SPB EXCHANGE

Native Protocol Market Data Service

Interface version 40

Document version 1.20.0

23 January 2024

Revision history

Version 1.20.0 December 23, 2024 года

1. Added the description of the TheorPrices topic in Section 3.7.
2. Added Option and OptionSeries messages in Section 3.8.
3. Added the price componen in Section 2.3 Common components of messages.
4. The Offset field is added to components of messages in Section 2.3.
5. Deleted the Futures message, because the balance instrument is not currently in use.

Version 1.19.5 October 27, 2023 года

Updated the description of Section 3.6 Commons topic.

Version 1.19.4 July 4, 2023 года

Added Section 2.8 Volume dimension.

Version 1.19.3 December 9, 2022 года

Changed the company logo.

Version 1.19.2 December 6, 2021 года

1. Updated the list of liquidity pools in Section 2.7 Liquidity pool identifiers.
2. The BondAccruedInterest message is added to Section 3.8 Instruments topic.
3. Clarified description of the topic_firstseq field in the TopicReject message in Section 4.1.11 Report on rejecting request.

Version 1.19.1 October 6, 2021

1. Added security type "Exchange traded bond" to the security_type field in the Bond message.
2. Added the table of correspondence between designations of instrument categories in the trading system and Interfax.

Version 1.19.0 August 4, 2021

1. The category field is added to the Instrument message in Section 3.8 Instruments topic.
2. Added error code 1405 in Appendix A.
3. Updated the list of liquidity pools in Section 2.7 Liquidity pool identifiers.

Version 1.18.2 March 2, 2021

1. Improved description in Section 3.4 CurrentPriceOfMarket topic.
2. Renamed the Trade message to Indiquote.

Version 1.18.1 November 23, 2020

Added the EmptyBook message in Section 3.5 BestPrices topic.

Version 1.18.0 September 2, 2020

1. The sub_aggr component is renamed to sub_dom. The yield field is added to the sub_dom component.
2. The AggrMsgOnline message is renamed to DomOnline.
3. The AggrMsgSnapshot message is renamed to DomSnapshot.
4. The aggr_entry field is added to messages DomOnline and DomSnapshot.
5. The aggr_offset field datatype in messages DomOnline and DomSnapshot is changed.
6. The yield field is added to the Trade message in the Trades topic.
7. The yield field is added to the Trade message in the CurrentPriceOfMarket topic.
8. The HIGH_LIQUIDITY flag is added to the Trade message in the CurrentPriceOfMarket topic.
9. The yield_close and yield_last parameters are added to the Commons topic.

Version 1.17.7 July 24, 2020

Parameter codes are added to Section 3.6 Commons topic in the tables 22 and 23.

Version 1.17.6 June 2, 2020

Added warning about inexpedience using the parameters of a specific trade mode for setting up the trade system to the description of TradeModes message.

Version 1.17.5 April 17, 2020

1. Description about the trading system's behavior in case a client uses other, than standard method of datafeeds processing was added to Section 1.4.
2. The allowed range of values for topic_seq and topic_seqend parameter has been changed in Section 4.1.9.

Version 1.17.4 January 24, 2020

Added Section 4.1.4, describing client's send rate limit.

Version 1.17.3 July 25, 2019

Section 4.1.3 is renamed to "Keeping section in active state". Description of active session state maintenance is updated.

Version 1.17.2 February 1, 2019

Added value 4 (MemberTariff) to the fee_schema field of the Instrument message.

Version 1.17.1 December 14, 2018

1. Document structure was changed.
2. Terminology related to topic data transmission was changed.
3. Names of messages of OrderBook, Trades, CurrentPriceOfMarket, BestPrices and Commons topics are specified.
4. Description of key fields in topic messages was added.
5. The Heartbeat (msgid=15236) message was renamed to MdHeartbeat.
6. The topic_header component was renamed to header.
7. The addresses component was renamed to Report_Address.
8. The CommonEntry component was renamed to CommonsUpdateEntry.
9. The PriceLevel component was renamed to sub_aggr.
10. The BestPrice component was renamed to sub_best.

Version 1.17.0 November 3, 2017

1. The BorrowingStatus message has been added to the Instruments topic.
2. The msgid field value changed for the TradeModes message.
3. The over_the_counter field added to the TradeModes message.
4. The msgid field value changed for the Instrument message.
5. The borrowing_status field added to the Instrument message.
6. The trading_status field of the TradingInstrumentStatus message renamed to status.
7. The list of parameters in the Commons topic, unavailable for over-the-counter instruments, has been added.
8. Terminology changes.
9. Error codes added.

Version 1.16.0 November 30, 2016

1. The markets field added to the Period component.
2. The msgid value changed in the Instrument message.

Version 1.15.0 23 March 2016

The Market message is added to the Instruments topic.

Version 1.14.0 9 March 2016

1. The message broadcast in the Commons snapshot is changed.
2. The Commons topic of additional snapshot is removed.
3. New values type=73 and 75 added in the Commons message.

Table of Contents

List of Tables

1. Service overview

1.1. Data topics

The gateway currently provides the following topics:

1. OrderBook is a list of buy and sell orders for a specific instrument, grouped by price level.The number of levels is 50.
2. Trades is a list of public trades matched at the liquidity pools during the current trading day.
3. CurrentPriceOfMarket is a current market price, changing by the trade and the best order.
4. BestPrices is the top of an instrument order book—the highest bid and the lowest ask.
5. Commons is a list of statistic data of the accessed liquidity pools.
6. Instruments is instrument reference information.

Connection parameters are listed in the document *Network Connectivity*.

Messages of each topic are numbered consecutively in the `seq` field.

1.2. Broadcast modes

Topics can broadcast data in two modes — **snapshot** and/or **updates**.

A snapshot is aggregation of all current data, e.g. a whole order book, transmitted at a specified frequency.

Updates are separate messages generated and transmitted to the client when an event occurs.

During a period of inactivity in an update feed the system sends a `MdHeartbeat` to acknowledge connection. If messages are not transmitted for a longer period, there is either a transmission delay or absence of connection.

1.3. Broadcast channels

Each market data topic is broadcast through two identical UDP channels — *A* and *B*. Both channels simultaneously transfer messages with the same numbers. The channel duplication provides more transmission fidelity and lowers the probability of package loss. The client is strongly recommended to process both channels. For example, if a client receives *n*+1 message after *n*–1 message in *A* channel, then *n* message will be probably found in *B* channel. If a package is lost in the both channels, a client should either wait the next snapshot or request the message via recovery gateway.

1.4. Algorithm of receiving and processing topic data

If you want to connect to a topic with snapshots and updates, it is recommended to connect in both modes. First, you should receive a complete snapshot, then start recording incoming updates. You are recommended to record messages from both UDP-channels (A and B) and sort them by number. If an update has been lost in one of the channels, it can be requested in recovery gateway (messages, missing from the snapshots, cannot be recovered) . If messages' recovery takes significant amount of time, it is recommended to request the snapshot instead of attempting to recover lost updates.

When snapshot is complete you should record the updates. Updates can replace or replenish earlier data, depending on the topic. For topics with replacement there are identifiers of updated data - `keys`. The `keys` are fields values of topic messages and are indicated in header of tables in section 3.

Table 1. Features of snapshot and updates

Topic	Update		Snapshot
	Replenishment	Replacement	
Trades	✓		Messages history since the start of the trading day
OrderBook CurrentPriceOfMarket BestPrices Commons	✓	✓	An aggregation of all current data
Instruments		✓	

1.4.1. Example for `OrderBook` topic

The trade system may deny execution of client's request with an error message if the client's requests do not follow the algorithm specified below.

Updates from `OrderBook` topic **replace** earlier data.

1. Connect to the updates mode of the required topic and save all incoming `DomOnline` messages.
2. Connect to the snapshot mode of the topic and wait for `SnapshotStarted`.
3. Save all incoming `DomSnapshot` messages, until `SnapshotFinished` is received.
   - If some `DomSnapshot` messages have been skipped, or `update_seq` values are different for `SnapshotStarted` and `SnapshotFinished` messages, then repeat actions **2** and **3**.
   - If there is no saved `DomOnline` message with the number equal to `update_seq+1` , then repeat actions **2** and **3**.
4. Compare keys values of `DomSnapshot` and each `DomOnline` with number `seq>update_seq` (the keys of `OrderBook` topic are `instrument` and `source_id`):
   - If the values are equal, you should replace the snapshot message with `DomOnline`.
   - If the values aren't equal, you should replenish the snapshot with `DomOnline`.

1.4.2. Example for `Trades` topic

The trade system may deny execution of client's request with an error message if the client's requests do not follow the algorithm specified below.

For example the client connect to `Trades` topic during the trade day (sequential numbers of messages are reset every night). Number of the last `Trade` message is 105.

Updates from `Trades` **replenish** earlier data.

