Moscow Exchange

SIMBA ASTS Market Data service
User Guide

---

Moscow Exchange
Version 1.01
August 18, 2022

# Table of contents

# A. Document History

| Date | Version | Changes |
|---|---|---|
| 14.11.2021 | 0.1 | Original version |
| 18.04.2022 | 1.0 | MktBidSize and MktOfferSize fields were added to the BesPrices message to indicate number of lots at best prices |
| 03.08.2022 | 1.01 | Added Trade message (msg id =16) to a list of message types for OLR channel |

# 1. Introduction

## 1.1. Document purpose

This document describes the protocol of marketdata publication for Moscow Exchange (MOEX) Equities and Currency markets, based on Simple Binary Encoding (SBE) technology. Specification covers session, presentation, and application levels of protocol. It does not cover administrative and networking aspects of service.

## 1.2. Targeted audience

This document is targeted to business analysts, system architects and software developers participating in building solutions that receive market information from Moscow Exchange over SIMBA ASTS marketdata service.

## 1.3. Terms and definitions

Following terms and abbreviations are in use in this document:

| Abbreviation | Definition |
|---|---|
| OLR | Order List Refresh |
| IDF | Instrument Definitions Feed |
| ISF | Instrument Status Feed |
| MTU | Maximum Transmission Unit |
| NBP | New Best Prices (best bid price & best ask price) |
| SBE | Simple Binary Encoding |
| UDP | User Datagram Protocol |

## 1.4. SIMBA ASTS Gateway definition

SIMBA ASTS Gateway – low latency publisher of market information from MOEX Equites and Currency trading systems. The service publishes updates to best prices and updates of a list of active orders, like OLR channel of running since 2011 FAST UDP multicast marketdata service.
Updates are published as FIX messages encoded in SBE format using group delivery via the UDP multicast technology.
SIMBA ASTS protocol is based on FIX Simple Binary Encoding (https://www.fixtrading.org/standards/sbe-online) technology; it is assumed that reader of this document already knows basic principles of this protocol. SIMBA ASTS protocol includes presentation, session, and application levels.

### 1.4.1. Data channels

**Main incremental channel**. SIMBA ASTS Gateway publishes updates to a list of active orders as pairs of identical incremental messages: one message is sent to the Incremental Feed A, another to the Incremental Feed B. Two multicast groups are used for reliability, because UDP protocol does not guarantee message delivery. Incremental messages in use are BestPrices (msg id=3), EmptyBook (msg id=4), OrderUpdate (msg id=5), OrderExecution (msg id=6), Heartbeat (msg id=1). Messages are inserted in incremental network packets (see section 2.3.1, "Incremental Packet format).

**Recovery channel**. SIMBA ASTS Gateway published periodical snapshots of a list of active orders as pairs of identical snapshot messages: one is sent to multicast group for Snapshot Feed A, another to Snapshot Feed B. Snapshot channel messages in use are OrderBookSnapshot (msg id=7), Heartbeat (msg id=1). Messages are inserted in Snapshot network packets (see section 2.3.2, "Snapshot packet format). Snapshot channel network traffic is limited by traffic shaper.

**TCP Replay subsystem**. SIMBA ASTS Gateway allows retrieving copies of missed incremental packets over TCP connection. TCP Replay channel uses messages of following types: Logon (msg id=1000), Logout (msg id=1001), MarketDataRequest (msg id=1002).

### 1.4.2. Security definition and trading status channels

SIMBA ASTS Gateway includes Instrument Definitions (IDF) channel, which periodically publishes FIX SecurityDefinition (msg id=8) encoded in SBE format. Single Security Definition message contains instrument description and key parameters and includes trading session and instrument trading status fields. Network traffic of IDF channel is limited by traffic shaper.

SIMBA ASTS Gateway includes incremental channel for publishing Instrument trading status updates, ISF, which publishes changes to security trading modes. The channel uses messages of types SecurityStatus (msg id=9), TradingSessionStatus (msg id=11), Heartbeat (msg id=1) encoded in SBE format. Data is published in pairs of identical messages: one message is sent to the Incremental Feed **A**, another to the Incremental Feed **B**. Two multicast groups are used for reliability, because UDP protocol does not guarantee message delivery.

### 1.4.3. Messages and channel

Table below lists all message types and channels

| Channel name | Channel Type | Message Name |
|---|---|---|
| OLR | Incremental | Heartbeat (msg id=1)<br>SequenceReset (msg id=2)<br>BestPrices (msg id=3)<br>EmptyBook (msg id=4)<br>OrderUpdate (msg id=5)<br>OrderExecution (msg id=6)<br>Trade (msg id=16) |
| OLS | Snapshot | Heartbeat (msg id=1)<br>SequenceReset (msg id=2)<br>OrderBookSnapshot (msg id=7) |
| TCP replay | Sending copies of missed OLR packets by request | From client to gateway:<br>Logon (msg id=1000)<br>Logout (msg id=1001)<br>MarketDataRequest (msg id=1002)<br>From gateway to client:<br>Heartbeat (msg id=1) |

| | | |
|---|---|---|
| | | BestPrices (msg id=3)<br>EmptyBook (msg id=4)<br>OrderUpdate (msg id=5)<br>OrderExecution (msg id=6)<br>Trade (msg id=16) |
| IDF | Insstrument Definitions | Heartbeat (msg id=1)<br>SequenceReset (msg id=2)<br>SecurityDefinition (msg id=8) |
| ISF | Instrument Status Incremental | Heartbeat (msg id=1)<br>SequenceReset (msg id=2)<br>SecurityStatus (msg id=9)<br>TradingSessionStatus (msg id=11) |

