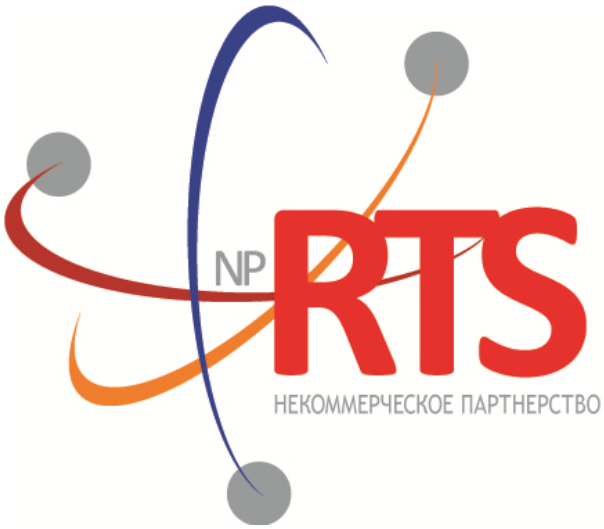

ADVANCED
TRADING & CLEARING
PLATFORM

Native Protocol Transactional Gateway

Interface version 22

Document version 1.2.8

08 July 2022

Revision history

Version 1.2.8 December 6, 2021

1. Updated the list of liquidity pools in Section 3.6 Liquidity pool identifiers.
2. Specified field descriptions in the AddOrder and AddReport messages.
3. Updated information in Section 4.2.2.2 Order execution report.

Version 1.2.7 August 4, 2021

1. Added error code 1405 in Appendix A.
2. Updated the list of liquidity pools in Section 3.6 Liquidity pool identifiers.

Version 1.2.6 February 16, 2021

1. Section 1.2.4 renamed to "Closing auction of Saint-Petersburg Exchange".
2. Value OC of the time_in_force field is updated in AddOrder and AddReport messages.

Version 1.2.5 September 4, 2020

Description in Section 4.1.6 is updated.

Version 1.2.4 July 1, 2020

1. The picture is added to Section 2.4, describing the order cancellation by the liquidity pool.
2. Pictures in Sections 2.2, 2.5, 2.6, 2.7, 2.9 are corrected.
3. The mistake is corrected: the field name reason in Section 4.2.2.3 replaced with the name cancel_reason.

Version 1.2.3 January 24, 2020

1. Added Section 4.1.4, describing client's send rate limit of the session level messages.
2. Added Section 4.2.1.6, describing client's send rate limit of the application level messages.

Version 1.2.2 July 25, 2019 года

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

Version 1.2.1 December 14, 2018

1. Document structure was changed.
2. The subsections of the "Interaction with gateway", "Client requests" and "Trading system reports" sections were renamed.
3. Terminology was changed.

Version 1.2.0 15 February 2018

1. The section "Instruments of trading system" has been added.
2. The sections "Login", "Trading platform gateways" and "Instruction and order reports discrimination" have been removed.
3. Terminology is changed.
4. Error codes were added.

Version 1.1.7 3 April 2017

Values Day and XH of field time_in_force is corrected in messages AddOrder and AddReport.

Version 1.1.0 22 September 2016

1. New value X of field TimeInForce is added to messages AddOrder and AddReport.
2. New values 1030, 1031, 1032, 1033 of field ExchangeSpecialInstructions is added to messages AddOrder and AddReport.

Table of Contents

List of Tables

1. Trading system overview

The trading system is designed to allow users to perform operations on financial markets. The main functions include:

1. Acceptance of orders submitted to over-the-counter and exchange markets.
2. Routing and placing of orders in available liquidity pools.
3. Registration of trades and processing of information on trades at liquidity pools.
4. Transmission of anonymous market data, collected from all liquidity pools, and non-anonymous market data as well as additional and reference data.
5. Control of clearing member's risks on operations with instruments registered in the system.
6. Other functionality for access to trading.

1.1. Instruments of trading system

The Instruments are divided into **exchange** and **over-the-counter (OTC)**. OTC instruments have the following attributes:

- Field `section` in `Instruments` messages has value **OTC**.
- Field `over_the_counter` in `TradeModes` messages has value **1**.
- Field `flags` has value 0x400000 (eOverTheCounter).

Table 1. Differences in the interpretation of messages fields

Instrument	Value of `order_id` field	Value of `deal_id` field
Exchange	Order ID	Trade ID
Over-the-counter	Tender ID	Contract ID

All instruments of trading system are available for trades.

1.2. Trade modes

1.2.1. Main trades mode

In the main trades mode anonymous orders are executed at liquidity pools.

The Main trades mode supports five order types. The order type is determined by the set of field values in the message.

1.2.1.1. Order types

1. Market order that will execute at the best available prices until it is fully filled; any remainder will be expired.
2. Day limit order that will execute at the specified or better price; the remainder, if any, is added to the order book and will be active till the end of the trading day.
3. Extended session limit order that will execute at the specified or better price; the remainder, if any, is added to the order book and will be active till the end of the extended trading session.
4. Fill or Kill (FOK) order that will execute immediately and completely, or canceled. This is an order with specified price and volume.
5. Immediate or Cancel (IOC) order that execute immediately, completely or partially, or canceled. This is an order with specified price and volume.

The set of order types available in the trading system may differ from the set of order types supported by a specific liquidity pool.

1.2.1.2. Execution of orders

For a group of instruments listed on the trading system, the **Main pool** is determined among several liquidity pools by the highest liquidity level. The Main liquidity pool status may influence the choice of routing strategy: by default the volume that cannot be matched against active orders in the order book will be routed to that pool.

A client order, submitted to the trading system, can be executed at liquidity pools where the indicated instrument is admitted to trading. If there is only one liquidity pool matching this criterion, the entire orders volume is routed to that pool. If there are several liquidity pools like that, the order will be executed in accordance with the best execution principles.

In the course of routing, the incoming order is consecutively matched with counter orders at each price level until the order volume is filled. If all the available price levels were checked and the incoming order has not been filled completely, the remaining volume is routed to the Main liquidity pool. After the volumes to be routed are determined, they are sent to the liquidity pools.

Routing of client order depends on the order type.

A Fill Or Kill order can be filled at one liquidity pool only, where the order initiator can get the best average weighted price; in case of several equal prices the trading system give the priority to the pool providing a lower latency.

An incoming order of other types (limit, market, Immediate Or Cancel) can be routed to several liquidity pools. For each price level consecutively, starting from the best one for the order initiator, the volume to be executed is determined on each available pool. After the volumes to be routed are determined, they are sent to the appropriate price levels to the liquidity pools.

1.2.2. Negotiated trades mode

The Negotiated trades mode supports negotiated orders with fully matching parameters.

Negotiated order is an order with an indication of price, volume, initiator and counterparty.

The counterparty is notified that order is submitted on its clearing account (for more details on interaction with trading gateway, refer to Section 2).

1.2.3. Negotiated repo trades mode

Price of order for repo trades is indicated in annual interest rate. In additional price field the client can indicate the price of the first-leg instrument. If client did not indicate a price, the additional price will be settled or will be indicated by the liquidity pool.

Repo trading instrument has three legs (balance instruments):

1. Change in the obligation to deliver securities under the first part of repo trade.
2. Change in the obligation to deliver currency under the first part of repo trade.
3. Change in the obligation to deliver securities under the second part of repo trade.

Currency obligation under the second part of repo trade is changed using the price setting tool for repo trading instrument.

1.2.4. Closing auction of SPB Exchange

The closing auction of SPB Exchange supports only market order with time in force - closing auction. Trades are executed at the official closing price of the instrument of the liquidity pool, on which the security was listed. Orders, leading to cross trade, will be automatically canceled by the liquidity pool.

Trading in the closing auction:

1. During the trading day, clients submit market orders in the trading system.
2. Submission of orders is stopped according to the approved schedule of trading and orders become unavailable to cancel.
3. Closing auction is held — counter orders, sorted by ascending of the time of submission, are matched together at instrument's closing price at Main liquidity pool.
4. Remainders of orders and unfilled orders are canceled.

