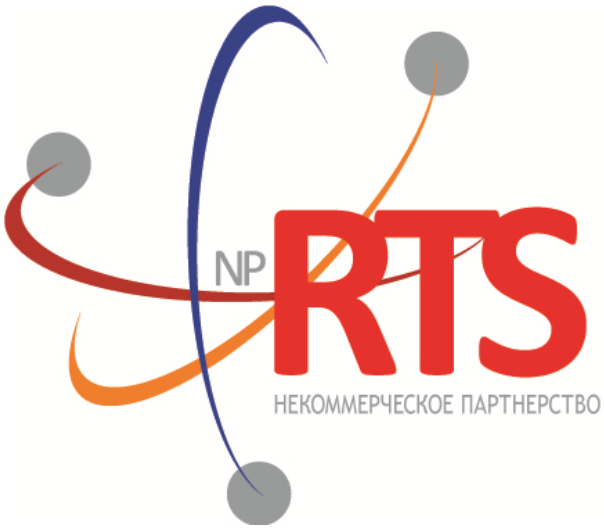

ADVANCED
TRADING & CLEARING
PLATFORM

Native Protocol Market Data Service

System version 1.7

Interface version 36

Document version 1.17.0

02 March 2018

Revision history

Version 1.17.0 November 3, 2017

1. The BorrowingStatus message has been added to the Instruments channel.
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 channel, unavailable for over-the-counter instruments, has been added.
8. Terminology changes.
9. Error codes added.

Version 1.16.0 November 30, 2016

1. New field markets 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 in the Instruments channel.

Version 10.14.0 9 March 2016

1. The message broadcast in the Commons snapshot is changed.
2. The Commons feed 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

The trading platforms provides the client with real-time market data.

1.1. Market data channels

The trading platform broadcasts several market data channels:

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. BestPrices is the top of an instrument order book—the highest bid and the lowest ask.
4. Commons is a list of statistic data of the accessed trading venues.
5. Instruments is instrument reference information.

The broadcast channels are listed in the *Network Connectivity* document.

1.2. Feeds

A channel is transmitted as two feeds—snapshot and updates. A snapshot is an entire set of the effective data, e.g. a whole order book of a specific trading instrument. An update is generated upon data change, e.g. a trade match, and an update message may contain several updates.

A set of messages that composes a snapshot is framed by the `SnapshotStarted` and `SnapshotDelivered` messages. The both messages contain the sequence number of the last update included in the snapshot. A following snapshot starts as soon as the previous one is completely transmitted.

The Instruments channel snapshot broadcasts instrument and trade mode reference data. The Instruments update transmits the `TradingInstrumentStatus` message on instrument status change and the `TradingInstrumentLimits` on price limit change.

A group of messages within a single snapshot starts with `SnapshotStarted` and ends with `SnapshotFinished` messages. These messages have the `update_seq` field, representing the number of message from the updates stream, after which the snapshot has been created.

During a period of inactivity in an update feed the server sends a `Heartbeat` to acknowledge connection. If no message comes for a longer period, there is either a transmission delay or absence of connection.

1.2.1. Feed duplication

Each market data feed is broadcast through two identical UDP feeds—*A* and *B*. The both feeds simultaneously transfer messages with the same numbers. The feed duplication provides more transmission fidelity and lowers the probability of package loss. The client is strongly recommended to process the both feeds. If a package is lost in the both feeds, the client should either expect the next snapshot or request the message via Recovery over TCP (for more details please refer to section 3).

1.3. Algorithm of receiving and processing market data

If you want to connect to a channel transmitting snapshots and updates, it is recommended to connect to both streams. First, you should receive a complete snapshot, them start recording incoming updates. A snapshot starts after receiving the `SnapshotStarted` message, containing the `update_seq` field. All subsequent updates should have `update_seq` values greater, than the `update_seq` value of the `SnapshotFinished` message, completing the snapshot. After that you can disconnect from the snapshots channel and receive updates only.

If an update has been lost, it should be requested in recovery gateway. There is delay between appearing of the same messages in market data gateway and messages in recovery gateway. If a significant number of messages is lost, it is recommended to connect to the snapshots stream instead of attempting to recover lost updates.

1.3.1. Example of receiving market data from OrderBook and BestPrices channels

1. Connect to the updates stream of the required channel and save all incoming messages.

2. Connect to the snapshot stream of the channel and wait for `SnapshotStarted`.
3. Save all incoming messages, until `SnapshotFinished` is received.
4. If some snapshot messages have been skipped, or `update_seq` values are different for `SnapshotStarted` and `SnapshotFinished` messages, then repeat actions **2** and **3**.
5. If there is no saved update message with the number equal to `update_seq+1`, then repeat actions **2** and **3**.
6. Add messages from update channel, which number is greater, than `update_seq`, to the saved snapshot.