### 1.4.4. New Best Prices (NBP) publication priority

Trading instructions and system events (below – transactions) lead to changes in a list of active orders and may result in trades. Processing of transaction is an atomic action which result becomes available for publishing only after all Trading System tables are updated. Multiple changes to a list of active orders are published by
SIMBA ASTS Gateway in a sequence, so it is necessary to get and process all messages before you get resulting state of active orders and new best prices. In some cases, single order may result in very long, up to thousands of passive orders execution messages.
Publishing of New Best Prices (NBP) messages in a separate packet prior to OLR packets for a result of transaction helps to quickly evaluate the final state of exchange orderbook.
NBP updates are published as BestPrices (msg id=3) messages in OLR Incremental Feed **A** and Incremental Feed **B** channel.
Important feature of NBP publishing:

- NBP is published only for transactions that change at least one best price value. Volume change at the same best price level is not published.
- If transaction results in empty best price in one or both directions, then it is indicated by Null price value.

### 1.4.5. Sequence numbers

Incremental or Snapshot feeds use packet sequence number counters. New packet is assigned incremented by one sequence number of preceding packet of a feed.
Incremental packets sequence numbers are always reset to 1 at SIMBA ASTS gateway start or re-start.
Snapshot packets are assigned sequence number 1 at the beginning of each snapshot cycle.

### 1.4.6. Data fragmentation between packets

If data size exceeds the MTU size value, then SIMBA ASTS gateway splits data along multiple packets. All packets that do not contain the last fraction of data per transaction are marked by a flag LastFragment=0 in Marketdata packet header. A packet that contains ending or data per transaction is marked by the LastFragment=1 flag.

### 1.4.7. Snapshot data fragmentation over multiple packets

The first packet in a sequence of packets with data for a shapshot of active orders per security OrderBookSnapshot (msg id=7) is market by the StartOfSnapshot=1 flag in Market Data Packet Header. Last packet containing data for a shapshot per security is marked by the EndOfSnapshot=1 flag in

Market Data Packet Header. If entire snapshot data fit in single packet, then it is marked by both StartOfSnapshot=1 и EndOfSnapshot=1 flags. These flags are helpful if you need to start building a list of active orders per selected securities not waiting for the beginning of new snapshot cycle.

## 1.5 Late join and recovery

SIMBA ASTS Gateway provides the following ways of building correct list of active orders and recovering missed messages:

- Getting orderbook state by listening the snapshot channel and applying changes from the incremental channel. Useful for late connection to feed.
- Getting missed packets via requests in separate TCP replay session. This method has limitations (see section 4.2.2, recovering missed packets with TCP Replay) and can be used for limited data volumes.

### 1.5.1. Orderbook state recovery with snapshots

Snapshot feed messages can be used to build current orderbook state after connection failure or in case of late connection to the feed.
This process implies the queuing the Incremental Market Data from Incremental Feeds while receiving Market Data Snapshots from Recovery Feeds. To avoid an excessive number of queued messages, it is recommended to process snapshots and apply the applicable incremental feed as the snapshots arrive.

1. Identify Feed(s) in which the client system is out of sync.
2. Listen to and queue the Incremental Market Data from the affected Feed(s).
3. Listen to the Market Recovery Feed corresponding to the affected Incremental Feed(s), receive and apply snapshots.
4. Verify that all snapshots have been received for a given Market Recovery feed, using one of the following approaches:
   - Message sequence numbers in each loop of snapshots start from 1. So, to determine the end of the loop one can wait until the next message with 34-MsgSeqNum = 1 arrives.
   - Snapshots in the Recovery Feeds are sent per combinations of Board and SecurityID fields. Flags StartOfSnapshot=1 and EndOfSnapshot=1 in packet header indicate the first and the last packets containing snapshot to a given security. Receiving all messages in packets per instrument from StartOfSnapshot=1 and EndOfSnapshot=1 ensures getting full snapshot for the instrument.
5. Apply all queued incremental data in the sequence, where:

- tag 34-MsgSeqNum is greater than the lowest value for tag 369-LastMsgSeqNumProcessed for a given instrument;

OR

- tag 83-RptSeq from the Market Data Incremental – Refresh (X) message is greater than the lowest value for tag 83-RptSeq from the Market Recovery feed for a given instrument.

6. Continue normal processing.

# 2. Presentation level

## 2.1. FIX syntax

SIMBA ASTS message types, structure, field names and field types follow the FIX standard: http://fiximate.fixtrading.org/.