2. Interaction with gateway

2.1. Order submission and rejection

The client submits a new order by sending the `AddOrder` message to the trading system. For each order, the client should provide the client identifier `clorder_id` unique for a login across a trading day.

After accepting the request, the trading system returns `AddReport` to the client with the `order_id` assigned. If the system rejects the request (due to an invalid value or a closed market), the order identifier will not be provided and the client will receive `RejectReport`.

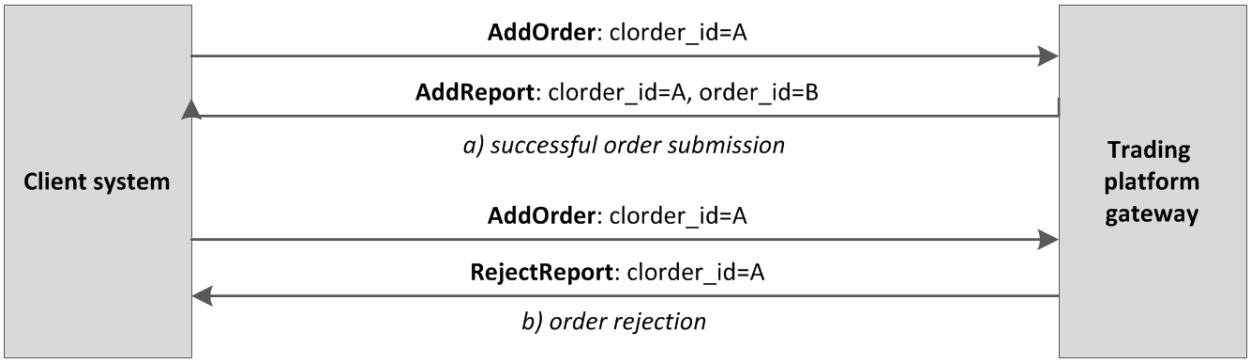

Figure 1. Order submission and rejection

2.2. Order placement and routing rejection

To ensure best execution, the order volume is split according to the current state of the order book and then it is routed to liquidity pools. After the liquidity pool reports successful routing, the gateway will send `AddReport` to the client with the `market_id` assigned.

If the liquidity pool reports unsuccessful routing, the trading system will return `RejectReport` to the client, and will cancel a portion of the order that equals to the rejected volume and will notify the client with the `CancelReport`.

A Fill Or Kill order can be routed to one liquidity pool only. If the pool reports successful routing, the gateway will send reports as described above. If the pool reports unsuccessful routing, the trading system will return `RejectReport` to the client, and will cancel the order and will notify the client with the `CancelReport`.

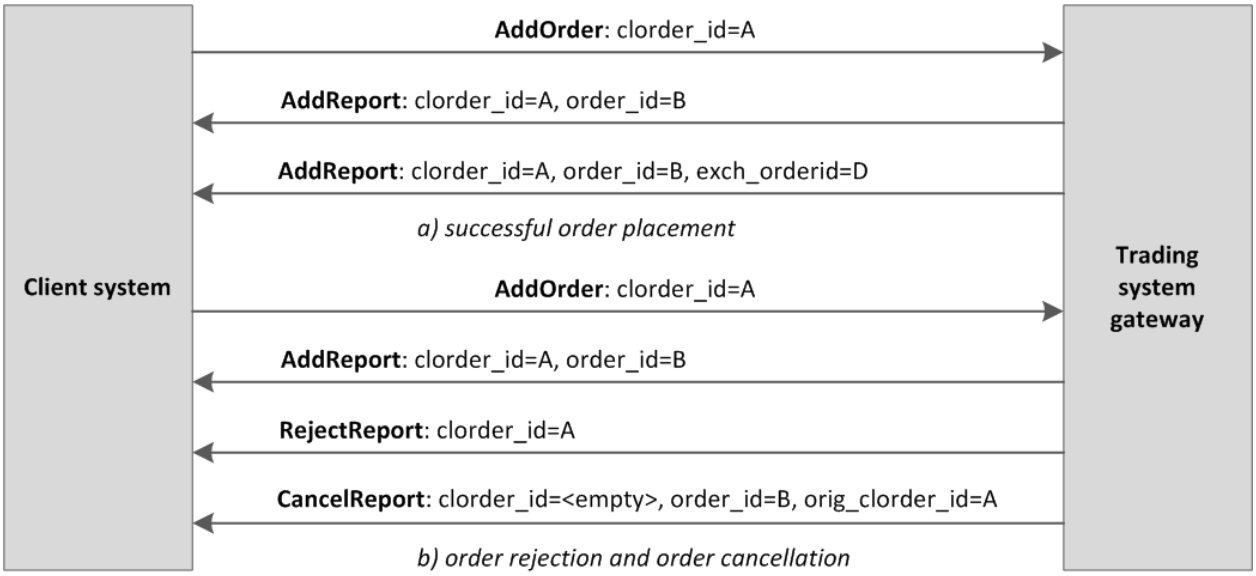

Figure 2. Order placement or rejection

2.3. Order execution

After a trade is done, the client will receive `Execution` reports (on execution at liquidity pool and changing amount of order in the trading system). Trades done within a single transaction, i.e. a sequence of concurrent trades of an incoming order, are included in one or more consequential reports with the active remainder after the trades in the `amount_rest` field and the trades information in the `deals` component.

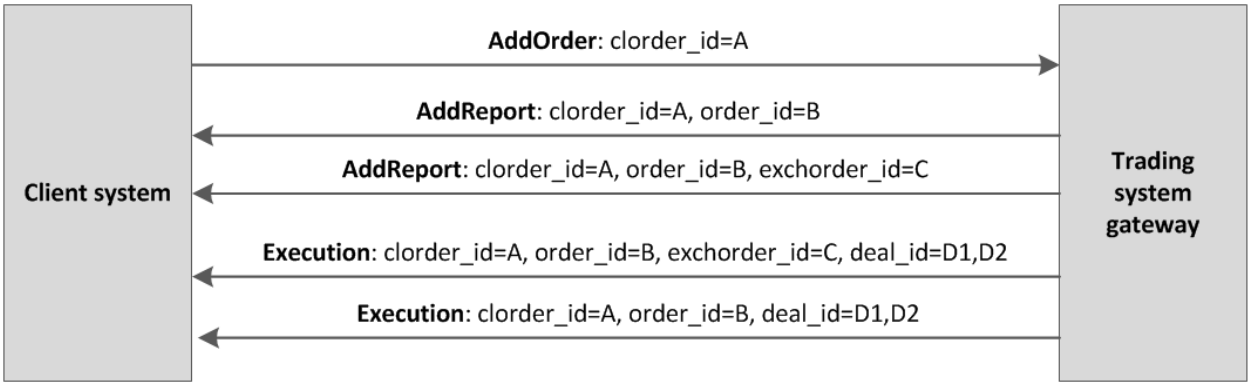

Figure 3. Order submission and execution

2.4. Order cancellation by the liquidity pool

Under some conditions, a liquidity pool cancels an order remainder, e.g. unfilled portion of a market or IOC order, or to prevent a cross trade. In this case a client will receive the `CancelReport` about full or partial cancellation after the `AddReports` and `Execution` reports.

Moreover, to ensure best execution, the trading system may cancel an order at a liquidity pool and place it to another. Then, after an `AddOrder` or `Execution` report, the client should also expect a `CancelReport` and another `AddOrder`.

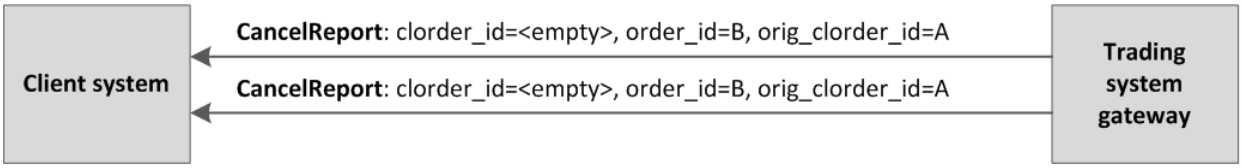

Figure 4. Order cancellation by the liquidity pool

2.5. Order cancellation by the client

The placed order can be canceled only in full volume. The part of the order cannot be canceled.