Messages from duplicate streams A and B should be separated during data collection. Received messages should be ordered by their numbers. If some messages in the updates stream are missing, they can be recovered in recovery gateway. Messages, missing from the snapshots stream, cannot be recovered.

1.3.2. Example of receiving market data from Trades

1. Connect to the UDP stream of the `Trades` channel and wait for first trade message. If trade message is not available withing 5 seconds after the connection, then the `Heartbeat` message is received.
2. Obtain sequential number `seq` of the first received message (for example, `Heartbeat` message with `seq=305`).
3. Suppose, that the last trade message received during the trade day has `last_seq=105`. Sequential numbers of messages are reset every night and every starting message of the day has `last_seq=0`.
4. This way we determine, that messages from `seq=106` to `seq=304`, were not received.
5. Connect to the gateway and request the required range of messages by sending the `TopicRequest` message.
6. The `TopicRequest` message must contain (a) identifier of recovered market data stream in the `topic` field and (b) range of requested messages (in this case `topic_seq=106` and `topic_seqend=304`). For identifiers of data streams refer to the *Network connectivity* document.
7. 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 `Trade` messages have gaps between `topic_seq` values, because the `Heartbeat` messages were received between `Trade` messages.

1.4. Datatypes

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*. Its value should be within the range from 0 to 8.

`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. Market data messages

For datatypes please refer to section 1.4.

2.1. Formats of common components

Table 1. Format of component `instrument`: length 6 bytes

Field	Datatype	Description
market_id	int2	Liquidity pool ID (please refer to section 3.6)
instrument_id	int4	Trading instrument ID

Table 2. Format of component `md_header`: length 10 bytes

Field	Datatype	Description
system_time	time8n	Timestamp of message generation
source_id	int2	Message source (for values please refer to section 3.5)

2.2. `Heartbeat`

The server sends a `Heartbeat` in the updates feed if no message is transmitted for more than a second.

Table 3. `Heartbeat: msgid=15236, size=14`

Offset	Field	Datatype	Description
	[frame]	[frame]	Session header
0	[md_header]	[md_header]	Standard header
10	reserved	int4	Reserved for future. Zero value

2.3. `EmptyBook`

The server sends the `EmptyBook` message to the OrderBook channel for an order book clearing after a trading platform restart.

Table 4. `EmptyBook: msgid=15300, size=16`

Offset	Field	Datatype	Description
	[frame]	[frame]	Session header
0	[md_header]	[md_header]	Standard header
10	instrument	[instrument]	Component for instrument identification

2.4. Snapshot start and finish acknoledgements

For all channels, a snapshot is preceded by a `SnaphotStarted` and followed by a `SnapshotDelivered`. 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 `ref_seq` grater than the `updates_seq`.

Table 5. `SnapshotStarted: msgid=12345, size=18`

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

Table 6. `SnapshotFinished: msgid=12312, size=18`

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

2.5. Processing messages with repetitive components and fields

Several message types contain one or more repeating groups 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 group entries. The `offset` field indicates an offset in bytes of first entry of the group from the beginning of this very field; its value is no less than 4.

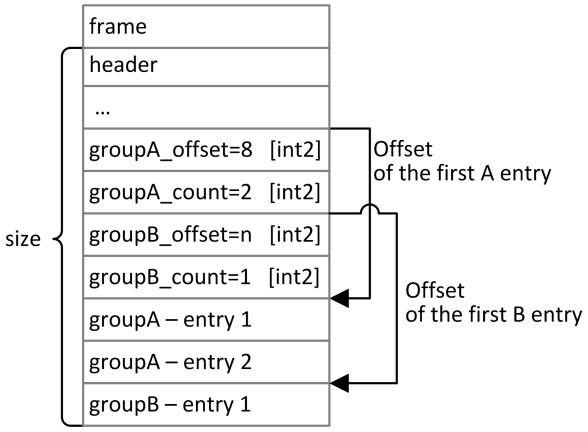

Figure 1. Template of a message with two repeating components

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

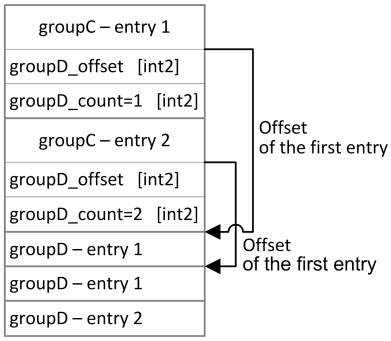

Figure 2. Template of embedded components

2.6. OrderBook channel

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 final part of an OrderBook message is the `PriceLevel` repeating group with the number of entries specified in the `PriceLevel_count` field. (For more information on processing of repeating component please refer to section 2.5.) A group entry includes price level, orders direction, add/update indicator, total disclosed amount of orders at the price level, 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.