## 2.2. SBE format

Messages are encoded using Simple Binary Encoding (SBE) standard version 1: https://www.fixtrading.org/standards/sbe-online/.
SIMBA ASTS message schemas are places in files at publicly available MOEX resource at address http://ftp.moex.com/pub/SIMBA/ASTS/schemas/ . If you detect discrepancies between this document message descriptions and xml schemas, please treat xml schemas as definitely correct.

## 2.3. Packet structure

Market data is published as messages included in network packets.

### 2.3.1. Incremental packet format

Each packet is composed from the following parts:

- Market Data Packet Header
- Incremental Packet Header.
- One or more SBE messages, each containing the following parts:
- SBE Header
- FIX message encoded with SBE.

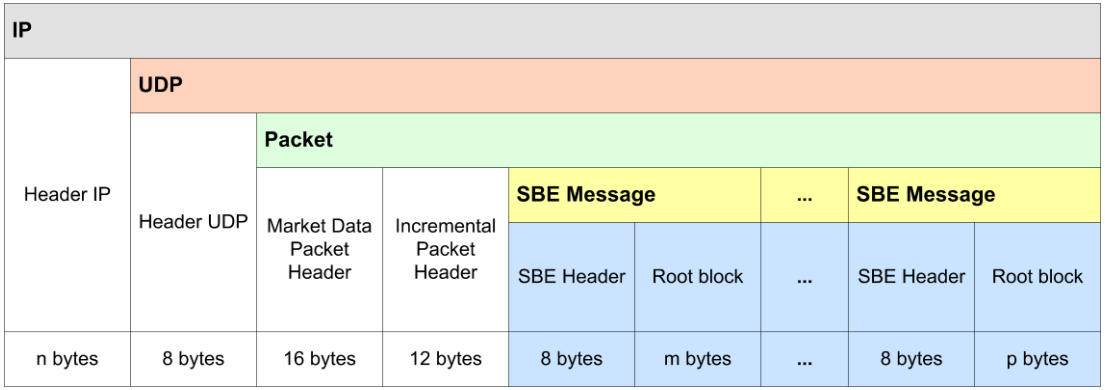


*Figure 1. Incremental packet structure*

### 2.3.2. Snapshot packet format

Each snapshot packet is composed from the following parts:

- Market Data Packet Header
- SBE message Header.
- FIX message encoded with SBE.

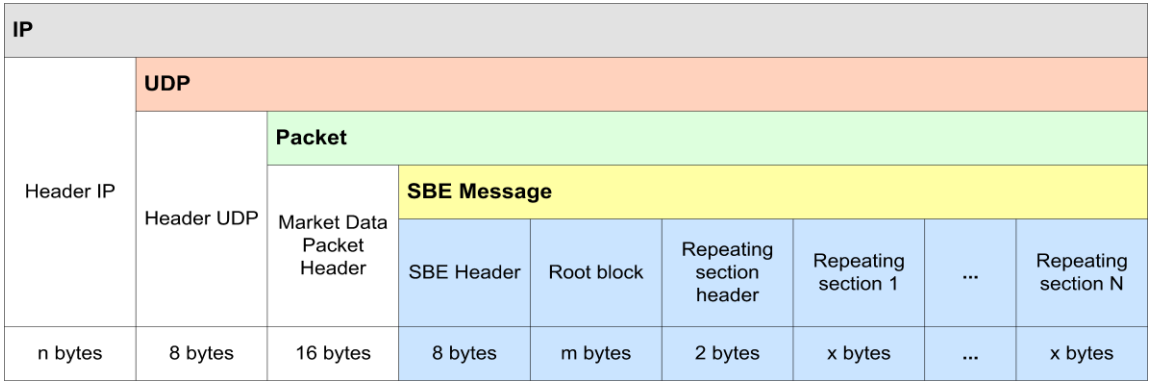


*Figure 2. Snapshot packet structure*

### 2.3.3. Market Data Packet Header

Market Data Packet Header contains packet sending time, sequence number, packet size field and a field with packet flags. Little-endian byte order is used in packet header.

| Field name | Type | Length | Required | Description |
|---|---|---|---|---|
| MsgSeqNum | uInt32 | 4 | Y | Packet sequence number. Next packet is assigned sequence number incremented by one. Incremental Packets sequence numbers are always reset to 1 at SIMBA ASTS gateway start or re-start.<br>Snapshot packets are assigned sequence number 1 at beginning of each snapshot cycle. |
| MsgSize | uInt16 | 2 | Y | Packet length in bytes, including packet header length. |
| MsgFlags | uInt16 | 2 | Y | Packet flags:<br>0x1 – Last data fragment (LastFragment): 0 – not the last data fragment, expect continuation in next packet; 1 – last data fragment, next packet will contain data for later event.<br>0x2 – indicates the first packet in snapshot per instrument (StartOfSnapshot);<br>0x4 – indicates the first packet in snapshot per instrument (EndOfSnapshot);<br>0x8 – IncrementalPacket flag: 0 – indicates the Snapshot packet, 1 – indicates the Incremental packet. |
| SendingTime | uInt64 | 8 | Y | Timestamp of sending packet. Number of nanoseconds since Unix epoch, UTC. |

