1	Initial published
11 April 2022	1.2	Change of message header structure (blockLength moved to the top of the header). Add a new API call to retrieve the instrument dictionary. Improve message descriptions with types and sample messages. Updated XML spec, sample trace and LUA dissector
13 April 2022	1.3	Fix the instrumentId field in the XML specification, update generated code libraries
2 May 2022	1.4	- Introduce Ticker message - Replace Quote event with Ticker - Extend the Instrument message with static instrument data - Include the Instrument message in the snapshots - Remove instrument name from snapshot

		message (included in the instrument message) - Include the Ticker message in the snapshots - Change the order of messages in the packet when a book changes (Trades/Ticker/Book) - Improve packet filling (e.g. large snapshot messages that are split to multiple packet don't start from a new packet, but fill the space after Instrument/Ticker messages) - Provide sample trace with production like multicast channel configuration - Update the wireshark LUA dissector with the changes in the messages
13 May 2022	1.4.1	- Fix root block length of the instrument message sample and provide new sample trace
23 May 2022	1.4.2	- Fix XML definition of instrumentKind enum (XML change) - Fix XML snippet in the document for float nullValues in the ticker message (doc only change) - Add note to snapshots about closed/settled instruments (doc only change) - Add comment about mid batch joining of snapshot channel (doc only change) - Change name of pcap sample file to match the doc version to avoid confusion