Table 7. OrderBook message (`msgid=1112` in snapshot or `msgid=1111` in update, dynamic length)

Offset	Field	Datatype	Description
	[frame]	[frame]	Session header
0	[md_header]	[md_header]	Standard header
10	[instrument]	[instrument]	Component for instrument identification
16	PriceLevel_offset	int2	Offset of the first `PriceLevel` entry from the start of this field. Value: `4`
18	PriceLevel_count	int2	Number of `PriceLevel` entries
	> [PriceLevel]	[PriceLevel]	List of price levels

Table 8. Component `PriceLevel`: size 22 bytes

Field	Type	Description
price	dec8	Price
type	int1	Side of orders: • `1` (BUY) • `2` (SELL)
flag	int1	New/update entry indicator: • `0x0` (UPDATE) • `0x1` (NEW)
amount	int4	Total volume of orders at price level

Field	Type	Description
time	time8n	Timestamp of latest update

2.7. Trades channel

Upon a trade execution, the trading platform generates a message containing trade parameters with the liquidity pool of execution in the `market` field of the `instrument` component.

Table 9. Trades message (`msgid=15210, size=46`)

Offset	Field	Datatype	Description
	[frame]	[frame]	Session header
0	[md_header]	[md_header]	Standard header
10	[instrument]	[instrument]	Component for instrument identification
16	trade_id	int8	Trade identifier assigned by liquidity pool
24	amount	int4	Trade volume
28	price	dec8	Trade price
36	trade_time	time8n	Transaction timestamp
44	trade_type	int1	Trade type: • `1` (REGULAR)
45	dir	int1	Taker's order side: • `1` (BUY) • `2` (SELL)

2.8. CurrentPriceOfMarket channel

A message in CurrentPriceOfMarket channel is created, when the current market price. The message includes: new `price`, time of price change `trade_time` and deal side `dir`.

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

1. If a deal is made, the current price becomes equal to the deal price.
2. If an anonymous buying order appears in the order book, and its price is higher, than the current market price, the current market price becomes equal to the buying order price.
3. If an anonymous selling order appears in the order book, and its price is lower, than the current market price, the current market price becomes equal to the selling order price.

Table 10. Trades message (`msgid=15210, size=46`)

Offset	Field	Datatype	Description
	[frame]	[frame]	Session header
0	[md_header]	[md_header]	Standard header
10	[instrument]	[instrument]	Component for instrument identification
16	trade_id	int8	Trade identifier assigned by liquidity pool

Offset	Field	Datatype	Description
24	amount	int4	Trade volume
28	price	dec8	Trade price
36	trade_time	time8n	Transaction timestamp
44	trade_type	int1	Trade type: • `1` (REGULAR)
45	dir	int1	Taker's order side: • `1` (BUY) • `2` (SELL)

2.9. BestPrices channel

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

Table 11. BestPrices message (`msgid=7653` in snapshot or `msgid=7651` in update, dynamic length)

Offset	Field	Datatype	Description
	[frame]	[frame]	Session header
0	[md_header]	[md_header]	Standard header
10	[instrument]	[instrument]	Component for instrument identification
16	BestPrice_offset	int2	Offset of the first `BestPrice` from the start of this field. Value: `4`
18	BestPrice_count	int2	Number of `BestPrice` entries
	> [BestPrice]	[BestPrice]	List of best price levels

Table 12. Component `BestPrice`: size 22 bytes

Field	Datatype	Description
price	dec8	Price level
type	int1	Side of orders: • `1` (BUY) • `2` (SELL) • `3` (LAST_TRADE)
pad0	ascii1	Reserved for future. Zero value
amount	int4	Total volume of orders at price level or trade volume
time	time8n	Timestamp of latest update or transaction

2.10. Commons channel

The Commons channel 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 message contains the `CommonEntry` repeating group and each entry describes a parameter. The datatype of the `value` field depends on the `type`. The number of entries is specified in the `CommonEntry_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 13. Snapshot and update parameters correspondence