### 2.3.4. Incremental Packet Header

Incremental Packet Header contains transaction registration time at Matching Engine and an ID of trading session. Little-endian byte order is used in this header.

| Tag | Field name | Type and size | Bytes | Required | Comments |
|---|---|---|---|---|---|
| 60 | TransactTime | uInt64 | 8 | Y | Timestamp of transaction. Number of nanoseconds since Unix epoch, UTC. |
| 5842 | Exchange TradingSessionID | Int32 | 4 | Y | Management field. Changing value within trading day has a meaning of Trading System full restart. |

### 2.3.5. SBE Header

SBE Header contains message's root part length field, template ID field, message schema ID and version.

| Field | Type and size | Bytes | Required | Comments |
|---|---|---|---|---|
| BlockLength | uInt16 | 2 | Y | A length of root part of message in bytes. Does not include SBE header, NoMDEntries field and repeating group of fields. |
| TemplateID | uInt16 | 2 | Y | Message Template ID. |
| SchemaID | uInt16 | 2 | Y | Message schema ID. |
| Version | uInt16 | 2 | Y | Schema version. |

## 2.4. Data types

Data types are listed below. Data types are also specified in xml files located at public MOEX resource at address
http://ftp.moex.com/pub/SBE/ASTS/xml/
If you detect discrepancies between this document data types descriptions and descriptions in xml schemas, please treat xml schemas as definitely correct.

### 2.4.1. Integer types

```
    <type name="uInt8" primitiveType="uint8"/>
    <type name="uInt8NULL" presence="optional" nullValue="255" primitiveType="uint8"/>
    <type name="uInt16" primitiveType="uint16"/>
    <type name="uInt16NULL" presence="optional" nullValue="65535" primitiveType="uint16"/>
    <type name="uInt32" primitiveType="uint32"/>
    <type name="uInt32NULL" presence="optional" nullValue="4294967295" primitiveType="uint32"/>
    <type name="uInt64" primitiveType="uint64"/>
    <type name="uInt64NULL" presence="optional" nullValue="18446744073709551615" primitiveType="uint64"/>
    <type name="Int32" primitiveType="int32"/>
    <type name="Int32NULL" presence="optional" nullValue="2147483647" primitiveType="int32"/>
    <type name="Int64" primitiveType="int64"/>
    <composite name="groupSize" description="Repeating group dimensions" semanticType="NumInGroup">
        <type name="blockLength" primitiveType="uint16"/>
        <type name="numInGroup" primitiveType="uint8"/>
    </composite>
    <type name="Int64NULL" presence="optional" nullValue="9223372036854775807" primitiveType="int64"/>
```

### 2.4.2. Decimal types

```
    <composite name="Decimal2NULL" description="Price type" semanticType="Price">
        <type name="mantissa" description="mantissa" presence="optional" nullValue="9223372036854775807" primitiveType="int64"/>
        <type name="exponent" description="exponent" presence="constant" primitiveType="int8">-2</type>
    </composite>
```

```
    <composite name="Decimal9NULL" description="Price type" semanticType="Price">
        <type name="mantissa" description="mantissa" presence="optional" nullValue="9223372036854775807" primitiveType="int64"/>
        <type name="exponent" description="exponent" presence="constant" primitiveType="int8">-9</type>
    </composite>
```

### 2.4.3. String types

```
    <type name="Char" primitiveType="char"/>
    <type name="String4" length="4" primitiveType="char" presence="optional" nullValue=" "/>
    <type name="String5" length="5" primitiveType="char" presence="optional" nullValue=" "/>
    <type name="String10" length="10" primitiveType="char"/>
    <type name="String12" length="12" primitiveType="char"/>
    <type name="String20" length="20" primitiveType="char"/>
    <type name="String256"  length="256" primitiveType="char"/>
    <type name="BoardID"    length="4" primitiveType="char" presence="optional" nullValue=" " description="Board ID value in trading system
SECBOARD(4)"/>
    <type name="SecurityID" length="12" primitiveType="char" presence="optional" nullValue=" " description="Instrument ID value in trading system
SECCODE(12)"/>
    <type name="ExchangeTradingSessionID"  primitiveType="uint32" presence="optional" nullValue="4294967295" description="Trading session ID"/>
    <type name="MarketID"  length="4" primitiveType="char" presence="constant">MOEX</type>
    <type name="SecurityIDSource" presence="constant" length="1" primitiveType="char">4</type>
    <composite  name="monthYearNull">
        <type name="year"  primitiveType="uint16" presence="optional" nullValue="65536" />
        <type name="month" primitiveType="uint8"  minValue="1" maxValue="12"  presence="optional"  nullValue="255"/>
        <type name="day"   primitiveType="uint8"  minValue="1" maxValue="31" presence="optional" nullValue="255" />
    </composite>
    <composite name="Utf8String" description="Variable-length utf-8 string">
        <type name="length" primitiveType="uint16" semanticType="Length"/>
        <type name="varData" length="0" primitiveType="uint8" semanticType="data" characterEncoding="UTF-8"/>
    </composite>
    <composite name="VarString" description="Variable-length string">
        <type name="length" primitiveType="uint16" semanticType="Length"/>
        <type name="varData" length="0" primitiveType="uint8" semanticType="data"/>
    </composite>
```