The client can cancel an active remainder of an order. To request an order cancellation the client should send the `CancelOrder` message to the server with an order identifier specified.

When the order successfully cancelled, the client will receive the followings `CancelReports` — reports on routed volumes cancellation and then report on order cancellation.

If the trading system expects a response from the liquidity pool for a considerable time, the client will receive `RejectReport` with `Pending cancel` in the `message` field.

If an order cannot be cancelled or the request originator does not have permissions, the request will be rejected with `RejectReport`.

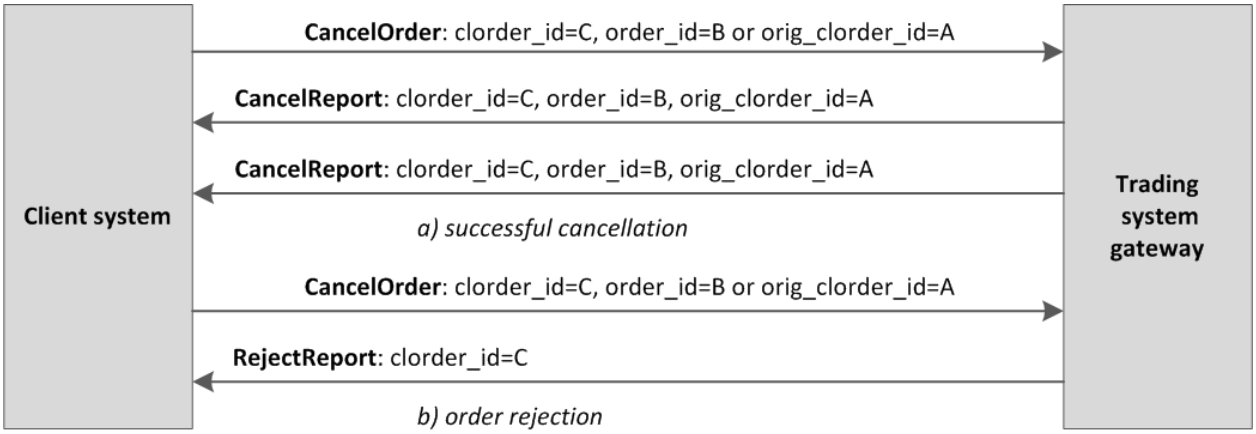

Figure 5. Order cancellation by the client

2.6. Order mass cancellation

The client can cancel a set of orders selected by a specicifc parameter: an instrument, a user, etc. To cancel a set of orders, client should send the `MassCancel` request with specified cancellation mode and, if relevant, order parameters.

The trading system receives the request and selects orders to cancel by the defined criteria, and then generates cancel requests and routes them to liquidity pools. If the orders are successfully cancelled, the gateway will send `CancelReport` reports on each order cancellation. The number of cancelled orders is specified in the `MassCancelReport` message that notifies of the request completion. If no order to cancel is found, the gateway will only return `MassCancelReport`.

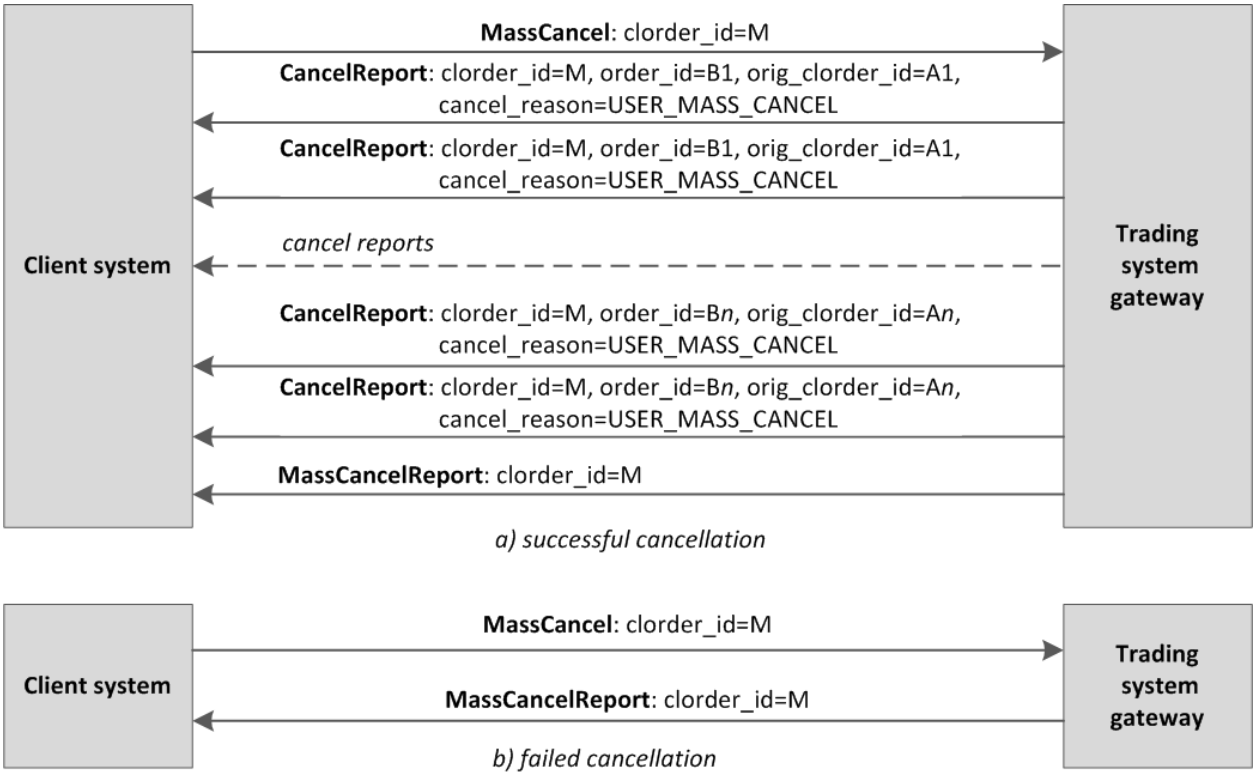

Figure 6. Order mass cancellation

2.7. Negotiated order submission and cancellation

The client submits a negotiated order via the `AddOrder` message with `type=NEGOTIATED`. A negotiated order should contain the `initiator_party` identifier of the order initiator and the counterparty identifier `ctrparty`. The initiator may enter an identifier for order matching in the `match_ref` field.

Similar to an anonymous order described above, the client will receive `AddReport` when the trading system accepts a negotiated order and then the liquidity pool accepts the order, or a `RejectReport`, if the system rejects the order (for more details please refer to section 2.1).

After the order acceptance, the trading system will notify the counterparty with `CounterReport`.

Until the counterparty submits the counter order, the initiator can cancel the order by sending the `CancelOrder` request to the gateway with any order identifier specified. When the order is cancelled, the trading system will send `CancelReport` to the order initiator and `CounterUpdateReport` to the counterparty.

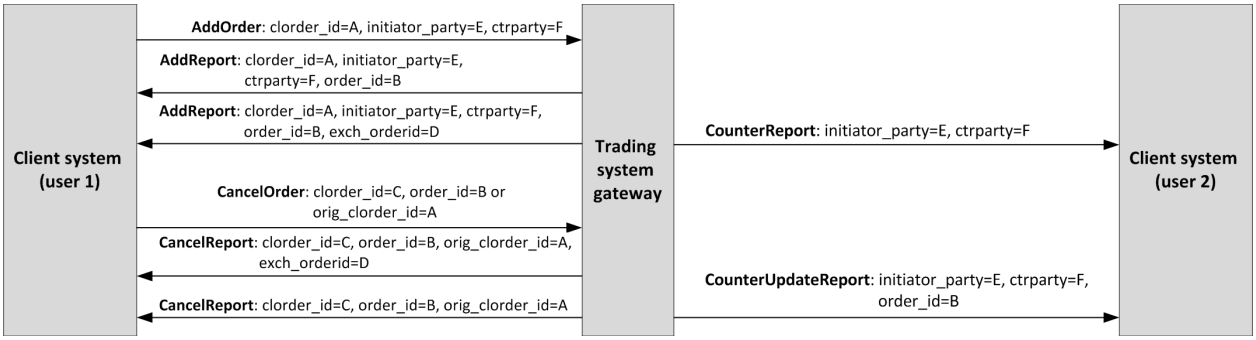

Figure 7. Negotiated order submission and cancellation

2.8. Negotiated counter order placement

To close a trade, the counterparty should submit an order of the same instrument and quantity at the same price with the opposite direction and counterparties.