1. Connect to `Trades` topic and wait for first trade message. If `Trade` message is not available withing 5 seconds after the connection, then the `MdHeartbeat` message is received.
2. Obtain sequential number `seq` of the first received message (for example, `MdHeartbeat` message with `seq=305`). You can determine, that messages from `seq=106` to `seq=304`, were not received.
3. To recover messages you schould connect to the gateway and send `TopicRequest` with the `topic=Trades`, `topic_seq=106`, and `topic_seqend=304`.
4. The `TopicRequest` will result in the following message sequence:
   - TopicReport (seq=0, Start);
   - Trade (seq=1, topic_seq=150);
   - Trade (seq=2, topic_seq=170);
   - Trade (seq=3, topic_seq=200);

- Trade (seq=4, topic_seq=303);
- TopicReport (seq=0, End).

The broadcast of recovered data is surrounded by the `TopicReport` messages. The `Trade` messages have gaps between `topic_seq` values, because the `Heartbeat` messages were received between `Trade` messages.

2. Protocol overview

2.1. Data types

The trading system uses little-endian byte order (same as in x86 processor); the client shall use same.

`asciiN` is an alphanumeric string of *N*-byte length; the unused part should be filled with zero bytes.

`charN+1` is a UFT-8 encoded string of *N+1*-byte length. The last byte is the end of line character and so the available length is *N*; the unused part should be filled with zero bytes.

`dec2` is an eight-byte integer representing a fraction multiplied by 10^2.

`dec8` is an eight-byte integer representing a fraction multiplied by 10^8.

`decn` is a nine-byte sequence; the first eight bytes are an integer representing a fraction multiplied by 10^n and the last byte is *n*.

`intN` is an *N*-byte integer.

`time4` is a four-byte integer representing the Unix time in seconds, i.e. the number of seconds since 1 January 1970.

`time8n` is an eight-byte integer representing the Unix time in nanoseconds, i.e. the number of nanoseconds since 1 January 1970.

`time8m` is an eight-byte integer representing the Unix time in milliseconds, i.e. the number of milliseconds since 1 January 1970. If a field of this datatype conveys a date, the value part representing hours, minutes, seconds and milliseconds should be neglected, i.e. that is to use an integer value (rounded down) of division by 86 400 000.

2.2. Message format

A native protocol message is a sequence of field values in a strict order. Each message starts with the `frame` header; this three-field component includes message size, message type, and sequence number. The message size is the length of the whole message, except for the frame header, in bytes. The size is constant for all message types which don not include any repeating component or field.

A message is transmitted in a network packet as a sequence of bytes.

2.3. Common components of messages

Table 2. Format of component `frame`: size 12 bytes

Offset	Field	Datatype	Description
0	size	int2	Message length in bytes, excluding the `frame` header
2	msgid	int2	Message type
4	seq	int8	Application message sequence number

Common components for topics

Table 3. Format of component `instrument`: size 6 bytes

Offset	Field	Datatype	Description
0	market_id	int2	Liquidity pool ID. For a description of values, refer to Section 2.7
2	instrument_id	int4	Trading instrument ID

Table 4. Format of component `price`: size 22 bytes

Offset	Field	Datatype	Description
0	price	dec8	The price value of the corresponding type specified in the `type` field
8	type	int1	Price type. Values: • `123` (BasePrice): base price; • `124` (Price): theoretical price
9	reserved1	int1	Reserved field. To be filled with null byte
10	reserved2	int4	Reserved field. To be filled with null byte
14	time	time8n	Timestamp of price generation

Table 5. Format of component `md_header`: size 10 bytes

Offset	Field	Datatype	Description
0	system_time	time8n	Timestamp of message generation
8	source_id	int2	Message source (for values please refer to section 2.6)

Common components for recovery gateway

Table 6. Format of component `header`: size 22 bytes

Offset	Field	Datatype	Description
0	topic_id	int4	Numerical ID of topic
4	topic_seq	int8	Message sequence number in topic
12	system_time	time8n	Message generation time
20	source_id	int2	Message source (for values please refer to section 2.6)

Table 7. Format of component `user_header`: size 20 bytes

Offset	Field	Datatype	Description
0	clorder_id	ascii20	Client order ID

Table 8. Format of component `gate_header`: size 46 bytes

Offset	Field	Datatype	Description
0	system_time	time8n	Client request processing time
8	source_id	int2	Message source (for values please refer to section 2.6)
10	clorder_id	ascii20	Client order ID
30	user_id	ascii16	Login, client gateway ID

2.4. Recovered message format

The format of a recovered message is identical to that of a broadcast message, except for the header — the `header` stands for the `md_header`. Therefore, offsets of all following fileds are increased by 12 bytes. This is due to the recovered message are transmitted via TCP, not UDP.

Broadcast message format

Offset	Field	Datatype	Description
	[frame]	[frame]	Session header
0	[md_header]	[md_header]	Market data header
10	[instrument]	[instrument]	Trading instrument

Recovered message format

Offset	Field	Datatype	Description
	[frame]	[frame]	Session header
0	[header]	[header]	Header
22	[instrument]	[instrument]	Trading instrument

Figure 1. Message format alternation

2.5. Repetitive components and fields

Several message types contain one or more repeating components or fields which may have an arbitrary number of entries. One message may include multiple repetitive components and fields. All same-type repetitive components has a constant length.

A repeating component or field is always preceded by the two fields — `offset` and `count`. The `count` field specifies the number of entries. The `offset` field indicates an offset in bytes of first entry from the beginning of this very field; its value is no less than 4.

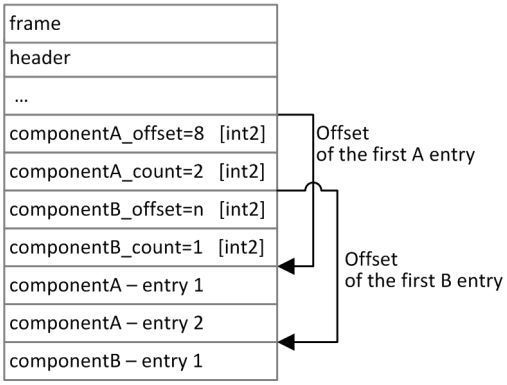

Figure 2. Template of a message with two repeating components

A repeating component may include another repeating component or field. In this case each entry refers to its own set of the embedded entries.

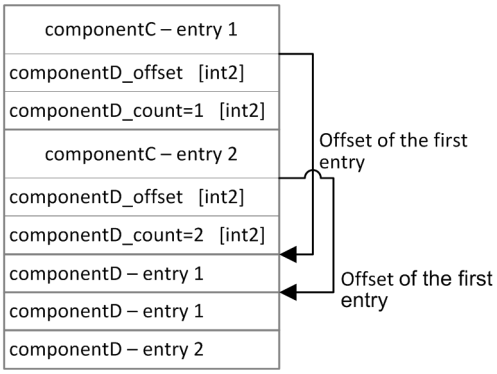

Figure 3. Template of embedded components

2.6. `Source_id` values

Field `source_id` is in the headers gate_header , md_header and header; the field specifies the module transmitting message to gateway for sending it to client.

Table 9. `Source_id` values to be returned to client

Range	Description
100–199	Trading system gateway
200–249	Clearing House risk parameter verification modules
250–259	Matching modules
300–499	Modules of generation and calculation of market data
500–549	Routing modules

2.7. Liquidity pool identifiers

Liquidity pools' identifiers may be in fields `markets`, `market_id` and `source_id` .

`0` (DEFAULT) — liquidity pool is defined by the trading system.

`1001` (TRADSYS) — all available liquidity pools

`1000` (SPB) — liquidity pool of SPB Exchange

`1010` (MOEX_FOND) — liquidity pool of Moscow Exchange

`1015` (IB) — execution at United States liquidity pools

`1017` (LSE) — liquidity pool of LSE

`1019` (SEHK) — liquidity pool of SEHK

`1021` (XETRA) — liquidity pool of german equities

2.8. Volume dimension

Regardless of the direction of routing, the volume of any order is indicated in the internal lots of the trading system.

The size of the internal lot in the trading system may differ from the size of the lot in the external system for the same trading instrument.

The lot size of a trading instrument depends on the period and is specified in the corresponding balance instrument .

3. Market data messages

3.1. Snapshot start and finish

For all channels, a snapshot is preceded by a `SnaphotStarted` and followed by a `SnapshotFinished`. The both messages contains the `updates_seq` field that conveys the sequence number of the last update message involved in the snapshot. Therefore, the update messages to be applied to the snapshot have a `seq` grater than the `updates_seq`.

Table 10. Format of message `SnapshotStarted`: msgid=12345, size=18

Offset	Field	Datatype	Description
	[frame]	[frame]	Session header
0	[md_header]	[md_header]	Header
10	update_seq	int8	Sequence number of the last update included in the snapshot

Table 11. Format of message `SnapshotFinished`: msgid=12312, size=18

Offset	Field	Datatype	Description
	[frame]	[frame]	Session header
0	[md_header]	[md_header]	Header
10	update_seq	int8	Sequence number of the last update included in the snapshot

3.2. OrderBook topic

*Snapshot is aggregation of all current data. Updates **replenish** and/or **replace** earlier data.*

The OrderBook snapshot conveys 50 or less price levels; the updates concern 50 disclosed price levels only.

An OrderBook message concerns the order book of an instrument which is specified in the `instrument` component.

The updates are transmitted via the `DomOnline` messages, the snapshots are transmitted via the `DomSnapshot` messages.