Parameter type	Update `type` field	Update `value` field
Latest trade price	3	dec8
Opening price	4	dec8
Closing price	5	dec8
Highest price	7	dec8
Lowest price	8	dec8
Closing auction price of previous day	73	dec8
Halt price	74	dec8
Timestamp of lowest current price	75	time8n
Indicative quote	76	dec8
Turnover of trades at closing auction price, in instrument units	79	int8
Turnover for calculation of previous day's market price 3	80	dec2
Turnover for calculation of current day's market price 3	81	dec2
Turnover for calculation of previous day's market price 2	82	dec2
Turnover for calculation of current day's market price 2	83	dec2
Timestamp of current price calculation	84	time8n
Change in current price against official closing price of previous day	85	dec8
Lowest current price	86	dec8
Latest trade price involved in current price	87	dec8
Volume disbalance at closing auction	88	int8
Market price 3 of previous day	89	dec8
Market price 3 of current day	90	dec8
Market price 2 of previous day	91	dec8
Market price 2 of current day	92	dec8
Last trade price of main session in previous day	93	dec8
Last trade price of main session in current day	94	dec8

Parameter type	Update `type` field	Update `value` field
Turnover of latest trade in price currency	95	dec2
Official closing price of previous day	96	dec8
Official current price	97	dec8
Volume weighted average price of main session in previous day	98	dec8
Volume weighted average price of main session in current day	99	dec8
Current price	100	dec8
Settlement price of latest main clearing	101	dec8
Settlement price of latest intermediate clearing	102	dec8
Number of buy orders	103	int8
Number of sell orders	104	int8
Volume of buy orders	105	int8
Volume of sell orders	106	int8
Number of anonymous trades	107	int8
Turnover of anonymous trades, in instrument lots	108	int8
Turnover of anonymous trades, in instrument units	109	int8
Currency turnover of anonymous trades	110	dec2
Total number of trades	111	int8
Total turnover, in instrument lots	112	int8
Total asset turnover	113	int8
Total currency turnover	114	dec2
Closing auction price	115	dec8
Closing auction volume	116	int8
Volume weighted average price	117	dec8
Highest bid price	118	dec8
Lowest ask price	119	dec8
Volume of latest trade	120	int8
Timestamp of latest trade	121	time8n
Price of last trade over previous trading period	122	dec8

Table 14. Parameters unavailable in over-the-counter trades

Parameter type	Update `type` field	Update `value` field
Closing price	5	dec8

Parameter type	Update `type` field	Update `value` field
Halt price	74	dec8
Timestamp of lowest current price	75	time8n
Turnover for calculation of previous day's market price 3	80	dec2
Turnover for calculation of current day's market price 3	81	dec2
Turnover for calculation of previous day's market price 2	82	dec2
Turnover for calculation of current day's market price 2	83	dec2
Timestamp of current price calculation	84	time8n
Change in current price against official closing price of previous day	85	dec8
Lowest current price	86	dec8
Latest trade price involved in current price	87	dec8
Market price 3 of previous day	89	dec8
Market price 3 of current day	90	dec8
Market price 2 of previous day	91	dec8
Market price 2 of current day	92	dec8
Last trade price of main session in current day	94	dec8
Official closing price of previous day	96	dec8
Official current price	97	dec8
Volume weighted average price of main session in previous day	98	dec8
Volume weighted average price of main session in current day	99	dec8

Table 15. Commons (`msgid=1115` in snapshot and `msgid=1113` in updates, dynamic length)

Offset	Field	Datatype	Description
	[frame]	[frame]	Session header
0	[md_header]	[md_header]	Standard header
10	[instrument]	[instrument]	Component for instrument identification
16	CommonEntry_offset	int2	Offset of the first `CommonEntry` entry from the start of this field. Value: `4`
18	CommonEntry_count	int2	Number of `CommonEntry` entries
	> [CommonEntry]	[CommonEntry]	List of commons

Table 16. Component `CommonEntry`: size 10 bytes

Field	Datatype	Description
type	int1	Entry type
pad0	int1	Reserved for future. Zero value
value	int8 / dec8 / dec2	Entry value

2.11. Instruments channel

The Instrument channel broadcasts reference data on trading instruments:

- `Currency` balance instrument,
- `Issue` balance instrument,
- `Spot` balance instrument,
- `Futures` balance instrument,
- `Bond` balance instrument.
- `TradeModes`,
- `Markets`
- trading `Instrument`.

The Instrument snapshot and updates feeds transmits the same messages. The update feed of the Instruments channel broadcasts the `TradingInstrumentsStatus` 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 channel cannot be recovered over TCP.