If the price, quantity, instrument, order direction or counterpaties of counter and original order mismatch, then the counter order will be placed as a new one.

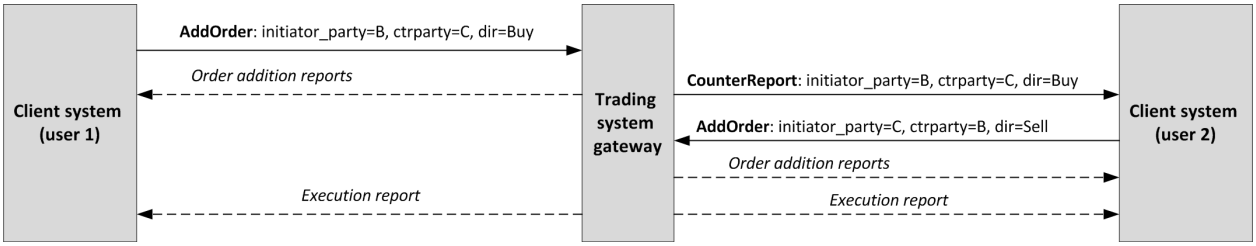

Figure 8. Negotiated counter order placement

2.9. Negotiated order rejection by the counterparty

The counterparty can reject an active negotiated order by sending the `CounterDecline` request to the gateway.

When the order is cancelled, the trading system will send `CounterDeclineReport` (first, with `market_id=1000`, then with `market_id=1001`) and `CounterUpdateReport` to the `CounterDecline` request originator and `CancelReport` to the order initiator.

If an order cannot be cancelled, the request will be rejected with `RejectReport`.

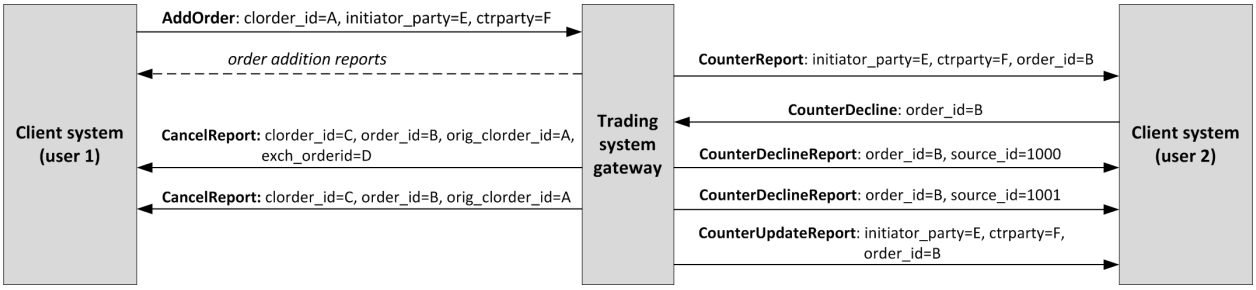

Figure 9. Negotiated order rejection

3. Protocol overview

3.1. 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*. 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.

3.2. 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.

3.3. Common components of messages

Table 2. 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

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

Field	Datatype	Description
market_id	int2	Liquidity pool ID. Values: `1000` and `1001`
instrument_id	int4	Trading instrument ID

Table 4. Format of component `user_header`: length 20 bytes

Field	Datatype	Description
clorder_id	ascii20	Client order ID

Table 5. 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

Field `user_id` can be empty, for example if order is automatically canceled by trading system.

Table 6. Format of component `account`: length 36 bytes

Field	Datatype	Description
member_id	int4	Trading member ID
account	ascii16	Clearing account ID
client_id	ascii16	Client code ID

Table 7. Format of component `deal`: length 20 bytes

Field	Datatype	Description
deal_price	dec8	Trade price
deal_id	int8	Trade ID assigned by liquidity pool
amount	int4	Trade volume

Table 8. Format of component `otccodes`: length 32 bytes

Field	Datatype	Description
initiator_party	ascii16	Negotiated order sender ID
ctrparty	ascii16	Negotiated order recipient ID

3.4. 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.

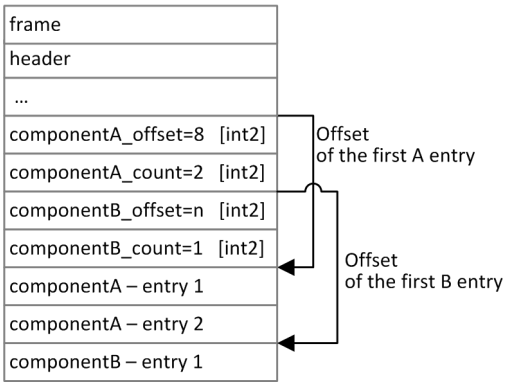

Figure 10. Template of a message with two repeating components

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

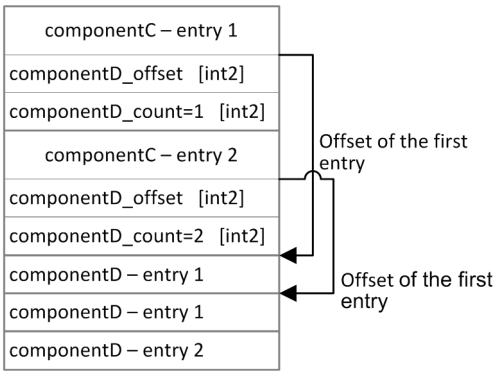

Figure 11. Template of embedded components

3.5. `Source_id` values

Field `source_id` is in the header gate_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
1000–1099	Liquidity pools identifiers

3.6. Liquidity pool identifiers

Liquidity pools' identifiers may be in fields `market`, `market_id`, `source_id`, `exec_market` and `prime_exchange` .

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

`1001` (TRADSYS) — all available liquidity pools

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

4. Protocol specification

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 3.2). The client should specify login and password, and the IP address of the client must be authorized for the specified login (user ID).

Table 10. 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 3.4). 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 11. Format of message `Report`: msgid=2, dynamic length

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
	> addresses	[Report_Address]	Address list

Table 12. Format of component `Report_Address`: length 52 bytes

Field	Datatype	Description
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
ver	int1	Interface version
pad0	int1	Reserved field, filled with zero bytes
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 13. 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 14. 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

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 15. 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`).

The gateway accumulates messages addressed to the client even when no connection established. If the `last_seq` filed is greater than the last message received during the previous session, the client should request not received messages via the `ResendRequest`.

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. Message resend request

If the client's system has not been connected to the gateway for some time, the gateway may accumulate messages intended for the client, but not received by him. In order to be convinced of the presence of such messages, it is necessary to compare the `seq` number of the last received message with the `last_seq` number in the `Logon` message. If the numbers are different, the client should use the `ResendRequest` message to retrieve the missed messages.

The client can request missed messages sent during the current and previous trading days. If the client forcefully resets the message numbering (`reset_seq = 1` in the `Login` message), then the request for missed messages which were sent prior to this reset is not possible.

The `ResendRequest` message must contain the number of the first message in the `from_seq` field and the number of the last message in the `till_seq` field within the requested messages range. Possible request parameters are listed below:

1. `from_seq=n, till_seq=m` — request for messages from *n* to *m* but not exceeding the maximum available number.
2. `from_seq=0, till_seq=n` — request for messages from the lowest number available to *n* but not exceeding the maximum available number.
3. `from_seq=n, till_seq=0` — request for messages from *n* to the last number available but not exceeding the maximum available number.