The final part of an OrderBook message is the `sub_dom` repeating component with the number of entries specified in the `PriceLevel_count` field (for more information on processing of repeating component, refer to Section 2.5). A component entry includes price level, orders direction, add/update indicator, total disclosed amount of orders at the price level (in lots), and latest update timestamp.

The value of the `flag` field indicates whether the price level is added or updated; and a price level removal will be described as amount update to zero. In a snapshot, price levels are defined as new.

Decoding should use the size of the `aggr_entry` field from the message. Don't use the `aggr_entry` field size from the schema to avoid format extension problems in the future.

Table 12. Format of message `DomOnline`: msgid=1120, dynamic length, keys=instrument

Offset	Field	Datatype	Description
	[frame]	[frame]	Session header
0	[md_header]	[md_header]	Header
10	[instrument]	[instrument]	Component specifying trading instrument

Offset	Field	Datatype	Description
16	aggr_offset	int4	Offset of the first `aggr` entry from the beggining of this field
20	aggr_count	int2	Number of the `aggr` group entries
22	aggr_entry	int2	Size of the group `aggr`
	> aggr	[sub_dom]	List of price levels

Table 13. Format of message `DomSnapshot`: msgid=1121, dynamic length, keys=instrument

Offset	Field	Datatype	Description
	[frame]	[frame]	Session header
0	[md_header]	[md_header]	Header
10	[instrument]	[instrument]	Component specifying trading instrument
16	aggr_offset	int4	Offset of the first `aggr` entry from the beggining of this field
20	aggr_count	int2	Number of the `aggr` group entries
22	aggr_entry	int2	Size of the group `aggr`
	> aggr	[sub_dom]	List of price levels

Table 14. Format of component `sub_dom`: size 30 bytes

Offset	Field	Datatype	Description
0	price	dec8	Price
8	yield	dec8	Yield
16	type	int1	Sides of orders. Values: • `1` (BUY_DIR): buy; • `2` (SELL_DIR): sell; • `3` (LAST_DEAL): last trade
17	flag	int1	Flag of new entry. Values: • `0x0` (UPDATE): updating; • `0x1` (NEW): adding
18	amount	int4	Total volume at the price level, in lots
22	time	time8n	Time of recent price level change

After a trading system restart, the gateway sends the `EmptyBook` message to the OrderBook topic for an order book clearing.

Table 15. Format of message `EmptyBook`: msgid=15300, size=16

Offset	Field	Datatype	Description
	[frame]	[frame]	Session header

Offset	Field	Datatype	Description
0	[md_header]	[md_header]	Header
10	instrument	[instrument]	Component specifying trading instrument

3.3. Trades topic

*Snapshot is an entire message history since the start of the trading day. Updates **replenish** earlier data.*

Upon a trade execution, the trading system generates a `Trade` message containing trade parameters with the liquidity pool of execution in the `market` field of the `instrument` component, with an unique trade identifier `trade_id`, with the trade volume `amount`, with the trade price `price`, with the transaction timestamp `trade_time`, and with taker's order side `dir`.

Table 16. Format of message `Trade`: msgid=19306, size=70, keys=instrument, trade_id

Offset	Field	Datatype	Description
	[frame]	[frame]	Session header
0	[md_header]	[md_header]	Header
10	instrument	[instrument]	Component specifying trading instrument
16	trade_id	int8	Trade ID assigned by liquidity pool
24	amount	int4	Trade volume in lots
28	price	dec8	Trade price
36	trade_time	time8n	Trade time
44	trade_type	int1	Trade type. Value: `1` (REGULAR): regular trade
45	dir	int1	Initiator"s order side. Values: • `1` (Buy): buy; • `2` (Sell): sell
46	pad0	dec8	Additional price
54	flags	int8	Deals flags
62	yield	dec8	Trade yield

3.4. CurrentPriceOfMarket topic

*Snapshot is aggregation of all current data. Updates **replenish** and/or **replace** earlier data.*

The current price of market is continuously calculated, based on trade prices and hard quotes according to the following rules:

1. If a trade is made, the current price of market becomes equal to the trade price.
2. If an anonymous buy order appears in the order book, and its price is higher than the current price of market, the current price of market becomes equal to the buy order price.
3. If an anonymous sell order appears in the order book, and its price is lower than the current price of market, the current price of market becomes equal to the sell order price.

When the current price of market changes, an `Indiquote` message is created. The message includes the new value of the current price of market `price`, time of making a trade or placing an order that caused the current price of market to be changed `trade_time` and a direction of this order `dir`.

Table 17. Format of message `Indiquote`: msgid=15411, size=70, keys=source_id, instrument, trade_id

Offset	Field	Datatype	Description
	[frame]	[frame]	Session header
0	[md_header]	[md_header]	Header
10	instrument	[instrument]	Component specifying trading instrument
16	trade_id	int8	For an order, the value is 0. For a trade — trade ID assigned by liquidity pool
24	amount	int4	For an order, the value is 0. For a trade — trade volume in lots
28	price	dec8	Current price of market
36	trade_time	time8n	Time of making a trade or placing an order
44	trade_type	int1	Trade type. Value: `1` (REGULAR): regular trade
45	dir	int1	Direction of the order that caused the current price of market to be changed. Values: • `1` (Buy): buy; • `2` (Sell): sell
46	pad0	dec8	Additional price
54	flags	int8	Trades flags. Value: `0x1` (HIGH_LIQUIDITY): high liquidity flag
62	yield	dec8	Trade yield

3.5. BestPrices topic

*Snapshot is aggregation of all current data. Updates **replenish** and/or **replace** earlier data.*

The BestPrices snapshot conveys the best offer, the best bid, and the latest trade. One message relates a trading instrument; the liquidity pool and the instrument are specified in the `instrument` component.

The updates are transmitted via the `PricesOnline` messages, the snapshots are transmitted via the `PricesSnapshot` messages.

The final part of a BestPrice message is the `sub_best` repeating component with the number of entries specified in the `sub_prices_count` field (for more information on processing of repeating component, refer to Section 2.5). The component entry includes price level, orders direction, add/update indicator, total disclosed amount of orders at the price level (in lots), and latest update timestamp.

Table 18. Format of message `PricesOnline`: msgid=7651, dynamic size, keys=instrument

Offset	Field	Datatype	Description
	[frame]	[frame]	Session header

Offset	Field	Datatype	Description
0	[md_header]	[md_header]	Header
10	instrument	[instrument]	Component specifying trading instrument
16	sub_prices_offset	int2	Offset of the first `sub_prices` entry from the beginning of this field
18	sub_prices_count	int2	Number of the `sub_prices` group entries
	> sub_prices	[sub_best]	List of the best price levels

Table 19. Format of message `PricesSnapshot`: msgid=7653, dynamic size, keys=instrument

Offset	Field	Datatype	Description
	[frame]	[frame]	Session header
0	[md_header]	[md_header]	Header
10	instrument	[instrument]	Component specifying trading instrument
16	sub_prices_offset	int2	Offset of the first `sub_prices` entry from the beginning of this field
18	sub_prices_count	int2	Number of the `sub_prices` group entries
	> sub_prices	[sub_best]	List of the best price levels

Table 20. Format of component `sub_best`: size 22 bytes

Offset	Field	Datatype	Description
0	price	dec8	Price
8	type	int1	Entry type. Values: • `1` (BEST_BUY): best price to buy; • `2` (BEST_SELL): best price to sell; • `3` (LAST_DEAL)
9	flag	int1	Flag of new entry. Values: • `0x0` (UPDATE): updating; • `0x1` (NEW): adding
10	amount	int4	Total volume of orders at level or volume of trade, in lots
14	time	time8n	Time of recent price level change or trade time

After a trading system restart, the gateway sends the `EmptyBook` message to the BestPrices topic to delete all data on prices from the beginning of the trading day.

Table 21. Format of message `EmptyBook`: msgid=15300, size=16

Offset	Field	Datatype	Description
	[frame]	[frame]	Session header

Offset	Field	Datatype	Description
0	[md_header]	[md_header]	Header
10	instrument	[instrument]	Component specifying trading instrument

3.6. Commons topic

*Snapshot is aggregation of all current data. Updates **replenish** and/or **replace** earlier data.*

The Commons topic transmits various market parameters, see the list below. A Commons message concerns a single trading instrument; the liquidity pool and the instrument are specified in the `instrument` component.

The updates are transmitted via the `CommonsUpdateOnline` messages, the snapshots are transmitted via the `CommonsUpdateSnapshot` messages.

The messages contain the `CommonsUpdateEntry` repeating component and each entry describes a parameter: the `type` field indicates a parameter name, the `value` field — a parameter value. The data type of the `value` field depends on the value of the `type` field. The number of entries is specified in the `entry_count` field (for more information on processing of repeating component please refer to section 2.5).

Snapshots are transmitted in succession. An update is generated on data change.