Table 17. Format of message `Currency`: msgid=931, size=266

Offset	Field	Datatype	Description
	[frame]	[frame]	Session header
0	[md_header]	[md_header]	Topic 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 18. Format of message `Issue`: msgid=932, size=474

Offset	Field	Datatype	Description
	[frame]	[frame]	Session header
0	[md_header]	[md_header]	Topic 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
431	security_type	int1	Security type. Values: • `1` (OrdinaryShare): ordinary share; • `2` (PreferredShare): preferred share; • `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 19. Format of message `Spot`: msgid=933, size=281

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

Offset	Field	Datatype	Description
0	[md_header]	[md_header]	Topic 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 20. Format of message `Futures`: msgid=934, size=280

Offset	Field	Datatype	Description
	[frame]	[frame]	Session header
0	[md_header]	[md_header]	Topic header
10	balance_id	int4	Balance instrument ID
14	code	char32+1	Futures 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	date_expire	time8m	Expiration date
274	underlying_id	int4	Underlying instrument ID
278	exec_type	int1	Futures type. Values: • `0` (FuturesThroughSpot): futures through spot; • `1` (FuturesCashSettlement): cash-settled futures

Offset	Field	Datatype	Description
279	is_test	int1	Flag of test instrument. Values: • `0` (REAL): Real; • `1` (TEST): Test

Table 21. Format of message `Bond`: msgid=935, dynamic length

Offset	Field	Datatype	Description
	[frame]	[frame]	Session header
0	[md_header]	[md_header]	Topic 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 beggining 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
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
444	issue_date	time8m	Date of issue
452	quotation_list	char32+1	Quotation list

Offset	Field	Datatype	Description
485	is_test	int1	Flag of test instrument. Values: • 0 (REAL): Real; • 1 (TEST): Test
	> coupon_payment	[coupon_payment]	Schedule of coupon payments

Table 22. Format of message `TradeModes`: msgid=942, size=210

Offset	Field	Datatype	Description
	[frame]	[frame]	Session header
0	[md_header]	[md_header]	Topic header
10	trade_mode_id	int2	Trade mode ID
12	name	char64+1	Name of trading mode in English
77	name_ru	char128+1	Name of trading mode in Russian
206	is_address	int1	Negotiated trading flag in trading 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 23. Format of message `Market`: msgid=936, size=208

Offset	Field	Datatype	Description
	[frame]	[frame]	Session header
0	[md_header]	[md_header]	Topic header
10	market_id	int4	Market ID
14	desc	char64+1	Full name in English
79	desc_ru	char128+1	Full name in Russian

Table 24. Format of message `Instrument`: msgid=973, dynamic length

Offset	Field	Datatype	Description
	[frame]	[frame]	Session header
0	[md_header]	[md_header]	Topic 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
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
273	fee_rate_offset	int2	Offset of the first `fee_rate` entry from the beggining of this field
275	fee_rate_count	int2	Number of the `fee_rate` group entries

Offset	Field	Datatype	Description
277	curr_price	char16+1	Currency of the instrument price
294	periods_offset	int2	Offset of the first `periods` entry from the beggining 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 beggining of this field
300	exchange_instrument_count	int2	Number of the `exchange_instrument` group entries
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
	> fee_rate	dec8	Fee rate
	> periods	[Period]	Group of trading periods (such as trading session) for instrument
	> exchange_instrument	[ExchangeInstrument]	Group specifying trading instruments at liquidity pools

In this version of the trading platform, 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 .

Table 25. Format of message `TradingInstrumentStatus`: msgid=2031, size=84

Offset	Field	Datatype	Description
	[frame]	[frame]	Session header
0	[md_header]	[md_header]	Topic header
10	instrument	[instrument]	Component specifying trading instrument
16	status	int1	Current status of trading instrument. Values: • `2` (HALT): trading is halted; • `17` (TRADING): instrument is traded; • `18` (NO_TRADING): trading finished or not started yet; • `102` (CLOSE): instrument is traded at closing auction; • `103` (CLOSE_PERIOD): close period trading; • `107` (DISCRETE_AUCTION): instrument is traded at discrete auction; • `118` (OPEN): instrument is traded at opening auction; • `120` (FIXED_PRICE_AUCTION): trading at closing auction price
17	reserved	char2+1	Reserved field. To be filled with null byte
20	comment	char63+1	Comments

Table 26. Format of message `TradingInstrumentLimits`: msgid=2032, size=30

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

Table 27. Format of message `BorrowingStatus`: msgid=2033, size=27

Offset	Field	Datatype	Description
	[frame]	[frame]	Session header
0	[topic_header]	[topic_header]	Topic header
22	instrument_id	int4	Trading instrument identifier
26	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 28. Format of component `coupon_payment`: length 16 bytes

Field	Datatype	Description
date	time8m	Date of payment
value	dec8	Amount of payment

Table 29. Format of component `Period`: length 30 bytes

Field	Datatype	Description
start	time8m	Start timestamp
finish	time8m	End timestamp
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
currency_id	int4	Currency ID of traded instrument
underlying_offset	int2	Offset of the first `underlying` entry from the beggining of this field
underlying_count	int2	Number of the `underlying` group entries
markets_offset	int2	Offset of the first `markets` entry from the beggining of this field
markets_count	int2	Number of the `markets` group entries
> underlying	[Underlying]	Group for trading instrument lot volume specification within a period of time
> markets	int2	List of available liquidity pools (please refer to section 3.6)