4. `from_seq=0, till_seq=0` — request for all available messages but not exceeding the maximum available number.

The number of requested messages in one request cannot exceed the specified value (for more details, refer to document *Network Connectivity*, Section 1.3). To request more messages, the client should send multiple consecutive `ResendRequest` messages.

Table 16. Format of message `ResendRequest`: msgid=8005, size=16

Offset	Field	Datatype	Description
	[frame]	[frame]	Session header
0	from_seq	int8	First requested message
8	till_seq	int8	Last requested message

In response to a correctly formed request, the trading system will transmit requested messages, preceding the sending by the `ResendReport` message with the `ACK` status. When messages are complete, the gateway will send the `ResendReport` message with status `MORE` or `FINISH`. The `MORE` status means that the number of the last message within the requested messages range is less than the number of the last message sent by the gateway. That is, there are messages that are not included in the request output. They could have been generated during the request execution, or the number of messages in one request exceeded the specified value. In this case, another `Resend Request` message should be made.

If the recovery of missed messages is performed by means of several consecutive `Resend Request` messages, each subsequent request should be performed after receiving all messages of the previous request. Otherwise, it will be rejected by the `ResendReport` message with the `DUPLICATE_REQUEST` status.

When connecting for the first time in the current trading day, it is recommended to use a request with parameters `from_seq = -1, till_seq = 0`. If, after sending, the gateway returns the `ResendReport` message with the `MORE` status, you should send another request, indicating in the `from_seq` field a number one more than the last one forwarded message, and `till_seq = 0`.

To recover missed messages after reconnection, you must send a request with the parameters `from_seq = n, till_seq = s`, where *n* is the number of the last received message before the connection was terminated plus one, and *s* is the number of the last message available to the client (`last_seq` field) received in the `Logon` message. If, after sending, the gateway returns the `ResendReport` message with the `MORE` status and the client has not yet received messages with the specified numbers, another request should be sent, indicating in the `from_seq` field a number one more than that of the last forwarded message, and `till_seq = s`.

The ResendRequest is processed by the gateway in parallel with the sending of current messages. That is, the client can receive both missed messages and messages sent after connecting. The client system must independently restore the correct order of received messages based on their seq numbers.

Table 17. Format of message `ResendReport`: msgid=8105, size=2

Offset	Field	Datatype	Description
	[frame]	[frame]	Session header
0	status	int2	Request status. Values: • `0` (ACK): gateway is ready to respond to a request; • `1` (MORE): gateway executed the query and still has data for client; • `2` (FINISH): all available data sent to the client; • `3` (DUPLICATE_REQUEST): server busy with the previous `ResendRequest`; • `4` (UNAVAILABLE): recovery service unavailable

4.1.7. Message numbers reset by the client

The client may change the number of expected message at the gateway. For this purpose, the client should send `SequenceReset` specifying next message number in the `next_seq` field. At that, the new number shall not be less than the current value at the gateway.

Table 18. Format of message `SequenceReset`: msgid=8004, size=8

Offset	Field	Datatype	Description
	[frame]	[frame]	Session header
0	next_seq	int8	Next sequence number expected from client

4.1.8. Session termination

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

Table 19. Format of message `Logout`: msgid=8002, size=16

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

4.1.9. 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 20. 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

4.1.10. 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.

4.2. Application layer

4.2.1. Client requests

4.2.1.1. Order submission

The client submits a new order by sending the message `AddOrder` with required fields.

Table 21. Required fields depending on the order types

Order type	Required fields	
Market	clorder_id market_id instrument_id dir routing_instruction routing_dest amount member_id account client_id prime_exchange	type=MARKET, time_in_force=IOC
Market order executed at the closing auction price		type=MARKET, time_in_force=OC
Limit order executed at the closing auction price		type=LIMIT, time_in_force=OC, price
Day active limit		type=LIMIT, time_in_force=Day, price
Limit order in extended trading session		type=LIMIT, time_in_force=XH, price
Fill or Kill (FOK)		type=LIMIT, time_in_force=FOK, price
Immediate or Cancel (IOC)		type=LIMIT, time_in_force=IOC, price
Negotiated		initiator_party, ctrparty, type=NEGOTIATED, time_in_force=Day, price

The `price` should be entered as an integer with last eight implied decimal places, e.g. 123,45 is specified as 12345000000. The price is an integer multiple of the step price (refer to *Instrument reference data*).

The client should provide a new order with a client order ID (`clorder`) which has to be unique for a user across a trading day and does not repeat identifier of the rejected order.

The negotiated order initiator may enter a `match_ref`, and the counterpatry will have to use the same value, otherwise the two orders will not match.

The client may provide a `comment` (a 23 byte string in UTF-8).

At the end of a trading session or extended trading session, all active orders (`time_in_force=Day` or `time_in_force=XH`) will be cancelled and the client will receive `CancelReport` with the `cancel_reason=EXPIRED`.

After processing a message, the trading system either rejects the message with `RejectReport` or confirms the acceptance with `AddReport`.

Table 22. Format of message `AddOrder`: msgid=101, size=194

Offset	Field	Datatype	Description
	[frame]	[frame]	Session header
0	[user_header]	[user_header]	Standard header
20	instrument	[instrument]	Component specifying trading instrument
26	dir	int1	Side. Values: • `1` (Buy): buy; • `2` (Sell): sell
27	type	int1	Order type. Values: • `1` (MARKET): market; • `2` (LIMIT): limit; • `103` (NEGOTIATED): negotiated

Offset	Field	Datatype	Description
28	time_in_force	int1	Order time in force. Values: • 0 (Day): day; • 2 (OO): opening auction; • 3 (IOC): immediate or cancel; • 4 (FOK): fill or kill; • 7 (OC): order execution at the closing auction price; • 100 (XH): during the extended trading session
29	passive_only	int1	Reserved field. Filled with zero.
30	auto_cancel	int1	Cancel order on logout/disconnection. Values: • 0 (OFF): no cancel; • 1 (AUTO_CANCEL): cancel on logout/disconnection
31	order_segment	int1	Reserved field. Filled with zero
32	routing_instruction	int2	Routing order for the remainder. Value - 0 (passive routing)
34	routing_dest	int2	Liquidity pool ID. Value - 1001
36	amount	int4	Order amount
40	amount_extra	int4	Reserved field. Filled with zero
44	price	dec8	Price, or annual percentage yield for repo only
52	price_extra	dec8	Extra price. Trade price for repo only
60	flags	int8	Order matching parameters. Value: 0x2000 (eIgnoreDynamicLimits): ignoring dynamic limits, available only for logins with flag CAN_IGNORE_DYNAMIC_LIMITS
68	time_valid	time8n	Deadline for order acceptance
76	date_expire	time4	Order expiration timestamp. Zero value
80	account	[account]	Component specifying client submitted order
116	parties	[otccodes]	Component specifying counterparties
148	comment	char23+1	Client comment
172	extra_ref	ascii12	Additional order ID
184	extra1	ascii4	Reserved field. Filled with zero
188	prime_exchange	int2	Main liquidity pool. For a description of values, refer to Section 3.6. Values: • 0 (DEFAULT); • 1000 (SPB)

Offset	Field	Datatype	Description
190	match_ref	int4	Optional identifier for negotiated orders matching

4.2.1.2. Order cancellation

The client may cancel active remainder of an order by sending the `CancelOrder` message. An order to cancel should contain required fields.

Table 23. Required fields depending on the cancel modes

Cancel mode	Required fields	
Canceling of order by request of order originator login	`clorder_id` `instrument_id`	`orig_clorder_id` (or `order_id`)

Canceling of order by request of login other than the order originator	`account`
	`client_id`
	`dir`
	`type`
		`order_id`

After processing a request, the trading system either rejects the message with `RejectReport` or confirms the cancellation with `CancelReport`. If the system rejects a request for cancellation by `order_id`, it will be specified in the `extra_data0` field of `RejectReport` message.