### 2.4.4. Enums

```
<enum name="MDUpdateAction" encodingType="uInt8">
  <validValue name="New" description="New"      >0</validValue>
  <validValue name="Change" description="Change">1</validValue>
  <validValue name="Delete" description="Delete">2</validValue>
</enum>

<enum name="MDEntryType" encodingType="Char">
  <validValue name="Bid" description="Bid"             >0</validValue>
  <validValue name="Offer" description="Offer"         >1</validValue>
  <validValue name="EmptyBook" description="Empty Book">J</validValue>
</enum>

<enum name="SecurityAltIDSource" encodingType="Char">
  <validValue name="ISIN" description="ISIN"                     >4</validValue>
  <validValue name="ExchangeSymbol" description="Exchange symbol">8</validValue>
</enum>
```

```
<enum name="SecurityTradingStatus" encodingType="uInt8NULL">
  <validValue name="TradingHalt" description="Trading halt"   >2</validValue>
  <validValue name="ReadyToTrade" description="Ready to trade">17</validValue>
  <validValue name="NotAvailableForTrading"
              description="Not available for trading"         >18</validValue>
  <validValue name="NotTradedOnThisMarket"
              description="Not traded on this market"         >19</validValue>
  <validValue name="PreOpen" description="Pre-open"           >21</validValue>
</enum>

<enum name="TradingSessionID" encodingType="uInt8NULL">
  <validValue name="Day" description="Day session"        >1</validValue>
  <validValue name="Morning" description="Morning session">0</validValue>
  <validValue name="Evening" description="Evening session">2</validValue>
</enum>

<enum name="MarketSegmentID" encodingType="Char">
  <validValue name="Derivatives" description="Derivatives">D</validValue>
</enum>

<enum name="TradSesStatus" encodingType="uInt8">
  <validValue name="Connected" description="Connection to trading system established">100</validValue>
  <validValue name="Disconnected" description="Connection to trading system lost"  >101</validValue>
  <validValue name="Restored" description="Connection to trading system restored. Trading system not restarted." >102</validValue>
  <validValue name="Reconnected"      description="Connection to trading system restored. Trading system restarted." >103</validValue>
  <validValue name="StartMain"        description="Start main trading session" >105</validValue>
  <validValue name="StopMain"         description="Stop main trading session" >106</validValue>
  <validValue name="StartEvening"  description="Start evening trading session" >107</validValue>
  <validValue name="StopEvening"   description="Stop evening trading session" >108</validValue>
  <validValue name="StartMorning"  description="Start morning trading session" >111</validValue>
  <validValue name="StopMorning"   description="Stop morning trading session" >112</validValue>
  <validValue name="Closed"           description="End of trading day" >109</validValue>
  <validValue name="Other"            description="Other event" >110</validValue>
</enum>
```

### 2.4.5. Bit masks

```
<set name="BPFlagsSet" encodingType="uInt8">
  <choice name="BidEmptyBook" description="Empty bid book"    >0</choice>
  <choice name="OfferEmptyBook" description="Empty offer book">1</choice>
</set>

<set name="MDFlagsSet" encodingType="uInt32">
  <choice name="Order" description="Orders: Day order"                          >0</choice>
  <choice name="Quote" description="Orders: External liquidity Quote"                      >1</choice>
  <choice name="Market in auction" description="Orders: Market in auction Order"          >2</choice>
  <choice name="LastFragment" description="Orders: last message in event"                        >3</choice>
  <choice name="Negotiated" description="Trades: Negotiated trade"                    >4</choice>
  <choice name="Confirmed" description="Trades: Confirmed LP trade"                    >5</choice>
  <choice name="DarkPool" description="Trades: DarkPool or hidden liquidity trade"                      >6</choice>
</set>

<set name="MsgFlagsSet" encodingType="uint16">
  <choice name="LastFragment" description="Message fragmentation flag"                >0</choice>
```

```
  <choice name="StartOfSnapshot" description="Flag of the first message in the snapshot
                                              for the instrument"                    >1</choice>
  <choice name="EndOfSnapshot" description="Flag of the last message in the snapshot
                                            for the instrument"                      >2</choice>
  <choice name="IncrementalPacket" description="Incremental packet flag"             >3</choice>
</set>
```

## 2.5. Message schemas

SIMBA ASTS message schemas are places in files at publicly available MOEX resource at address http://ftp.moex.com/pub/SBE/ASTS/xml/
If you detect discrepancies between this document message descriptions and xml schemas, please treat xml schemas as definitely correct.

```
<?xml version="1.0" encoding="UTF-8"?>
<?xml-stylesheet href="sbeasts.xsl" type="text/xsl"?>
<sbe:messageSchema package="sbe.asts" byteOrder="littleEndian" id="19780" version="0"
 semanticVersion="FIX5SP2" description="20201005"
 xmlns:sbe="http://fixprotocol.io/2016/sbe"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:schemaLocation="http://fixprotocol.io/2016/sbe sbe.xsd">
</sbe:messageSchema>
```

Schema attributes:

| Attribute | Definition | Value |
|---|---|---|
| id | Unique schema ID | 19780 |
| version | Schema version | 0 |
| package | Schema name | "sbe.asts" |
| byteOrder | Byte order | "littleEndian" |

# 3. Session level

## 3.1. Supported messages

- **Logon (msg id=1000)** – Initiates and confirms session with TCP Replay service of missed packets retrieval
- **Logout (msg id=1001)** - Initiates and confirms TCP Replay service session end.
- **Heartbeat (msg id=1)** - SIMBA ASTS Gateway sends this message if publishing channel has no meaningful messages to send for heartbeat interval. Established interval is 1 second.