Table 30. Format of component `ExchangeInstrument`: length 61 bytes

Field	Datatype	Description
instrument	[instrument]	Component specifying trading instrument
code_group	char16+1	Market section
code	char16+1	Instrument ticker
code_extra	char16+1	Instrument code
status	[instrument_status]	Current status of trading instrument

Table 31. Format of component `instrument_status`: length 4 bytes

Field	Datatype	Description
trading_status	int1	Current status of trading instrument. Values: • `2` (HALT): trading is halted; • `17` (TRADING): instrument is traded; • `18` (NO_TRADING): trading finished or not started yet; • `102` (CLOSE): instrument is traded at closing auction; • `103` (CLOSE_PERIOD): close period trading; • `107` (DISCRETE_AUCTION): instrument is traded at discrete auction; • `118` (OPEN): instrument is traded at opening auction; • `120` (FIXED_PRICE_AUCTION): trading at closing auction price
suspend_status	int1	Reserved field. To be filled with null byte
routing_status	int1	Reserved field. To be filled with null byte
reason	int1	Reserved field. To be filled with null byte

Table 32. Format of component `Underlying`: length 15 bytes

Field	Datatype	Description
balance_id	int4	Balance instrument ID
qty	decn	Number of balance instrument units
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. Market data recovery

The recovery gateway allows the client to request a resend of update messages, if they were lost via UDP. You can request the following channels' data 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 channels. Due to technological limitations messages from the previous trade day can be recovered. For all other channels a client can recover only recent messages.

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

3.1. 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 *Network Connectivity*.

After establishing connection with the Discovery service, the client should send the `Hello` message. The IP address of the client must be authorized for the specified login (user ID); otherwise, the connection request will be rejected. The message contains the session header `frame` (for more details refer to section 3.7.1).

Table 33. Format of request `Hello: msgid=1, size=32, seq=0`

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 group `addresses`; the number of group records will be specified in the field `addresses_count` (for more details on processing of repeating groups please see section 2.5). The group includes fields `type` (gateway attribute) and `addresses` (host address and gateway port). Gateway attributes may combine.

For some time after the trading system response, the gateway will expect the client's login connection to the specified address. In case of failure, the client should 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 34. Format of response `Report: msgid=2, seq=0`, dynamic length

Offset	Field	Datatype	Description
	[frame]	[frame]	Session header
0	status	int2	Request status. Values: • `0` (success), • `1` (reject due to invalid login/password)
2	reason	char127+1	Textual description
130	addresses_offset	int2	Offset of the first underlying entry from the beginning of this field. Value: `4`
132	addresses_count	int2	Number of `addresses` group entries

Offset	Field	Datatype	Description
	> [addresses]	[addresses]	Addresses list

Table 35. Format of component `addresses`: size 52 bytes

Field	Datatype	Description
type	int2	Gateway attributes, bit mask. Values: • `0x0` (No); • `0x1` (Trading); • `0x2` (Drop copy); • `0x4` (Risk management); • `0x8` (Dictionaries); • `0x10` (Market data); • `0x4000` (Backup)
ver	int1	Protocol version
pad0	int1	Reserved field, filled with zero bytes
address	char47+1	Address of host and gateway port

3.2. Sequence numbers

To recover market data messages, the client should connect to the recovery gateway and request a range of messages by sending a `TopicRequest`.

Recovered messages have two numbers—the recovery gate sequence number `seq` and the topic data message number `topic_seq`. The latter equals to the message sequence number of a market data feed.

In a `TopicRequest`, the client should specify the topic identifier (for values refer to *Network Connectivity* document) and the range of requested messages with the first number in the `topic_seq` and the last one in the `topic_seqend`. In the next following versions of the trading platform, the maximum range will be set.

3.3. Recovered message format

The format of a recovered message is identical to that of a broadcast message, except for the header—the `topic_header` stands for the `md_header`. Therefore, offsets of all following fileds are increased by 12 bytes.