Table 22. Snapshot and update parameters correspondence

Parameter type	Parameter name	Value of the `value` field	Data type of the `value` field
Latest trade price	price_last	3	dec8
Opening price	price_open	4	dec8
Closing price	price_close	5	dec8
Highest price	price_high	7	dec8
Lowest price	price_low	8	dec8
Close yield of previous day	yield_close	71	dec8
Last trade yield	yield_last	72	dec8
Closing auction price of previous day	price_auction_close_prev	73	dec8
Halt price	price_halt	74	dec8
Timestamp of lowest current price	price_official_min_time	75	time8n
Indicative quote	price_indicative	76	dec8
Turnover of trades at closing auction price, in instrument units	vol_auction_close_extra	79	int8
Turnover for calculation of previous day's market price 3*	price3_turnover_prev	80	dec2
Turnover for calculation of current day's market price 3*	price3_turnover	81	dec2
Turnover for calculation of previous day's market price 2*	price2_turnover_prev	82	dec2

Parameter type	Parameter name	Value of the `value` field	Data type of the `value` field
Turnover for calculation of current day's market price 2*	price2_turnover	83	dec2
Timestamp of current price calculation	price_official_time	84	time8n
Change in current price against official closing price of previous day	price_official_delta	85	dec8
Lowest current price	price_official_min	86	dec8
Latest trade price involved in current price	last_trade_official	87	dec8
Volume disbalance at closing auction	close_imbalance	88	int8
Market price 3 of previous day*	price3_prev	89	dec8
Market price 3 of current day*	price3	90	dec8
Market price 2 of previous day*	price2_prev	91	dec8
Market price 2 of current day*	price2	92	dec8
Last trade price of main session in previous day	price_last_day_prev	93	dec8
Last trade price of main session in current day	price_last_day	94	dec8
Turnover of latest trade in price currency	turnover_last	95	dec2
Official closing price of previous day	price_close_prev	96	dec8
Official current price	price_official	97	dec8
Volume weighted average price of main session in previous day	price_vwap_day_prev	98	dec8
Volume weighted average price of main session in current day	price_vwap_day	99	dec8
Current price	price_current	100	dec8
Settlement price of latest main clearing	price_clearing	101	dec8
Settlement price of latest intermediate clearing	price_inter_clearing	102	dec8
Number of buy orders	orders_buy	103	int8
Number of sell orders	orders_sell	104	int8
Volume of buy orders	buy_vol	105	int8
Volume of sell orders	sell_vol	106	int8
Number of anonymous trades	trades_count	107	int8
Turnover of anonymous trades, in instrument lots	turnover	108	int8
Turnover of anonymous trades, in instrument units	turnover_asset	109	int8
Currency turnover of anonymous trades	turnover_currency	110	dec2

Parameter type	Parameter name	Value of the `value` field	Data type of the `value` field
Total number of trades	total_trades_count	111	int8
Total turnover, in instrument lots	total_turnover	112	int8
Total asset turnover	total_turnover_asset	113	int8
Total currency turnover	total_turnover_currency	114	dec2
Closing auction price	price_auction_close	115	dec8
Closing auction volume	vol_auction_close	116	int8
Volume weighted average price	price_average	117	dec8
Highest bid price*	buy_extreme	118	dec8
Lowest ask price*	sell_extreme	119	dec8
Volume of latest trade in lots	amount_last	120	int8
Timestamp of latest trade	time_last	121	time8n
Price of last trade over previous trading period	price_prev_period_close	122	dec8

* in the current version of the system these parameters are not calculated and not transmitted

Table 23. Parameters unavailable in over-the-counter trades

Parameter type	Parameter name	Value of the `value` field	Data type of the `value` field
Closing price	price_close	5	dec8
Halt price	price_halt	74	dec8
Timestamp of lowest current price	price_official_min_time	75	time8n
Turnover for calculation of previous day's market price 3*	price3_turnover_prev	80	dec2
Turnover for calculation of current day's market price 3*	price3_turnover	81	dec2
Turnover for calculation of previous day's market price 2*	price2_turnover_prev	82	dec2
Turnover for calculation of current day's market price 2*	price2_turnover	83	dec2
Timestamp of current price calculation	price_official_time	84	time8n
Change in current price against official closing price of previous day	price_official_delta	85	dec8
Lowest current price	price_official_min	86	dec8
Latest trade price involved in current price	last_trade_official	87	dec8
Market price 3 of previous day*	price3_prev	89	dec8
Market price 3 of current day*	price3	90	dec8

Parameter type	Parameter name	Value of the `value` field	Data type of the `value` field
Market price 2 of previous day*	price2_prev	91	dec8
Market price 2 of current day*	price2	92	dec8
Last trade price of main session in current day	price_last_day	94	dec8
Official closing price of previous day	price_close_prev	96	dec8
Official current price	price_official	97	dec8
Volume weighted average price of main session in previous day	price_vwap_day_prev	98	dec8
Volume weighted average price of main session in current day	price_vwap_day	99	dec8

* in the current version of the system these parameters are not calculated and not transmitted

Table 24. Format of message `CommonsUpdateOnline`: msgid=1113, dynamic size, keys=instrument

Offset	Field	Datatype	Description
	[frame]	[frame]	Session header
0	[md_header]	[md_header]	Header
10	instrument	[instrument]	Component specifying trading instrument
16	entry_offset	int2	Offset of the first `entry` entry from the beginning of this field
18	entry_count	int2	Number of the `entry` group entries
	> entry	[CommonsUpdateEntry]	List of commons

Table 25. Format of message `CommonsUpdateSnapshot`: msgid=1115, dynamic size, keys=instrument

Offset	Field	Datatype	Description
	[frame]	[frame]	Session header
0	[md_header]	[md_header]	Header
10	instrument	[instrument]	Component specifying trading instrument
16	entry_offset	int2	Offset of the first `entry` entry from the beginning of this field
18	entry_count	int2	Number of the `entry` group entries
	> entry	[CommonsUpdateEntry]	List of commons

Table 26. Format of component `CommonsUpdateEntry`: length 10 bytes

Field	Datatype	Description
type	int1	Parameter name

Field	Datatype	Description
flags	int1	Flag of value. Values: • `0x0` (NORMAL): valid; • `0x1` (DELETE): deleted
value	int8	Parameter value (valid when flags=0)

3.7. TheorPrices topic

*Snapshot is aggregation of all current data. Updates **replenish** and/or **replace** earlier data.*

The TheorPrices topic transmits information about theoretical option prices via the `TheoreticalPrices` messages. Resending lost messages through the recovery gateway is not available.

Table 27. Format of message `TheoreticalPrices`: msgid=1155, dynamic size

Offset	Field	Datatype	Description
	[frame]	[frame]	Session header
0	[md_header]	[md_header]	Header
10	instrument	[instrument]	Component specifying trading instrument
16	price_offset	int2	Offset of the first `price` entry from the beginning of this field
18	price_count	int2	Number of the `price` group entries
	> price	[price]	Prices

3.8. Instruments topic

*Snapshot is aggregation of all current data. Updates **replace** earlier data.*

The Instrument topic broadcasts reference data on trading instruments:

- balance instrument Currency,
- balance instrument Issue,
- balance instrument Spot,
- option series (OptionSeries),
- balance instrument Option,
- balance instrument Bond,
- accrued coupon income (BondAccruedInterest),
- trading modes (TradeModes),
- liquidity pools (Market),
- trading instrument (Instrument).

The Instrument snapshot and updates transmits the same messages. The update of the Instruments topic broadcasts the TradingInstrumentStatus message on instrument status change and the TradingInstrumentLimits on price limit change. The BorrowingStatus message is sent, when short selling availability of an instrument has changed.

The Instruments topic cannot be recovered over TCP.

Table 28. Format of message `Currency`: msgid=931, size=266, keys=balance_id

Offset	Field	Datatype	Description
	[frame]	[frame]	Session header
0	[md_header]	[md_header]	Header
10	balance_id	int4	Balance instrument ID
14	code	char32+1	Currency code
47	desc	char64+1	Full name of currency in English
112	desc_ru	char128+1	Full name of currency in Russian
241	section	char8+1	Market section
250	min_volume	dec8	Minimum volume of asset
258	cfi_code	char6+1	CFI code
265	is_test	int1	Flag of test instrument. Values: • `0` (REAL): Real; • `1` (TEST): Test

Table 29. Format of message `Issue`: msgid=932, size=474, keys=balance_id

Offset	Field	Datatype	Description
	[frame]	[frame]	Session header
0	[md_header]	[md_header]	Header
10	balance_id	int4	Balance instrument ID
14	code	char32+1	Instrument ticker
47	desc	char64+1	Full name of stock in English
112	desc_ru	char128+1	Full name of stock in Russian
241	section	char8+1	Market section
250	min_volume	dec8	Minimum volume of lot
258	isin	char32+1	ISIN
291	cfi_code	char6+1	CFI code
298	reg_num	char32+1	Registration number
331	issuer_name	char64+1	Name of issuer or management company (for stakes)
396	issuer_country	char8+1	Issuer country
405	face_value	dec8	Face value
413	face_value_currency	char8+1	Face value currency
422	total_amount	decn	Total amount of issue

Offset	Field	Datatype	Description
431	security_type	int1	Security type. Values: • 1 (OrdinaryShare): ordinary share or REIT; • 2 (PreferredShare): preferred share; • 3 (OpenEndedMutualFund): open-end mutual fund; • 4 (ClosedEndMutualFund): closed-end mutual fund; • 5 (ETF): security of foreign exchange traded fund; • 6 (RDR): Russian depositary receipt; • 7 (ADR): American depositary receipt; • 8 (GDR): global depositary receipt; • 9 (IntervalMutualFund): share of mutual fund
432	issue_date	time8m	Issue or registration date
440	quotation_list	char32+1	Quotation list
473	is_test	int1	Flag of test instrument. Values: • 0 (REAL): Real; • 1 (TEST): Test