Table 24. Format of message `CancelOrder`: msgid=112, size=100

Offset	Field	Datatype	Description
	[frame]	[frame]	Session header
0	[user_header]	[user_header]	Standard header
20	instrument	[instrument]	Component specifying trading instrument
26	dir	int1	Side. Values: • `1` (Buy): buy; • `2` (Sell): sell
27	type	int1	Order type. Values: • `1` (MARKET): market; • `2` (LIMIT): limit; • `103` (NEGOTIATED): negotiated
28	order_id	int8	Order ID assigned by the trading system
36	account	[account]	Component specifying client submitted order
72	flags	int8	Flags.
80	orig_clorder_id	ascii20	Client ID of order to cancel

4.2.1.3. Order mass cancellation

The client may mass cancel orders via the `MassCancel` request with the `mode` assumed.

The client should provide the unique identifier `clorder_id` for the command. The value should not start with `onlogout_`.

Value of the `mode` field specifies mass cancel mode. Fields of the `MassCancel` message should be filled according to selected mode.

Table 25. Required fields depending on the mass cancel modes

Mass cancel mode	Required fields
Canceling of all orders on request of order originator login	`clorder_id`, `mode=7`
Canceling of all orders of an instrument on request of order originator login	`clorder_id`, `mode=23`, `instrument_id`, `market_id`
Canceling of all orders by specifying an instrument and a clearing account	`clorder_id`, `mode=39`, `instrument_id`, `market_id`, `account`
Canceling of all orders by specifying an instrument and a client ID	`clorder_id`, `mode=55`, `instrument_id`, `market_id`, `client_id`

The value of `clorder_id` field schould not start with `onlogout_`.

When applying the BY_LOGIN mode, the client should not fill the fields `instrument_id` and `market_id`.

After processing the request, the trading system will confirm each order cancellation with `CancelReport` and will send `MassCancelReport` upon completion.

Table 26. Format of message `MassCancel`: msgid=103, size=63

Offset	Field	Datatype	Description
	[frame]	[frame]	Session header
0	[user_header]	[user_header]	Standard header
20	instrument	[instrument]	Component specifying trading instrument
26	mode	int1	Mass cancel mode. Values: • `7` (BY_LOGIN): canceling all orders submitted by the user which is the request originator; • `23` (BY_INSTR_LOGIN): cancelling all orders of an instrument submitted by the user which is the request originator; • `39` (BY_INSTR_ACCOUNT): cancelling all orders of an instrument and an account; • `55` (BY_INSTR_CLIENT): cancelling all orders of an instrument and a client ID
27	account	[account]	Component specifying trading member, account, and client ID

4.2.1.4. Automatic order cancellation

Active remainders of order submitted by the login can be canceled at disconnection or session termination. For this the client should set value `AUTO_CANCEL` in `auto_cancel` field of the `AddOrder` messages. When orders will be automatically canceled, the trading system will send `CancelReport` with `reason=DISCONNECT` to the client. Upon reconnection, the client will receive `CancelReport` reports and `MassCancelReport` with the `clorder_id` starting with `onlogout_`.

4.2.1.5. Negotiated order rejection by the counterparty

The counterparty may reject a negotiated order via `CounterDecline` that should contains `clorder_id` assigned by the order originator, the counterparties identifiers `initiator_party` and `ctrparty_id`, and, if filled in the original order, the `match_ref`.

After processing the request, the trading system will confirm order rejection with `CounterDeclineReport` or will reject it with `RejectReport`.

Table 27. Format of message `CounterDecline`: msgid=105, size=72

Offset	Field	Datatype	Description
	[frame]	[frame]	Session header
0	[user_header]	[user_header]	Standard header
20	instrument	[instrument]	Component specifying trading instrument
26	dir	int1	Side. Values: • `1` (Buy): buy; • `2` (Sell): sell
27	type	int1	Order type. Value: `103` (NEGOTIATED): negotiated
28	parties	[otccodes]	Component specifying counterparties
60	order_id	int8	Order ID assigned by the trading system
68	match_ref	int4	Identifier for negotiated orders matching

4.2.1.6. Send rate limit for client requests

The rate at which a client can send requests into the system is limited. There are two limits:

1. When the first threshold of send rate is reached, the system starts declining the application level requests and transmits the report on rejecting request with reason "Number of messages exceeded limit".
2. When the second threshold of send rate is reached, the system terminates the user session.

4.2.2. Trading system reports

4.2.2.1. Order addition and routing report

When the client's order is successfully accepted, the trading system sends the `AddReport`. The report contains all parameters of `AddOrder` including counterparties identifiers `initiator_party` and `ctrparty`, the code for order matching `match_ref` and also the client order identifier `order_id` assigned by the trading system and unique across a trading session.

The trading system routes orders to liquidity pools and expects replies after that. If a liquidity pool accepts an order, the client will be sent `AddReport` with parameters of routed amount and the `market_id`. If a liquidity pool rejected an order, the trading system will generate `RejectReport`.

Table 28. Format of message `AddReport`: msgid=212, size=260

Offset	Field	Datatype	Description
	[frame]	[frame]	Session header
0	[gate_header]	[gate_header]	Standard header
46	instrument	[instrument]	Component specifying trading instrument

Offset	Field	Datatype	Description
52	dir	int1	Side. Values: • 1 (Buy): buy; • 2 (Sell): sell
53	type	int1	Order type. Values: • 1 (MARKET): market; • 2 (LIMIT): limit; • 103 (NEGOTIATED): negotiated; • 104 (OUT_OF_BOOK): out of book
54	time_in_force	int1	Order time in force. Values: • 0 (Day): day; • 2 (OO): opening auction; • 3 (IOC): immediate or cancel; • 4 (FOK): fill or kill; • 7 (OC): order execution at the closing auction price; • 100 (XH): during the extended trading session
55	passive_only	int1	Reserved field. Filled with zero.
56	auto_cancel	int1	Cancel order on logout/disconnection. Values: • 0 (OFF): no cancel; • 1 (AUTO_CANCEL): cancel on logout/disconnection
57	order_segment	int1	Set of client IDs; the order will only match orders submitted by a client ID of this set
58	routing_instruction	int2	Routing order for the remainder
60	routing_dest	int2	Liquidity pool ID. Value - 1001
62	amount	int4	Order amount
66	amount_extra	int4	Disclosed amount
70	price	dec8	Price, or annual percentage yield for repo only
78	price_extra	dec8	Trade price for repo only

Offset	Field	Datatype	Description
86	flags	int8	Order matching parameters. Values: • `0x1` (eUserLastRec): last transaction message: last report on trades executed within a single transaction; • `0x2` (eMMObligations): indicator of market maker executing the obligations at internal exchange, to be assigned to visible limit orders; • `0x4` (eNoMMTrade): indicator of order by market maker that not to be matched with another market maker's order; • `0x8` (ePresettlement): pre-delivery trade; • `0x10` (eExternalActivity): transaction executed through external interfaces; • `0x20` (eDelivery): delivery trade; • `0x40` (eDeliverySwapGood): transfer of a bona fide participant during delivery; • `0x80` (eDeliverySwapBad): transfer of a mala fide participant during delivery; • `0x100` (eDeliveryDonorTrade): delivery transfer of participant with donor involvement; • `0x200` (eNoSystem): negotiated trade indicator; • `0x2000` (eIgnoreDynamicLimits): ignoring dynamic limits; • `0x100000` (eClientPartialExecute): partial execution of address order sent by the client; • `0x200000` (eHaltPeriodOrder): marker of issuing an order during a suspension period; • `0x400000` (eOverTheCounter): marker of an order or a deal with over-the-counter instrument; • `0x800000` (eRFQ): request for quote mode indicator
94	date_expire	time4	Order expiration timestamp
98	time_valid	time8n	Deadline for order acceptance
106	account	[account]	Component specifying client submitted order
142	parties	[otccodes]	Component specifying counterparties
174	order_id	int8	Order ID assigned by the trading system
182	orig_orderid	int8	Order ID on the previous trading day
190	exch_orderid	ascii20	Child order ID at liquidity pool
210	price_entry	int1	Price level the order is placed at. Not processed in the current system version
211	pad1	ascii1	Reserved field. Filled with zero
212	comment	char23+1	Client comment

Offset	Field	Datatype	Description
236	extra_ref	ascii12	Additional order ID
248	extra1	ascii4	Additional textual field
252	prime_exchange	int2	Main liquidity pool. For a description of values, refer to Section 3.6.
254	match_ref	int4	Optional ID for negotiated orders matching
258	orig_market	int2	Liquidity pool specified by the client at submission (for a description of values, refer to Section 3.6)

4.2.2.2. Order execution report

When a trade matching is done, the trading system will send the `Execution` report to the client. A report contains details of one or more trades made at the same liquidity pool. The liquidity pool is specified in the `exec_market` field (for a description of values, refer to Section 3.6).