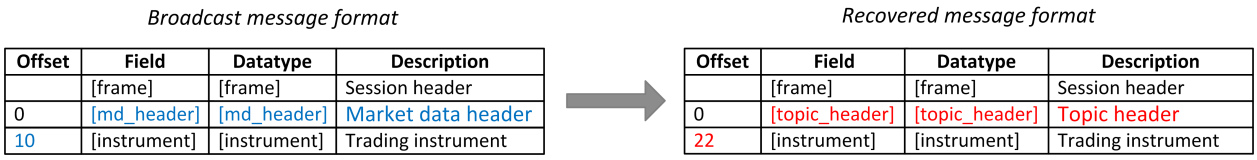

Figure 3. Message format alteration

3.4. Components of recovered messages

Table 36. Format of component `user_header`: length 20 bytes

Field	Datatype	Description
clorder_id	ascii20	Client order ID

Table 37. Format of component `gate_header`: length 46 bytes

Field	Datatype	Description
system_time	time8n	Client request processing time
source_id	int2	Message source (for values please refer to section 3.5)
clorder_id	ascii20	Client order ID
user_id	ascii16	Login, client gateway ID

Table 38. Format of component `topic_header`: length 22 bytes

Field	Datatype	Description
topic_id	int4	Numerical code of topic
topic_seq	int8	Message sequence number in topic
system_time	time8n	Message generation time
source_id	int2	Message source (for values please refer to section 3.5)

3.5. `Source_id` values

An application header includes the `source_id` field that specifies the system module that generated the message.

Table 39. `Source_id` values returned to the client

Range	Description
100–199	Gateway
200–249	Risk management module of the Clearing Center
250–259	Matching module
300–499	Market data sender
500–549	Routing module
1000–1099	Liquidity pool identifiers

3.6. Liquidity pool identifiers

Liquidity pool identifier are values for `source_id`, `market`, `prime_exchange`, and `exec_market`.

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

`1001` (TRADSYS) — all accessible liquidity pools.

`1000` — liquidity pool of Saint-Petersburg Exchange.

`1010` — liquidity pool of Moscow Exchange.

`1015` — execution at US liquidity pools.

`1016` — market data from US liquidity pools.

`1030` — liquidity pool of NYSE.

`1031` — liquidity pool of ARCA.

`1032` — liquidity pool of NASDAQ.

`1033` — liquidity pool of BATS.

3.7. General session layer

3.7.1. Message format

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

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

Table 40. Format of component `frame`: length 12 bytes

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

3.7.2. Session initialization

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

Once connection is established, the client can send the `Login` message to initiate session. The message includes a user ID and a password. The server validates the authentication parameters and answers with the `Logon` message and then 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 receives a second connection attempt via the same login while a valid session is already underway, the server will respond with a `Reject`.

Table 41. 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 42. 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
8	expected_seq	int8	Next application message expected from client
16	system_id	ascii8	Deployment ID

3.7.3. Heartbeats

Heartbeat messages is used to exercise the communication line during periods of inactivity and to verify the interfaces at each end. The server sends a `Heartbeat` anytime it has not transmitted a message for the heartbeat interval. The client is expected to employ the same logic.

The client specifies the heartbeat interval in the `heartbeat_ms` field of the `Logon` message.

If the server detects inactivity for a period longer than the specified heartbeat interval, the server will break the connection with the client. The client is expected to do the same if inactivity is detected on the part of the server.

Table 43. Format of message `HeartBeat`: msgid=8103, size=0

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

3.7.4. Application message sequence numbers

Any application messages have a sequence number unique across a trading day. The client and the server each maintain a separate and independent set of incoming and outgoing message sequence numbers. Sequence numbers should be initialized to 1 (one) at the first session of a trading day and be incremented throughout the session. Monitoring sequence numbers will enable parties to identify and react to missed messages.

Sequence numbers are not assigned to session messages. The `seq` field is assigned zero value for these messages.

3.7.5. Gap fill

The server may respond to the client's `ResendRequest` with a `GapFill` to increase the expected sequence number for missing some messages during resending.

Table 44. Format of message `GapFill`: msgid=8106, size=8

Offset	Field	Datatype	Description
	[frame]	[frame]	Session header
0	next_seq	int8	Next sequence number to be expected by the recipient

3.7.6. Session termination

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

Table 45. Format of message `Logout`: msgid=8002, size=16

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

3.7.7. Message rejection

If a client message is either malformed or contains invalid values, the server rejects this message and responds with the `Reject` message. The `reason` field conveys a code referring the reason for the rejection and the `message` may contain a textual description.

Table 46. Format of message `Reject`: msgid=8102, size=45

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

Offset	Field	Datatype	Description
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

3.7.8. Connection break

The server will break the connection upon receipt of

- an unknown `msgid`,
- a `size` incorrect for the specified message type,
- a `seq` other than expected.

3.8. Subscription management

Subscription management messages do not belong to application level and have no message number `seq` assigned. Nevertheless, messages with subscription data have number `seq`.

Each message with the data generated by the trading platform in a subscription stream is assigned with the `topic_seq` number starting from 1. As the client receives messages in accordance with login access rights, numbering of messages sent to client will be discontinuous.

Subscription management messages have no `topic_seq` number.