Table 30. Format of message `Spot`: msgid=933, size=281, keys=balance_id

Offset	Field	Datatype	Description
	[frame]	[frame]	Session header
0	[md_header]	[md_header]	Header
10	balance_id	int4	Balance instrument ID
14	code	char32+1	Spot code
47	desc	char64+1	Full name in English
112	desc_ru	char128+1	Full name in Russian
241	section	char8+1	Market section
250	lot	int8	Lot volume in balance instrument units (instrument ID specified in `underlying_id`)
258	date_exec	time8m	Execution date
266	shift	int2	Shift of execution date from today
268	underlying_id	int4	Underlying instrument ID
272	accrued_interest	dec8	Accrued interest as of the delivery date
280	is_test	int1	Flag of test instrument. Values: • 0 (REAL): Real; • 1 (TEST): Test

Table 31. Format of message `OptionSeries`: msgid=980, size=289, keys=series_id

Offset	Field	Datatype	Description
	[frame]	[frame]	Session header
0	[md_header]	[md_header]	Header
10	series_id	int4	Series ID
14	code	char32+1	Option code
47	desc	char64+1	Full name in English
112	desc_ru	char128+1	Full name in Russian
241	section	char8+1	Market section
250	lot	dec8	Lot volume in balance instrument units (instrument ID specified in `underlying_id`)
258	date_exec	time8m	Execution date
266	date_expire	time8m	Expiration date
274	underlying_id	int4	Underlying instrument ID
278	exec_type	int1	Option type. Values: • `0` (ThroughSpot): Option through spot; • `1` (CashSettlement): cash-settled option
279	exec_style	int1	Values: • `1` (American); • `2` (European)
280	series_type	int1	Values: • `1` (Daily); • `2` (Weekly); • `3` (Monthly); • `4` (Quarterly)
281	strike_step	dec8	Strike step

Table 32. Format of message `Option`: msgid=981, size=264, keys=balance_id

Offset	Field	Datatype	Description
	[frame]	[frame]	Session header
0	[md_header]	[md_header]	Header
10	balance_id	int4	Balance instrument ID
14	series_id	int4	Series ID
18	code	char32+1	Option code
51	desc	char64+1	Full name in English

Offset	Field	Datatype	Description
116	desc_ru	char128+1	Full name in Russian
245	section	char8+1	Market section
254	strike	dec2	Strike volume
262	option_type	int1	Option type. Values: • `1` (Call); • `2` (Put)
263	is_test	int1	Flag of test instrument. Values: • `1` (Real); • `2` (Test)

Table 33. Format of message `Bond`: msgid=935, dynamic size, keys=balance_id

Offset	Field	Datatype	Description
	[frame]	[frame]	Session header
0	[md_header]	[md_header]	Header
10	balance_id	int4	Balance instrument ID
14	code	char32+1	Bond code
47	desc	char64+1	Full name in English
112	desc_ru	char128+1	Full name in Russian
241	section	char8+1	Market section
250	min_volume	dec8	Minimum volume of lot
258	isin	char32+1	ISIN
291	cfi_code	char6+1	CFI code
298	date_maturity	time8m	Maturity date
306	coupon_payment_offset	int2	Offset of the first `coupon_payment` entry from the beginning of this field
308	coupon_payment_count	int2	Number of the `coupon_payment` group entries
310	reg_num	char32+1	Registration number of bond issue
343	issuer_name	char64+1	Name of issuer or management company (for stakes)
408	issuer_country	char8+1	Issuer country
417	face_value	dec8	Face value
425	face_value_currency	char8+1	Face value currency
434	issue_amount	decn	Total amount of issue

Offset	Field	Datatype	Description
443	security_type	int1	Security type. Values: • 1 (GovernmentBond): government bond; • 2 (MunicipalBond): municipal bond; • 3 (CentralBankBond): Central bank bond; • 4 (CorporateBond): corporate bond; • 5 (FinancialInstitutionBond): financial institution bond; • 6 (ExchangeTradedBond): exchange traded bond
444	issue_date	time8m	Date of issue
452	quotation_list	char32+1	Quotation list
485	is_test	int1	Flag of test instrument. Values: • 0 (REAL): Real; • 1 (TEST): Test
	> coupon_payment	[coupon_payment]	Schedule of coupon payments

Table 34. Format of message `BondAccruedInterest`: msgid=937, dynamic size, keys=balance_id

Offset	Field	Datatype	Description
	[frame]	[frame]	Session header
0	[header]	[header]	Header
22	balance_id	int4	Balance instrument (bond) ID
26	accrued_interest_offset	int2	Offset of the first `accrued_interest` entry from the beginning of this field
28	accrued_interest_count	int2	Number of the `accrued_interest` group entries
	> accrued_interest	[coupon_payment]	Coupon payment schedule

The list of trading modes, transmitted via the `TradeModes` messages, is the subject to modification. It is not recommended to use parameters of a specific trading mode for setting up the trading system.

Table 35. Format of message `TradeModes`: msgid=942, size=210, keys=trade_mode_id

Offset	Field	Datatype	Description
	[frame]	[frame]	Session header
0	[md_header]	[md_header]	Header
10	trade_mode_id	int2	Trade mode ID
12	name	char64+1	Name of trade mode in English
77	name_ru	char128+1	Name of trade mode in Russian

Offset	Field	Datatype	Description
206	is_address	int1	Negotiated trading flag in trade mode. Values: • 0 (No): non-negotiated; • 1 (Yes): negotiated
207	is_multileg	int1	Multi-leg trade indicator. Values: • 0 (No): sigle-leg; • 1 (Yes): multi-leg
208	is_ext_close	int1	Closing auction indicator. Values: • 0 (No): not traded at closing auction; • 1 (Yes): traded at closing auction
209	over_the_counter	int1	Over-the-counter trade mode indicator. Values: • 0 (No): not present; • 1 (Yes): present

Table 36. Format of message `Market`: msgid=936, size=208, keys=market_id

Offset	Field	Datatype	Description
	[frame]	[frame]	Session header
0	[md_header]	[md_header]	Header
10	market_id	int4	Liquidity pool ID (for a description of values, refer to Section 2.7)
14	desc	char64+1	Full name of market in English
79	desc_ru	char128+1	Full name of market in Russian

Table 37. Format of message `Instrument`: msgid=973, dynamic size, keys=instrument_id

Offset	Field	Datatype	Description
	[frame]	[frame]	Session header
0	[md_header]	[md_header]	Header
10	instrument_id	int4	Trading instrument ID
14	symbol	char32+1	Symbolic instrument ID
47	desc	char64+1	Full instrument name in English
112	desc_ru	char128+1	Full instrument name in Russian
241	status	[instrument_status]	Current status of trading instrument

Offset	Field	Datatype	Description
245	type	char3+1	Trading instrument type: • f: futures; • t: T+N; • o: option; • r: repo; • pr: related trades; • sw: swap; • c: calendar spread; • sf: spot-futures spread; • dvp: delivery versus payment
249	auction_dir	int1	Type of auction. Values: • `0` (Direct): direct auction; • `1` (Inverse): inverse auction
250	price_increment	dec8	Price increment
258	step_price	dec8	Step price
266	legs_count	int2	Number of legs
268	trade_mode_id	int2	Trading mode ID
270	scalping_type	int2	Scalping type. Values: • `0` (NoScalping): no scalping; • `1` (Custom): custom scalping; • `2` (InverseScalping): inverse scalping
272	fee_schema	int1	Fee scheme. Values: • `1` (MakerTakerSpot): maker-taker for spot; • `2` (MakerTakerFutures): maker-taker for futures; • `3` (REPO): repo; • `4` (MemberTariff): maker-taker for spot at members
273	fee_rate_offset	int2	Offset of the first `fee_rate` entry from the beginning of this field
275	fee_rate_count	int2	Number of the `fee_rate` group entries
277	curr_price	char16+1	Currency of the instrument price
294	periods_offset	int2	Offset of the first `periods` entry from the beginning of this field
296	periods_count	int2	Number of the `periods` group entries
298	exchange_instrument_offset	int2	Offset of the first `exchange_instrument` entry from the beginning of this field
300	exchange_instrument_count	int2	Number of the `exchange_instrument` group entries