Reports about trades executed at liquidity pools with identifiers not equal to 1000 and 1001, contain values type=104 (OUT_OF_BOOK) and exec_market=1000.

Trade details (price, amount, and unique identifier) are listed in the `deals` component. The number of entries (i.e. number of trades being reported) is specified in the `deals_count` field. The size of the `Execution` message is dynamic and depends on the number of `deals` entries (for details of processing such messages, refer to Section 3.4).

Table 29. Format of message `Execution`: msgid=207, dynamic length

Offset	Field	Datatype	Description
	[frame]	[frame]	Session header
0	[gate_header]	[gate_header]	Standard header
46	instrument	[instrument]	Component specifying trading instrument
52	dir	int1	Side. Values: • `1` (Buy): buy; • `2` (Sell): sell
53	type	int1	Order type. Values: • `1` (MARKET): market; • `2` (LIMIT): limit; • `103` (NEGOTIATED): negotiated; • `104` (OUT_OF_BOOK): out of book
54	price	dec8	Price, or annual percentage yield for repo only
62	price_extra	dec8	Offering price, or trade price for repo only

Offset	Field	Datatype	Description
70	flags	int8	Order matching parameters. Values: • `0x1` (eUserLastRec): last transaction message: last report on trades executed within a single transaction; • `0x2` (eMMObligations): indicator of market maker executing the obligations at internal exchange, to be assigned to visible limit orders; • `0x4` (eNoMMTrade): indicator of order by market maker that not to be matched with another market maker's order; • `0x8` (ePresettlement): pre-delivery trade; • `0x10` (eExternalActivity): transaction executed through external interfaces; • `0x20` (eDelivery): delivery trade; • `0x40` (eDeliverySwapGood): transfer of a bona fide participant during delivery; • `0x80` (eDeliverySwapBad): transfer of a mala fide participant during delivery; • `0x100` (eDeliveryDonorTrade): delivery transfer of participant with donor involvement; • `0x200` (eNoSystem): negotiated trade indicator; • `0x2000` (eIgnoreDynamicLimits): ignoring dynamic limits; • `0x100000` (eClientPartialExecute): partial execution of address order sent by the client; • `0x200000` (eHaltPeriodOrder): marker of issuing an order during a suspension period; • `0x400000` (eOverTheCounter): marker of an order or a deal with over-the-counter instrument; • `0x800000` (eRFQ): request for quote mode indicator
78	exec_market	int2	Liquidity pool of execution (for a description of values, refer to Section 3.6)
80	account	[account]	Component specifying client submitted order
116	parties	[otccodes]	Component for specifying counterparties
148	order_id	int8	Order ID assigned by the trading system
156	exch_orderid	ascii20	Child order ID at liquidity pool
176	amount_rest	int4	Remainder amount
180	deals_offset	int2	Offset of the first `deals` entry from the beginning of this field
182	deals_count	int2	Number of the `deals` group entries
	> deals	[deal]	Trade information

4.2.2.3. Order cancellation report

When an order is successfully canceled, the trading system will send `CancelReport`. The report contains order parameters, order IDs `order_id` and `orig_clorder_id`, and reason of cancellation in the `cancel_reason` field.

Liquidity pool has a several reasons for automatically cancel a part of order (see table below).

Table 30. Reasons for automatically cancel

Value of `cancel_reason` field	Order type	Reason for cancel
`cancel_reason=EXPIRED_NOTRADES`	Immediate Or Cancel	Order is partially filled
`cancel_reason=EXPIRED_CROSSTRADE`	All types	Part of the order that can lead to a cross trade. The rest of the order will be executed as usual, including other trades at the cross trade price level.
`cancel_reason=EXPIRED_ORDERBOOK_CROSS`		Part of the order that can lead to a crossed order book.

Table 31. Format of message `CancelReport`: msgid=214, size=172

Offset	Field	Datatype	Description
	[frame]	[frame]	Session header
0	[gate_header]	[gate_header]	Standard header
46	instrument	[instrument]	Component specifying trading instrument
52	dir	int1	Side. Values: • `1` (Buy): buy; • `2` (Sell): sell
53	type	int1	Order type. Values: • `1` (MARKET): market; • `2` (LIMIT): limit; • `103` (NEGOTIATED): negotiated; • `104` (OUT_OF_BOOK): out of book
54	amount	int4	Canceled amount
58	amount_rest	int4	Remainder amount
62	price	dec8	Price, or annual percentage yield for repo only
70	price_extra	dec8	Offering price, or trade price for repo only

Offset	Field	Datatype	Description
78	flags	int8	Order matching parameters. Values: • `0x1` (eUserLastRec): last transaction message: last report on trades executed within a single transaction; • `0x2` (eMMObligations): indicator of market maker executing the obligations at internal exchange, to be assigned to visible limit orders; • `0x4` (eNoMMTrade): indicator of order by market maker that not to be matched with another market maker's order; • `0x8` (ePresettlement): pre-delivery trade; • `0x10` (eExternalActivity): transaction executed through external interfaces; • `0x20` (eDelivery): delivery trade; • `0x40` (eDeliverySwapGood): transfer of a bona fide participant during delivery; • `0x80` (eDeliverySwapBad): transfer of a mala fide participant during delivery; • `0x100` (eDeliveryDonorTrade): delivery transfer of participant with donor involvement; • `0x200` (eNoSystem): negotiated trade indicator; • `0x2000` (eIgnoreDynamicLimits): ignoring dynamic limits; • `0x100000` (eClientPartialExecute): partial execution of address order sent by the client; • `0x200000` (eHaltPeriodOrder): marker of issuing an order during a suspension period; • `0x400000` (eOverTheCounter): marker of an order or a deal with over-the-counter instrument; • `0x800000` (eRFQ): request for quote mode indicator
86	account	[account]	Component specifying client submitted order
122	order_id	int8	Order ID assigned by the trading system
130	exch_orderid	ascii20	Child order ID at liquidity pool

Offset	Field	Datatype	Description
150	cancel_reason	int2	Cancel reason. Values: • 0 (USER_CANCEL): canceled on a user's CancelOrder request; • 1 (USER_MASS_CANCEL): canceled on a user's MassCancelOrder request; • 2 (BROKER_CANCEL): canceled on the firm's CancelOrder request; • 4 (BROKER_MASS_CANCEL): canceled on the firm's MassCancelOrder request; • 5 (DISCONNECT): canceled on disconnection; • 6 (EXPIRED): canceled upon expiration; • 8 (OPERATOR): canceled by the trading system administrator; • 9 (EXPIRED_NOTRADES): Immediate Or Cancel remainder cancellation; • 10 (EXPIRED_CROSSTRADE): canceled to prevent a cross trade; • 11 (EXPIRED_ORDERBOOK_CROSS): canceled to prevent a crossed order book; • 12 (CTRPARTY_DECLINE): canceled on the counterparty's CounterDecline request; • 14 (FILLED): negotiated order matching; • 15 (EXT_REJECTED): canceled on rejection by liquidity pool; • 16 (EXT_EXPIRED): canceled on expiration at liquidity pool
152	orig_clorder_id	ascii20	Optional client ID of order to cancel

4.2.2.4. Order mass cancellation report

The trading system will respond to the MassCancel request with MassCancelReport. The result of cancellation is specified in the cancel_status field.

If the request resulted in one or more orders cancellation, the trading system will send each order's CancelReport before the MassCancelReport. The orders can be canceled in any sequence.