Description of each message contains the following rows:

- **Tag** – field ID, following FIX rules;
- **Name** – field name;
- **Required** – indicates if nullValue is valid for this field:
- **Y** – required field, nullValue not allowed;
- **N** – optional field, nullValue is allowed;
- **C** – conditionally required, nullValue allowed only under special conditions.
- **Type** – Field data type;
- **Description** – detailed description of a field.

### 3.1.1. Logon (msg id=1000)

A message from client to SIMBA ASTS Gateway that initiates TCP Replay session to request missed packets. SIMBA ASTS Gateway reply to successful logon message. UserName field must not be empty.

It is recommended to fill UserName field with a value that helps in identification of client for support actions.

| Tag | Field name | Required | Data type | Description |
|---|---|---|---|---|
| <messageHeader> | | Y | | |
| 553 | UserName | Y | String12 | Username. |

### 3.1.2. Logout (msg id=1001)

A message from SIMBA ASTS Gateway that initiates end of TCP Replay session. Client reply message has an identic format.

| Tag | Field name | Required | Data type | Description |
|---|---|---|---|---|
| <messageHeader> | | Y | | |
| 58 | Text | Y | String256 | Free format string. May contain logout reason. |

### 3.1.3. Heartbeat (msg id=1)

SIMBA ASTS Gateway sends this message if publishing channel has no meaningful messages to send for heartbeat interval. Established interval is currently set to 1 second.

| Tag | Field name | Required | Data type | Description |
|---|---|---|---|---|
| <messageHeader> | | Y | | |

# 4. Application level

## 4.1. Supported message types

- **BestPrices (msg id=3)** – New Best Prices (best bid & best ask) message
- **EmptyBook (msg id=4)** – indication of empty orderbook.
- **OrderUpdate (msg id=5)** – add, change order visible quantity, or delete order.
- **OrderExecution (msg id=6)** – execution of passive order of internal liquidity. Contains trade data
- **Trade (msg id=16)** – negotiated trades or trades in hidden liquidity boards
- **OrderBookSnapshot (msg id=7)** – snapshot of a list of active orders.
- **SecurityDefinition (msg id=8)** – description of security.
- **SecurityStatus (msg id=9)** – Changes of security trading status and trading mode
- **TradingSessionStatus (msg id=11)** – trading session status and type changes
- **MarketDataRequest (msg id=1002)** – request of missed incremental messages via TCP Replay.

### 4.1.1. BestPrices (msg id=3)

BestPrices message is published if best bid or best ask price has been changed as a result of processing transaction. BestPrices message is published in a separate packet before packet(s) containing changes to the list of active orders that were changed or deleted because of transaction. May contain changes to multiple instruments.
Nullvalue for price field has a meaning of no orders in corresponding side.

| Tag | Field name | Required | Data type | Description |
|---|---|---|---|---|
| <messageHeader> | | Y | | |
| 268 | NoMDEntries | Y | groupSize | A number of elements in repeating group |
| => 645 | MktBidPx | N | Decimal9NULL | Best buy price. |
| => 646 | MktOfferPx | N | Decimal9NULL | Best sell price. |
| => 647 | MktBidSize | N | Int64NULL | Number of lots available at best price for sell |
| => 648 | MktOfferSize | N | Int64NULL | Number of lots available at best price for buy |
| => 20336 | Board | Y | BoardID | Trading board code in ASTS trading system |
| => 55 | Symbol | Y | SecurityID | Security ID, SECCODE in ASTS trading system. |

### 4.1.2. EmptyBook (msg id=4)

EmptyBook message indicates that you must clear all previously received data and start listening to incremental updates. EmptyBook message is always sent at SIMBA ASTS Gateway start for any reason.

| Tag | Field name | Required | Data type | Description |
|---|---|---|---|---|
| <messageHeader> | | Y | | |

### 4.1.3. OrderUpdate (msg id=5)

Adding order, changing order visible balance, deleting order. Pre-matched trades with external liquidity orders result in publishing this type of messages.