Offset	Field	Datatype	Description
302	limit_up	dec8	Price limit up
310	limit_down	dec8	Price limit down
318	is_test	int1	Flag of test instrument. Values: • `0` (REAL): Real; • `1` (TEST): Test
319	te_id	int2	Trading engine ID
321	be_mode	int1	Best execution mode. Values: • `0` (External): external trades; • `1` (Internal): internal trades at external prices
322	borrowing_status	int1	Short selling availability for the instrument. Values: • `1` (HARD_TO_BORROW): short selling unavailable; • `2` (EASY_TO_BORROW): short selling available
323	category	int4	Instrument category bitmask. Values: • `0x1` (UNQUALIFIED_CLIENT_PROHIBITION); • `0x2` (FOREIGNSECURITY_CLIENT_PROHIBITION); • `0x4` (FOREIGNETF_CLIENT_PROHIBITION); • `0x8` (UNQUOTRUSECURITY_CLIENT_PROHIBITION); • `0x10` (DERIVATIVES_CLIENT_PROHIBITION); • `0x20` (UNRATEDRUBOND_CLIENT_PROHIBITION); • `0x40` (FOREIGNBOND_CLIENT_PROHIBITION); • `0x80` (STRUCTEDBOND_CLIENT_PROHIBITION); • `0x100` (STRUCTEDINCOMEBOND_CLIENT_PROHIBITION); • `0x200` (REPO_CLIENT_PROHIBITION); • `0x400` (CLOSEDFUND_CLIENT_PROHIBITION); • `0x800` (DELISTED_CLIENT_PROHIBITION)
	> fee_rate	dec8	Fee rate
	> periods	[Period]	Component of trading periods (such as trading session) for instrument
	> exchange_instrument	[ExchangeInstrument]	Component specifying trading instruments at liquidity pools

In this version of the trading system, the `fee_rate` group has five entries. The group has the following sequence of entries:

1. Minimum fee rate, in instrument currency.
2. Fee rate for pre-delivery trades, in instrument currency.
3. Taker fee rate depending on fee scheme: portion of trade volume in price currency for shares; amount of price currency per contract for derivatives; portion of the first leg value multiplied by repo duration for repo.

4. Maker fee rate depending on fee scheme: portion of trade volume in price currency for shares; amount of price currency per contract for derivatives; portion of the first leg value multiplied by repo duration for repo.
5. Accuracy.

Values of third and fourth records are based on the mechanism of fee calculation specified in the `fee_schema` field.

The `category` field indicates the instrument category in the trading system in accordance with designations adopted on the SPB Exchange. The correspondence between designations of instrument categories in the trading system and Interfax is shown in the table below.

Table 38. Correspondence between designations of instrument categories in the trading system and Interfax

Bitmask	Instrument category in the trading system	Instrument category in Interfax
0x1	UNQUALIFIED_CLIENT_PROHIBITION	0
0x2	FOREIGNSECURITY_CLIENT_PROHIBITION	10
0x4	FOREIGNETF_CLIENT_PROHIBITION	11
0x8	UNQUOTRUSECURITY_CLIENT_PROHIBITION	9
0x10	DERIVATIVES_CLIENT_PROHIBITION	—
0x20	UNRATEDRUBOND_CLIENT_PROHIBITION	6
0x40	FOREIGNBOND_CLIENT_PROHIBITION	7
0x80	STRUCTEDBOND_CLIENT_PROHIBITION	4
0x100	STRUCTEDINCOMEBOND_CLIENT_PROHIBITION	8
0x200	REPO_CLIENT_PROHIBITION	—
0x400	CLOSEDFUND_CLIENT_PROHIBITION	5
0x800	DELISTED_CLIENT_PROHIBITION	—

Table 39. Format of message `TradingInstrumentStatus`: msgid=2031, size=84, keys=instrument

Offset	Field	Datatype	Description
	[frame]	[frame]	Session header
0	[md_header]	[md_header]	Header
10	instrument	[instrument]	Component specifying trading instrument
16	trading_status	int1	Current status of trading instrument. Values: • `2` (HALT): trading is halted; • `17` (TRADING): trading in progress; • `18` (NO_TRADING): no trading; • `102` (CLOSE): trading during closing auction; • `103` (CLOSE_PERIOD): trading during close period; • `107` (DISCRETE_AUCTION): trading during discrete auction; • `118` (OPEN): trading during opening auction; • `120` (FIXED_PRICE_AUCTION): trading at closing auction price

Offset	Field	Datatype	Description
17	reserved	char2+1	Reserved field. To be filled with null byte
20	comment	char63+1	Comments

Table 40. Format of message `TradingInstrumentLimits`: msgid=2032, size=30, keys=instrument_id

Offset	Field	Datatype	Description
	[frame]	[frame]	Session header
0	[md_header]	[md_header]	Header
10	instrument_id	int4	Trading instrument ID
14	limit_up	dec8	Price limit up
22	limit_down	dec8	Price limit down

Table 41. Format of message `BorrowingStatus`: msgid=2033, size=15, keys=instrument_id

Offset	Field	Datatype	Description
	[frame]	[frame]	Session header
0	[md_header]	[md_header]	Header
10	instrument_id	int4	Trading instrument ID
14	borrowing_status	int1	Short selling availability for the instrument. Values: • `1` (HARD_TO_BORROW): Short selling unavailable; • `2` (EASY_TO_BORROW): Short selling available

Table 42. Format of component `coupon_payment`: size 16 bytes

Offset	Field	Datatype	Description
0	date	time8m	Date of payment
8	value	dec8	Amount of payment

Table 43. Format of component `Period`: size 30 bytes

Offset	Field	Datatype	Description
0	start	time8m	Start timestamp
8	finish	time8m	End timestamp

Offset	Field	Datatype	Description
16	mode	int2	Type of auction. Values: • `0` (ProRata): pro rata two-way anonymous auction; • `1` (Parity): parity two-way anonymous auction; • `2` (TimePriority): time priority anonymous auction; • `3` (Address): negotiated trading; • `4` (OpenAuction): opening auction; • `5` (CloseAuction): closing auction; • `6` (NoTrade): no trading; • `7` (ExtClose): closing auction at liquidity pool
18	currency_id	int4	Currency ID of traded instrument
22	underlying_offset	int2	Offset of the first `underlying` entry from the beginning of this field
24	underlying_count	int2	Number of the `underlying` group entries
26	markets_offset	int2	Offset of the first `markets` entry from the beginning of this field
28	markets_count	int2	Number of the `markets` group entries
	> underlying	[Underlying]	Component for specifying the lot volume of a trading instrument within a period
	> markets	int2	List of available liquidity pools (for a description of values, refer to section 2.7)

Table 44. Format of component `ExchangeInstrument`: size 61 bytes

Offset	Field	Datatype	Description
0	instrument	[instrument]	Component specifying trading instrument
6	code_group	char16+1	Market section
23	code	char16+1	Instrument ticker
40	code_extra	char16+1	Instrument code
57	status	[instrument_status]	Current status of trading instrument

Table 45. Format of component `instrument_status`: size 4 bytes

Offset	Field	Datatype	Description
0	trading_status	int1	Current status of trading instrument. Values: • `2` (HALT): trading is halted; • `17` (TRADING): trading in progress; • `18` (NO_TRADING): no trading; • `102` (CLOSE): trading during closing auction; • `103` (CLOSE_PERIOD): trading during close period; • `107` (DISCRETE_AUCTION): trading during discrete auction; • `118` (OPEN): trading during opening auction; • `120` (FIXED_PRICE_AUCTION): trading at closing auction price
1	suspend_status	int1	Reserved field. To be filled with null byte
2	routing_status	int1	Reserved field. To be filled with null byte
3	reason	int1	Reserved field. To be filled with null byte

Table 46. Format of component `Underlying`: size 15 bytes

Offset	Field	Datatype	Description
0	balance_id	int4	Balance instrument ID
4	qty	decn	Number of balance instrument units
13	flags	int2	Flags field. Values: • `0x1` (CORP_DUE_BILL): additional liability in connection with corporate event; • `0x2` (CORP_CORRECTION): liability adjustment by clearing center in connection with corporate event; • `0x4` (CORP_INCOME_RETURN): transfer of income in connection with corporate event; • `0x8` (PRINCIPAL_OBLIGATION): principal liability flag

3.9. Heartbeat message

The gateway sends a `MdHeartbeat` in the topic updates if no message is transmitted for more than a second.

Table 47. Format of message `MdHeartbeat`: msgid=15236, size=14

Offset	Field	Datatype	Description
	[frame]	[frame]	Session header
0	[md_header]	[md_header]	Header
10	reserved	int4	Reserved field. Zero value

4. Recovery gateway

The recovery gateway allows the client to request resending of update messages, if they were lost via UDP. You can request the following topics' updates via the recovery gateway: OrderBook, Trades, BestPrices, Commons and CurrentPriceOfMarket.

The whole history from the start of a trade day is available for recovery only for Trades and CurrentPriceOfMarket topics; for all other topics a client can recover only recent messages. Due to technological limitations, messages from the previous trade day can be recovered.

Client should use the discovery service to connect to the recovery gateway.

4.1. Session layer

4.1.1. Discovery service

The Discovery service provides a host address for client connections to the trading system gateway. The client should request the service for address allocation each time before connecting to the gateway. Upon receipt of response, the client should disconnect from the login server and connect to a gateway through the received address.

For the address for accessing the Discovery service, please refer to document *Network Connectivity*.

After establishing connection with the Discovery service, the client should send the `Hello` message. The message contains the session header `frame` (for more details, refer to Section 2.2). The client should specify login and password, and the IP address of the client must be authorized for the specified login (user ID).

Table 48. Format of message `Hello`: msgid=1, size=32

Offset	Field	Datatype	Description
	[frame]	[frame]	Session header
0	login	ascii16	Login
16	password	ascii16	Password

In response to request, the server sends the `Report` message. If this message has `status`=0, the message contains repetitive component `Report_Address`; the number of component records will be specified in the field `addresses_count` (for more details on processing of repeating groups, refer to Section 2.5). The component includes fields `type` (gateway attribute) and `address` (host address and gateway port). Gateway attributes may combine.