3.8.1. Subscription request

In order to request an update subscription or one-time snapshot, the client should send `TopicRequest` to the trading platform gateway specifying `topic` ID and `mode` (snapshot or snapshot and updates). The client does not have to fill the `clorder_id` field.

As for logging data stream, the client may specify the first number of requested data `topic_seq`. For a non-logging stream, `topic_seq` is 0 in a request. When making an initial subscription request for logging stream, the client should use 0, and in a repeating request, the number shall be one more than that of the last message received.

When requesting a logging stream snapshot (`mode=0`), the client may request messages up to a definite number to be specified in the `topic_seqend` field.

When requesting a non-logging stream snapshot, the client (`mode=1`) should specify 0 value in `topic_seqend`.

If a request is incorrect or cannot be processed, the gateway will answer with `TopicReject`. Otherwise, the client will receive `TopicReport` and after that should expect topic data messages. In case of requesting a non-logging stream snapshot, such requested messages may be sent along with messages with updates.

Table 47. 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 code
84	topic_seq	int8	First number of requested topic data
92	topic_seqend	int8	Last number of requested topic data

Offset	Field	Datatype	Description
100	mode	int1	Data request mode. Values: • `0` (DATA_SLICE): snapshot; • `1` (SUBSCRIBE): snapshot and subsequent updates

3.8.2. Unsubscription

The client shall send `TopicCancel` to the trading platform gateway, to stop receiving messages with subscription data, specifying one or both stream identifiers—`topic` and `topic_id`.

If a request is incorrect or cannot be executed, the gateway will respond with `TopicReject`. In case of successful request processing, the subscription will be canceled and the client will receive notification `TopicReport` with `status=2` and the subscription will be terminated; still for some time after notification client may continue receiving messages with data.

Table 48. Format of message `TopicCancel`: msgid=302, size=88

Offset	Field	Datatype	Description
	[frame]	[frame]	Session header
0	[user_header]	[user_header]	Standard header
20	topic	ascii64	Topic code
84	topic_id	int4	Numerical code of topic

3.8.3. Subscribe or unsubscribe rejection

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

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

Table 49. 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 code
110	topic_id	int4	Numerical code of topic
114	status	int2	Subscription status. Values: • `0` (DATA_SLICE): snapshot; • `1` (ACTIVE): active subscription; • `2` (INACTIVE): inactive subscription

Offset	Field	Datatype	Description
116	reason	int2	Reason for rejection. Values: • 1 (BAD_TOPIC): invalid topic identifier; • 2 (ALREADY_SUBSCRIBED): subscription is already active; • 3 (NOT_SUBSCRIBED): subscription not activated; • 4 (DATA_NOT_AVAILABLE): data not available; • 5 (DUPLICATE_REQUEST): repeated request; • 6 (BAD_SEQ): non-existent number in topic; • 7 (BAD_MODE): invalid mode
118	topic_firstseq	int8	Number of first available message
126	topic_lastseq	int8	Last number of topic data
134	topic_lastseqsent	int8	Last number of topic data sent to client

3.8.4. Subscription notification

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

- successful execution of a request for a new subscription;
- successful execution of request for removing a subscription;
- completion of snapshot transmission under a subscription.

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

Table 50. 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 code
110	topic_id	int4	Numerical code of topic
114	status	int2	Subscription mode. Values: • 0 (DATA_SLICE): snapshot; • 1 (ADD_SUBSCRIBE): new subscription; • 2 (DEL_SUBSCRIBE): cancellation of subscription
116	marker	int2	Indicator of start and finish of data transfer. Values: • 0 (START): start of data transfer; • 1 (END): end of the data transfer; • 2 (SLICE_END): snapshot transfer completed
118	topic_lastseq	int8	Last number of message with topic data
126	topic_lastseqsent	int8	Last number of topic data sent to client under subscription

Appendix A. Error codes

Table 51. Error codes list

Code	Description
0	Ok
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.
1200	Invalid segment.

Code	Description
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 declined 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.
1310	User has no permissions to decline this order.
1311	Order currently being replaced.
1312	Order sent before system crash, but received after recovery.

Code	Description
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.
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.
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.

Code	Description
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.
1980	Invalid stages in info field.
2100	Account does not belong to member_id.
2200	No permissions to submit trading instructions.
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.

Code	Description
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.
3100	TCA of maker and that of taker have no conversion bank indicator.
3911	Incorrect te_id.
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 declined 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.
4200	Invalid client for TCA registered at liquidity pool.
4201	Invalid TCA for liquidity pool.
5000	Invalid application message type.

Code	Description
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	Subscription already activated.
5302	Subscription not activated.
5303	Requested data not available.
5304	Another request processing in progress.
5400	Reset_seq indicated, but seqnums cannot be reset.
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 channel 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.