| Tag | Field name | Required | Data type | Description |
|---|---|---|---|---|
| <messageHeader> | | Y | | |
| 278 | MDEntryID | Y | Int64Null | Order ID in marketdata feed. Unique within trading day. |
| 270 | MDEntryPx | Y | Decimal9Null | Order price. |
| 271 | MDEntrySize | Y | Int64Null | Order visible quantity in lots. |
| 20017 | MDFlags | Y | MDFlagsSet | Bit mask with additional information:<br>0x0 – internal liquidity order (Day)<br>0x1 – external liquidity order (quote) from liquidity provider<br>0x2 – market order that was activated in auction collection phase<br>0x4 – LastFragment, indication that this message contains the last change of data per transaction |
| 83 | RptSeq | Y | uInt32 | Sequence number of this change per Board + Symbol combination. It is incremented by 1 for each OrderUpdate, OrderExecuted, Trade message per Board + Symbol combination. |
| 279 | MDUpdateAction | Y | MDUpdateAction | Type of update:<br>'0' - New<br>'1' – Change<br>'2' - Delete |
| 269 | MDEntryType | Y | MDEntryType | Order side:<br>'0' – Bid<br>'1' - Ask |
| 20336 | Board | Y | BoardID | Trading board ID |
| 55 | Symbol | Y | SecurityID | Security ID, SECCODE in ASTS trading system. |

### 4.1.4. OrderExecution (msg id=6)

Trades with visible passive internal liquidity orders. Contains fields for publication of new visible balance or order deletion as a result of trades with internal liquidity orders. Used in trading boards that allow publishing active orders.

| Tag | Field name | Required | Data type | Description |
|---|---|---|---|---|
| <messageHeader> | | Y | | |
| 278 | MDEntryID | Y | Int64Null | Order ID in marketdata feed. Unique within trading day. |
| 270 | MDEntryPx | Y | Decimal9Null | Order price. |
| 271 | MDEntrySize | Y | Int64Null | Order visible quantity in lots. |
| 31 | LastPx | Y | Decimal9Null | Trade price. |
| 32 | LastQty | Y | Int64Null | Trade quantity in lots. |
| 1003 | TradeID | Y | Int64Null | Trade ID. Unique value within market history |
| 20017 | MDFlags | Y | MDFlagsSet | Bit mask with additional information:<br>0x0 – internal liquidity order (Day)<br>0x1 – external liquidity order (quote) from liquidity provider<br>0x2 – market order that was activated in auction collection phase<br>0x4 – LastFragment, indication that this message contains the last change of data per transaction |
| 83 | RptSeq | Y | uInt32 | Sequence number of this change per Board + Symbol combination. It is incremented by 1 for each OrderUpdate, OrderExecuted, Trade message per Board + Symbol combination. |
| 279 | MDUpdateAction | Y | MDUpdateAction | Type of update:<br>'0' - New<br>'1' – Change<br>'2' - Delete |
| 269 | MDEntryType | Y | MDEntryType | Order side:<br>'0' – Bid<br>'1' - Ask |
| 20336 | Board | Y | BoardID | Trading board ID |
| 55 | Symbol | Y | SecurityID | Security ID, SECCODE in ASTS trading system. |

### 4.1.5 Trade (msg id=16)

A trade in negotiated trading boards, in boards with hidden liquidity, or confirmed trade with external liquidity order. A message has no order parameters.

| Tag | Field name | Required | Data type | Description |
|---|---|---|---|---|
| <messageHeader> | | Y | | |
| 31 | LastPx | Y | Decimal9Null | Trade price. |

| | | | | |
|---|---|---|---|---|
| 32 | LastQty | Y | Int64Null | Trade quantity in lots. |
| 1003 | TradeID | Y | Int64Null | Trade ID. Unique value within market history |
| 20017 | MDFlags | Y | MDFlagsSet | Bit mask with additional information:<br>0x4 – LastFragment, indication that this message contains the last change of data per transaction |
| 83 | RptSeq | Y | uInt32 | Sequence number of this change per Board + Symbol combination. It is incremented by 1 for each OrderUpdate, OrderExecuted, Trade message per Board + Symbol combination. |
| 279 | MDUpdateAction | Y | MDUpdateAction | Type of update:<br>'0' - New |
| 20336 | Board | Y | BoardID | Trading board ID |
| 55 | Symbol | Y | SecurityID | Security ID, SECCODE in ASTS trading system. |

### 4.1.6. OrderBookSnapshot (msg id=7)

A snapshot of a list of active orders. Empty list of active orders per security is published as NoMDEntries = 0.

| Tag | Field name | Required | Data type | Description |
|---|---|---|---|---|
| <messageHeader> | | Y | | |
| 369 | LastMsgSeqNumProcessed | Y | uInt32 | MsgSeqNum of last published incremental packet at the time of snapshot creation. |
| 83 | RptSeq | Y | uInt32 | RptSeq field value of last incremental message that was sent for this instrument before the snapshot creation. |
| 20336 | Board | Y | BoardID | Trading board ID |
| 55 | Symbol | Y | SecurityID | Security ID, SECCODE in ASTS trading system. |
| 268 | NoMDEntries | Y | groupSize | Number of groups of repeating fields in this message. This field value is equal 0 if no active orders exist for a given instrument. |
| =>278 | MDEntryID | Y | Int64NULL | Order ID in marketdata feed. Unique within trading day. |
| =>60 | TransactTime | Y | uInt64 | Order registration timestamp. Number of nanoseconds since UNIX epoch in UTC. |
| =>270 | MDEntryPx | Y | Decimal9Null | Order price. |
| =>271 | MDEntrySize | Y | Int64NULL | Visible order quantity in lots. |
| =>20017 | MDFlags | Y | MDFlagsSet | Bit mask with additional information:<br>0x0 – internal liquidity order (Day)<br>0x1 – external liquidity order (quote) from liquidity provider<br>0x2 – market order in auction that was activated in auction collection phase |

| | | | | |
|---|---|---|---|---|
| =>269 | MDEntryType | Y | MDEntryType | Order side:<br>'0' - Bid<br>'1' - Ask |