After the trading system responds, the gateway will expect the client's login connection to the specified address. In case of failure, the client is recommended to make two additional connection attempts with an interval of half a second. If the login is invalid or blocked, the server response will contain `status=1`.

Table 49. Format of message `Report`: msgid=2, dynamic size

Offset	Field	Datatype	Description
	[frame]	[frame]	Session header
0	status	int2	Request status. Values: • `0` (Success): success; • `1` (Fail): reject due to invalid login/password
2	reason	char127+1	Textual description
130	addresses_offset	int2	Offset of the first `addresses` entry from the beginning of this field
132	addresses_count	int2	Number of the `addresses` group entries

Offset	Field	Datatype	Description
	> addresses	[Report_Address]	Address list

Table 50. Format of component `Report_Address`: size 52 bytes

Offset	Field	Datatype	Description
0	type	int2	Gateway attributes, bit mask. Values: • `0x1` (Transaction): trading; • `0x2` (DropCopy): drop-copy; • `0x4` (Risk): risk management; • `0x8` (Dictionary): dictionaries; • `0x10` (MarketData): market data recovery; • `0x4000` (Backup): backup
2	ver	int1	Interface version
3	pad0	int1	Reserved field, filled with zero bytes
4	address	char47+1	Address of host and gateway port

4.1.2. Session initialization

A session is established over a network connection between the client's system and the gateway of the trading system.

Once connection is established, the client can send the `Login` message to initiate a session. The message includes the user ID and the password. The system validates the authentication parameters and answers with the `Logon` message and so the session is active. Upon receipt of a malformed `Login` message or invalid login/password, the server breaks the connection.

A login may have a single concurrent session. If the server detects a second connection attempt via the same login while a valid session is already underway, the server will respond with `Reject`.

Table 51. Format of message `Login`: msgid=8001, size=37

Offset	Field	Datatype	Description
	[frame]	[frame]	Session header
0	login	ascii16	Login
16	password	ascii16	Password
32	reset_seq	int1	Reset sequence numbers indicator. Values: • `0` (no): sequence numbers continue; • `1` (yes): sequence numbers reset
33	heartbeat_ms	int4	`Heartbeat` frequency in milliseconds

Table 52. Format of message `Logon`: msgid=8101, size=24

Offset	Field	Datatype	Description
	[frame]	[frame]	Session header
0	last_seq	int8	Last application message available to client. If altered form the last received message, `ResendRequest` is to be sent

Offset	Field	Datatype	Description
8	expected_seq	int8	Next application message expected from client
16	system_id	ascii8	Deployment ID

4.1.3. Keeping session in active state

The client and the gateway must exchange `Heartbeat` messages to maintain session in active state. Heartbeat must be sent, if no session or application message has been sent within the heartbeat interval.

When initiating a session, the client sets the heartbeat interval in the field `heartbeat_ms` of the `Login` message.

If the server detects that the client has not sent any messages, including the `Heartbeat` messages, for a period longer than the specified interval, the system will break the connection. The client is expected to do the same, if inactivity is detected on the part of the server.

Table 53. Format of message `Heartbeat`: msgid=8103, size=0

Offset	Field	Datatype	Description
	[frame]	[frame]	Session header

4.1.4. Send rate limit for session messages

The rate at which a client can send session level messages into the system is limited. When client's send rate exceeds the limit, the system terminates the user session.

4.1.5. Message numbers

All application messages have a unique number throughout the trading day. Messages by each session side (the client and the gateway) are sequentially numbered with positive integers starting with 1. This allows to request and resend messages lost in case of unexpected disconnection.

Sequence numbers are not assigned to session messages — the `seq` value is always 0.

In order to maintain sequential numbering of messages, at session initialization the gateway provides two key values in its `Logon` message — the number of the last message sent (`last_seq`) and the expected number of the following message (`expected_seq`).

If the message number differs from the expected one, the gateway terminates the connection. After disconnection, the client should reconnect by addressing the Discovery service and restore the number of messages according to the values obtained in the `Logon` message from the gateway. The gateway never initiates a change in numbering when receiving a message with the number higher than expected.

The trading system supports continuous message numbering between trading sessions, including trading days. The client should set `reset_seq=1` in message `Login` at session initialization to reset numbering.

4.1.6. Session termination

The server or the client sends `Logout` to terminate the session and expects the other party to disconnect.

Table 54. Format of message `Logout`: msgid=8002, size=16

Offset	Field	Datatype	Description
	[frame]	[frame]	Session header
0	login	ascii16	Login, client gateway ID

4.1.7. Message rejection

If the client's message is either malformed or contains invalid values, the system rejects such message and responds with `Reject`. The `ref_msgid` field specifies message type, `ref_seq` contains the application level message number or has 0 for session message, fields `reason` and `message` contain, correspondingly, code of rejection reason and its description.

Table 55. Format of message `Reject`: msgid=8102, size=45

Offset	Field	Datatype	Description
	[frame]	[frame]	Session header
0	ref_seq	int8	Sequence number of rejected message
8	ref_msgid	int2	Type of rejected message
10	reason	int2	Code of rejection reason
12	message	char32+1	Rejection parameters or textual description

4.1.8. Disconnection

System disconnects when receiving message:

- with unknown value of `msgid`,
- with a `size` incorrect for the specified message type,
- with a `seq` number other than expected.

4.1.9. Data request

To request data, client should send `TopicRequest` to the trading system gateway specifying `topic` ID. The client does not have to fill the `clorder_id` field.

The client can specify the range of requested messages through `topic_seq` and `topic_seqend` fields:

- `topic_seq=n, topic_seqend=m` — request for messages from *n* to *m*.
- `topic_seq=0, topic_seqend=n` — request for messages from the lowest number available to *n*.

If a request can be processed, the client will receive the TopicReport message and after that should expect data messages. After data transfer is completed, the client will also receive `TopicReport`.

If a request is incorrect or cannot be processed, the client will receive the TopicReject message.

If you want to request a new topic, wait until you have received all messages, related to the previous topic request, to avoid network overload.

Table 56. Format of message `TopicRequest`: msgid=301, size=101

Offset	Field	Datatype	Description
	[frame]	[frame]	Session header
0	[user_header]	[user_header]	Standard header
20	topic	ascii64	Topic ID
84	topic_seq	int8	First number of requested messages
92	topic_seqend	int8	Last number of requested messages
100	mode	int1	Broadcast mode. Value: `0` (DATA_SLICE): snapshot

4.1.10. Report on executing request

The client will receive the `TopicReport` message in the following cases:

- successful execution of the data request TopicRequest;
- completion of snapshot transmission.

The message includes reference fields `topic_lastseq` (the number of the last message generated in the topic) and `topic_lastseqsent` (the number of the last message sent to the client).

Table 57. Format of message `TopicReport`: msgid=401, size=134

Offset	Field	Datatype	Description
	[frame]	[frame]	Session header
0	[gate_header]	[gate_header]	Standard header
46	topic	ascii64	Topic ID
110	topic_id	int4	Numerical topic ID
114	status	int2	Status of data transfer. Value: `0` (DATA_SLICE): snapshot transfer
116	marker	int2	Indicator of start and finish of data transfer. Values: • `0` (START): start of data transfer; • `2` (SLICE_END): snapshot transfer completed
118	topic_lastseq	int8	Number of the last message generated in the topic
126	topic_lastseqsent	int8	Number of the last message sent to the client

4.1.11. Report on rejecting request

If the client's request is incorrect or cannot be processed, the client will receive the TopicReject message. The reason for rejection is specified in the `reason` field.

The `TopicReject` message includes reference fields `topic_lastseq` (the number of the last message generated in the topic) and `topic_lastseqsent` (the number of the last message sent to the client).

For streams with `Trades` and `IQ` identifiers (refer to document *Network Connectivity*) in the `TopicReject` messages, the number in the `topic_firstseq` field is the same as the number of the first available message.

Table 58. Format of message `TopicReject`: msgid=402, size=142

Offset	Field	Datatype	Description
	[frame]	[frame]	Session header
0	[gate_header]	[gate_header]	Standard header
46	topic	ascii64	Topic ID
110	topic_id	int4	Numerical topic ID
114	status	int2	Status of data transfer. Values: • `0` (DATA_SLICE): snapshot transfer; • `2` (INACTIVE): no data transfer
116	reason	int2	Reason for rejection. Values: • `1` (BAD_TOPIC): invalid topic identifier; • `4` (DATA_NOT_AVAILABLE): data not available; • `5` (DUPLICATE_REQUEST): repeated request; • `6` (BAD_SEQ): invalid message number in the topic; • `7` (BAD_MODE): invalid mode

Offset	Field	Datatype	Description
118	topic_firstseq	int8	Number of the first message since the beginning of the trading day
126	topic_lastseq	int8	Number of the last message generated in the topic
134	topic_lastseqsent	int8	Number of the last message sent to the client

Appendix A. Error codes

Table 59. Error codes list

Code	Description
0	Ok
1	Sell short is not allowed.
5	Missed tag.
100	Filled excess tag.
999	Internal error.
1000	Incorrect login.
1001	Incorrect instrument.
1002	Incorrect client ID.
1003	Invalid member_id.
1004	Invalid account.
1005	Incorrect client group.
1006	Incorrect exchange.
1007	Instrument not traded.
1008	Invalid routing options.
1100	Invalid order direction.
1101	Incorrect price.
1102	Incorrect price_extra.
1103	Incorrect amount.
1104	Incorrect amount_extra.
1105	Invalid order type.
1106	Invalid time_in_force.
1107	Invalid passive_only.
1108	Invalid auto_cancel.
1109	Invalid flags.
1110	Invalid mode.
1111	Incorrect clorder_id.
1112	Incorrect orig_clorder_id.
1113	Invalid prime_exchange.
1114	Invalid date_expire.
1115	Invalid comment.