Table 32. Format of message MassCancelReport: msgid=206, size=94

Offset	Field	Datatype	Description
	[frame]	[frame]	Session header
0	[gate_header]	[gate_header]	Standard header
46	instrument	[instrument]	Component specifying trading instrument

Offset	Field	Datatype	Description
52	mode	int1	Mass cancel mode. Values: • `7` (BY_LOGIN): canceling all orders submitted by the user which is the request originator; • `23` (BY_INSTR_LOGIN): canceling all orders of an instrument submitted by the user which is the request originator; • `39` (BY_INSTR_ACCOUNT): canceling all orders of an instrument and an account; • `55` (BY_INSTR_CLIENT): canceling all orders of an instrument and a client ID
53	account	[account]	Component specifying trading member, account, and client ID
89	cancel_reason	int2	Field reserved for future. Zero byte value.
91	num_orders	int2	Number of canceled orders
93	cancel_status	int1	Mass cancellation result. Values: • `0` (NOTHING_TO_CANCEL): no orders to cancel found; • `1` (CANCELED_OK): one or more orders canceled; • `2` (CANCEL_FAILED): undefined status of one or more orders

4.2.2.5. Order rejection report

The trading system rejects an application request with `RejectReport` message in the following cases:

- Request does not correspond to the Login permission.
- Request contains an invalid value.
- Request cannot be processed (for example, due to a closed market).

The `reason` will be specified in `RejectReport` and the `message` field will have a textual description or rejection parameters.

Table 33. Format of message `RejectReport`: msgid=201, size=91

Offset	Field	Datatype	Description
	[frame]	[frame]	Session header
0	[gate_header]	[gate_header]	Standard header
46	market	int2	Liquidity pool rejecting client's order (for a description of values, refer to Section 3.6)
48	reason	int2	Code of rejection reason
50	message	char32+1	Rejection code parameters or textual description of the rejection reason
83	extra_data0	int8	Order ID, when request to cancel was identified by order_id

4.2.2.6. Negotiated counter order report

When the client's negotiated order is successfully accepted, the trading system notifies the counterparty with `CounterReport`. The report contains parameters of `AddOrder` and also the client order identifier `order_id` assigned by the trading system unique across a trading day.

Table 34. Format of message `CounterReport`: msgid=203, size=122

Offset	Field	Datatype	Description
	[frame]	[frame]	Session header
0	[gate_header]	[gate_header]	Standard header
46	instrument	[instrument]	Component specifying trading instrument
52	dir	int1	Side. Values: • `1` (Buy): buy; • `2` (Sell): sell
53	type	int1	Order type. Value: `103` (NEGOTIATED): negotiated
54	amount	int4	Amount
58	price	dec8	Price, or annual percentage yield for repo only
66	price_extra	dec8	Extra price. Trade price for repo only

Offset	Field	Datatype	Description
74	flags	int8	Order matching parameters. Values: • `0x1` (eUserLastRec): last transaction message: last report on trades executed within a single transaction; • `0x2` (eMMObligations): indicator of market maker executing the obligations at internal exchange, to be assigned to visible limit orders; • `0x4` (eNoMMTrade): indicator of order by market maker that not to be matched with another market maker's order; • `0x8` (ePresettlement): pre-delivery trade; • `0x10` (eExternalActivity): transaction executed through external interfaces; • `0x20` (eDelivery): delivery trade; • `0x40` (eDeliverySwapGood): transfer of a bona fide participant during delivery; • `0x80` (eDeliverySwapBad): transfer of a mala fide participant during delivery; • `0x100` (eDeliveryDonorTrade): delivery transfer of participant with donor involvement; • `0x200` (eNoSystem): negotiated trade indicator; • `0x2000` (eIgnoreDynamicLimits): ignoring dynamic limits; • `0x100000` (eClientPartialExecute): partial execution of address order sent by the client; • `0x200000` (eHaltPeriodOrder): marker of issuing an order during a suspension period; • `0x400000` (eOverTheCounter): marker of an order or a deal with over-the-counter instrument; • `0x800000` (eRFQ): request for quote mode indicator
82	parties	[otccodes]	Component specifying counterparties
114	order_id	int8	Order ID assigned by the trading system

4.2.2.7. Negotiated order execution and cancellation report

After an order successfully executed or canceled by the `CancelOrder` or `CounterDecline` requests, the trading system will send `CounterUpdateReport` to the client. The report contains parameters of `AddOrder`, order IDs `order_id` and `clorder_id`, counterparties' IDs `initiator_party` and `ctrparty`.

Table 35. Format of message `CounterUpdateReport`: msgid=209, size=123

Offset	Field	Datatype	Description
	[frame]	[frame]	Session header
0	[gate_header]	[gate_header]	Standard header
46	instrument	[instrument]	Component specifying trading instrument

Offset	Field	Datatype	Description
52	dir	int1	Side. Values: • `1` (Buy): buy; • `2` (Sell): sell
53	type	int1	Order type. Value: `103` (NEGOTIATED): negotiated
54	amount_rest	int4	Remainder amount
58	price	dec8	Price, or annual percentage yield for repo only
66	price_extra	dec8	Extra price. Trade price for repo only
74	flags	int8	Order matching parameters. Values: • `0x1` (eUserLastRec): last transaction message: last report on trades executed within a single transaction; • `0x2` (eMMObligations): indicator of market maker executing the obligations at internal exchange, to be assigned to visible limit orders; • `0x4` (eNoMMTrade): indicator of order by market maker that not to be matched with another market maker's order; • `0x8` (ePresettlement): pre-delivery trade; • `0x10` (eExternalActivity): transaction executed through external interfaces; • `0x20` (eDelivery): delivery trade; • `0x40` (eDeliverySwapGood): transfer of a bona fide participant during delivery; • `0x80` (eDeliverySwapBad): transfer of a mala fide participant during delivery; • `0x100` (eDeliveryDonorTrade): delivery transfer of participant with donor involvement; • `0x200` (eNoSystem): negotiated trade indicator; • `0x2000` (eIgnoreDynamicLimits): ignoring dynamic limits; • `0x100000` (eClientPartialExecute): partial execution of address order sent by the client; • `0x200000` (eHaltPeriodOrder): marker of issuing an order during a suspension period; • `0x400000` (eOverTheCounter): marker of an order or a deal with over-the-counter instrument; • `0x800000` (eRFQ): request for quote mode indicator
82	parties	[otccodes]	Component specifying counterparties
114	order_id	int8	Order ID assigned by the trading system

Offset	Field	Datatype	Description
122	reason	int1	Cancel reason. Values: • `0` (USER_CANCEL): canceled on a user's `CancelOrder` request; • `12` (CTRPARTY_DECLINE): canceled on the counterparty's `CounterDecline` request; • `14` (FILLED): negotiated order matching

4.2.2.8. Negotiated order rejection report

After a negotiated order is successfully rejected, the trading system will send `CounterDeclineReport` to the counterparty. The report contains parameters of `AddOrder`, order IDs `order_id` and `clorder_id`, counterparties' IDs `initiator_party` and `ctrparty`.

Table 36. Format of message `CounterDeclineReport`: msgid=208, size=94

Offset	Field	Datatype	Description
	[frame]	[frame]	Session header
0	[gate_header]	[gate_header]	Standard header
46	instrument	[instrument]	Component specifying trading instrument
52	dir	int1	Side. Values: • `1` (Buy): buy; • `2` (Sell): sell
53	type	int1	Order type. Value: `103` (NEGOTIATED): negotiated
54	parties	[otccodes]	Component specifying counterparties
86	order_id	int8	Order ID assigned by the trading system

Appendix A. Error codes

Table 37. 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.
1116	Invalid level.

Code	Description
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.
1310	User has no permissions to reject this order.

Code	Description
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.
1720	Incorrect limit for limited margin.

Code	Description
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.
2100	Account does not belong to member_id.

Code	Description
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.
4000	ECN not available or no liquidity pool available.

Code	Description
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.
5304	Another request processing in progress.

Code	Description
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.