Code	Description
1116	Invalid level.
1117	Invalid trade_mode.
1200	Invalid segment.
1201	Incorrect extra1.
1202	Incorrect OTC code for negotiated trade initiator.
1203	Incorrect OTC code for counter party.
1204	Invalid order_type for this instrument.
1205	Order_type not supported by exchange.
1206	Invalid order_type for Client ID.
1207	Incorrect price for this order_type.
1208	Incorrect amount_extra for this order_type.
1209	Invalid time_in_force for this order_type.
1210	Invalid flags for this order_type.
1211	Invalid instrument for replacement mode.
1212	Invalid member_id for replacement mode.
1213	Invalid client_id for replacement mode.
1214	Invalid account for replacement mode.
1215	Invalid parameters of rejected counter order.
1216	Invalid replacement parameters.
1217	Invalid time_in_force for this instrument.
1218	Invalid replacement mode for this login.
1219	Invalid flags for this instrument.
1300	Both orig_clorder_id and order_id filled.
1301	Duplicate clorder_id.
1302	Price exceeds limits.
1303	Order type not supported for this client ID.
1304	Order type not supported by exchange.
1305	Invalid prime_exchange for this instrument.
1306	Liquidity pool unavailable for client ID.
1307	Invalid order_type for this instrument.
1308	User has no permissions to cancel orders of account specified.
1309	User has no permissions to replace orders of account specified.

Code	Description
1310	User has no permissions to reject this order.
1311	Order currently being replaced.
1312	Order sent before system crash, but received after recovery.
1313	Limitation not available for this instrument.
1314	User has no permissions to use this mode.
1315	This exchange is prohibited for clearing member.
1316	This exchange is prohibited for trade member.
1317	Order submission via the login is blocked.
1318	Order submission via the login is blocked for the client code.
1319	Order submission via the login is blocked for the TCA.
1400	Instrument not available for market maker.
1401	No permissions to trade this instrument.
1402	No permissions to indicate 'No matching another market maker's orders'.
1403	Client has no permissions to trade with using this account.
1404	Liquidity pool not available for this smart order router.
1405	No permissions to trade this instrument category.
1500	Trade engine IDs (te_id) do not match.
1501	Incorrect te_id.
1502	Request received during the limited margin update.
1700	User has no permission for limited margin service.
1701	Client has no permissions for limited margin service.
1702	Client group has no permissions for limited margin service.
1703	Account has no permissions for limited margin service.
1704	Main account has no permissions for limited margin service.
1710	Invalid parameters for limited margin of client.
1711	Invalid parameters for limited margin of client group.
1712	Invalid parameters for limited margin of account.
1713	Invalid parameters for limited margin of main account.
1714	Request for limited margin update for client received when the previous request still processing.
1715	Request for limited margin update for client group received when the previous request still processing.
1716	Request for limited margin update for TCA received when the previous request still processing.
1717	Request for limited margin update for principal TCA received when the previous request still processing.

Code	Description
1720	Incorrect limit for limited margin.
1721	Incorrect instrument limit for limited margin.
1722	Incorrect order limit for limited margin.
1723	Incorrect extra limit for limited margin.
1750	Insufficient limit for limited margin of client.
1751	Insufficient instrument limit for limited margin of client.
1752	Insufficient order limit for limited margin of client.
1753	Insufficient extra limit for limited margin of client.
1754	Insufficient limit for limited margin of client group.
1755	Insufficient instrument limit for limited margin of client group.
1756	Insufficient order limit for limited margin of client group.
1757	Insufficient extra limit for limited margin of client group.
1758	Insufficient limit for limited margin of account.
1759	Insufficient instrument limit for limited margin of account.
1760	Insufficient order limit for limited margin of account.
1761	Insufficient extra limit for limited margin of account.
1762	Insufficient limit for limited margin of main account.
1763	Insufficient instrument limit for limited margin of main account.
1764	Insufficient order limit for limited margin of main account.
1765	Insufficient extra limit for limited margin of main account.
1766	The client has active orders of limited margin.
1767	The client group has active orders of limited margin.
1768	The TCA has active orders of limited margin.
1769	The principal TCA has active orders of limited margin.
1770	Limited margin suspended for client.
1771	Limited margin suspended for client group.
1772	Limited margin suspended for account.
1773	Limited margin suspended for main clearing account.
1780	Invalid liquidity pool for limited margin service.
1800	Incorrect yield type specified.
1801	Incorrect yield conversion direction specified.
1980	Invalid stages in info field.

Code	Description
2100	Account does not belong to member_id.
2200	No permissions to submit trading instructions.
2201	Client group level prohibition is set.
2202	Trade member level prohibion is set.
2203	Clearing member prohibion is set.
2204	Trade administrator level prohibion is set.
2300	No permissions to place an unsecured order.
2400	No permissions to cancel order.
2600	No permissions to set limit for clearing account.
2601	No permissions to set limits for client ID.
2602	No permissions to set limits for client group.
2603	Invalid type.
2604	Invalid value.
2605	Ambiguous type.
2700	Client ID has insufficient funds.
2701	Client ID has insufficient assets.
2702	Client group has insufficient funds.
2703	Client group has insufficient assets.
2704	Account has insufficient funds.
2705	Account has insufficient assets.
2706	Main clearing account has insufficient funds.
2707	Main clearing account has insufficient assets.
2708	Clearing member has insufficient funds.
2709	Insufficient blocked assets.
3000	Market or IOC order expired after no trades.
3001	Order canceled after no trades, to avoid a cross trade.
3002	Order canceled after no trades, to avoid a crossed book.
3003	Client order not found.
3004	Instrument trading suspended.
3005	User has no permission to trade this instrument during the current trading period.
3100	TCA of maker and that of taker have no conversion bank indicator.
3911	Incorrect te_id.

Code	Description
4000	ECN not available or no liquidity pool available.
4001	The specified liquidity pool not available.
4002	Order forcedly routed to a liquidity pool after rejected by risk management at the trading system.
4003	Client ID not registered at all the available liquidity pools.
4004	Client ID not registered at the trading system.
4005	Client ID not registered at liquidity pool.
4006	Order cannot be routed to any liquidity pool.
4100	Order pending cancel.
4101	The order was rejected by an external platform.
4200	Invalid client for TCA registered at liquidity pool.
4201	Invalid TCA for liquidity pool.
5000	Invalid application message type.
5001	Invalid routing_dest.
5002	Invalid message type for this login.
5003	Login has no permissions to submit such instruction.
5200	User already logged in.
5201	Discovery service settings timeout.
5202	Incorrect heatbeat_ms.
5203	Incorrect user ID / password.
5204	Incorrect message sequence number.
5205	Invalid session message type.
5206	User not logged in.
5207	Another resend request processing in progress.
5208	Incorrect range limit.
5209	Invalid reset_seq.
5210	Requested messages range excess.
5211	Invalid session message size.
5212	Disconnected by the operator.
5300	Invalid topic.
5301	Snapshot with updates has already been requested.
5302	Snapshot with updates has not been requested.
5303	Requested data not available.

Code	Description
5304	Another request processing in progress.
5400	Reset_seq indicated, but seqnums cannot be reset.
5401	Number of messages exceeded limit.
5601	Both account and parties filled.
7000	Order canceled before sending to ASTS.
7001	Order canceled as no answer received.

Also you can get errors come in range —11000-11999. These are the error codes returned by the trading system of the Moscow stock exchange (ASTS). To get the ASTS error id , you need to subtract 11000 from the internal error id. The description of these errors, a client can get from the ASTS documentation.

Appendix B. Revision history

Version 1.13.0 24 December 2015

1. The is_test field is added in Currency, Issue, Spot, Futures, and Bond messages.
2. In Instrument message, the is_test, te_id, and be_mode fields are added, the reserved removed and the msgid changed.
3. In the Underlying component, the flags field is added and the size of the qty field is altered.

Version 1.12.0 10 November 2015

The broadcast of the CurrentPriceOfMarket topic started.

Version 1.11.1 14 October 2015

The datatype for the type=76 value is altered in the Commons update.

Version 1.11.0 1 October 2015

The size of the Underlying component is altered, because the `qty` datatype is changed.

Version 1.10.0 2 July 2015

1. New value 76 of the `type` field in the Commons Update added.
2. The format of Instrument message amended—size of `trade_mode_id` reduced to 2 bytes and `reserved` field added.