### 4.1.7. SecurityDefinition (msg id=8)

Description of financial instrument. Contains fields for security trading status at time of current publication cycle start.

| Tag | Field name | Required | Data type | Description |
|---|---|---|---|---|
| <messageHeader> | | Y | | |
| 911 | TotNumReports | Y | uInt32 | Total number of Security Definition messages in current publishing cycle |
| 20336 | Board | Y | BoardID | Trading board ID |
| 55 | Symbol | Y | SecurityID | Security ID, SECCODE in ASTS trading system. |
| 336 | TradingSessionID | Y | SecuritySessionStatus | 'NA' – No trading<br>'O' – Opening period<br>'S' - Opening auction period<br>'C' – Closing period<br>'N' – Normal trading period<br>'L' – Closing auction period<br>'I' – Discrete auction period<br>'D' – Dark pool auction period<br>'E' – Trading at the closing auction price period<br>'A' - Auction: Order entry phase<br>'a' - Auction: Trade conclusion phase<br>'b' - Auction: Bookbuilding phase, orders are locked<br>'p'- Auction: After auction trade phase |
| 625 | TradingSessionSubID | Y | SecuritySessionStatus | If this field value is 'B' , then you can find trading period value in tag 336<br>'NA' – No trading<br>'O' – Opening period<br>'S' - Opening auction period<br>'C' – Closing period<br>'B' – break in trading<br>'N' – Normal trading period<br>'L' – Closing auction period<br>'I' – Discrete auction period<br>'D' – Dark pool auction period<br>'E' – Trading at the closing auction price period<br>'A' - Auction: Order entry phase<br>'a' - Auction: Trade conclusion phase |

| | | | | |
|---|---|---|---|---|
| | | | | 'b' - Auction: Bookbuilding phase, orders are locked<br>'p'- Auction: After auction trade phase |
| 167 | SecurityType | Y | String(6) | 'CORP' (Corporate Bond);<br>'FOR' (Foreign Exchange Contract);<br>'CS' (Common Stock);<br>'PS' (Preferred Stock);<br>'EUSOV' (Euro Sovereigns);'<br>'BN' Bank Notes;<br>'MF' Mutual Fund<br>'MUNI' (Municipal bonds);<br>'RDR' – (Russian depositary receipt)<br>'ETF' – (Exchange traded fund);<br>'COFP' (Certificate Of Participation):<br>'XCN' (Extended Comm Note);<br>'STRUCT' (Structured Notes);<br>'WAR' (Warrant)<br>'GCD' (General collateral certificate)<br>'x' (Other)<br>This list may be extended in the future. Please contact help@moex.com for explanations of the meaning of new values. |
| 561 | RoundLot | Y | uInt32 | Lot size – number of units of security in one trading lot |
| 10512 | LotDivider | Y | uInt16 | Indicates trading in 1/LotDivider fractions of RoundLot. For example, RoundLot=1 and LotDivider =100 combination for USD/RUB has a meaning of trading in quantities of 1 cent. |
| 2349 | PricePrecision | Y | uInt8 | Maximum number of non-zero digits in price after decimal point. |
| 969 | MinPriceIncrement | Y | Decimal9Null | Minimum price increment. Price must be a whole number of MinPriceIncrement |
| 15 | Currency | N | String(4) | Settlement currency code |
| 5508 | FaceValue | N | Decimal9Null | Nominal value |
| 120 | SettlCurrency | N | String(4) | Facevalue currency code |
| 64 | SettlDate1 | N | monthYearNull | Settlement date in YYYYMMDD format |
| 20064 | SettlDate2 | N | monthYearNull | Settlement date 2 in YYYYMMDD. Applied to swap and REPO trading boards |
| 5459 | SettlType | N | String(12) | Settlement code |
| 5556 | BaseSwapPx | N | Decimal9Null | Base Price. May be indicated in negotiated swap trades |
| 1301 | MarketId | Y | MarketID | MIC code:<br>'MOEX' - Moscow Exchange. Constant. |
| 1300 | MarketSegmentId | N | Char | Market type:<br>'E' Equities market<br>'C' Currency Market |

| | | | | |
|---|---|---|---|---|
| 351 | EncodedSecurityDesc | Y | Utf8String | Instrument name, Russian language string encoded in utf-8 |
| 107 | SecurityDesc | Y | Utf8String | Instrument name in English language, encoded in utf-8 |
| 5383 | EncodedShortSecurity Desc | Y | Utf8String | Short Instrument name, Russian language string encoded in utf-8 |

### 4.1.8. SecurityStatus (msg id=9)

SecurityStatus message indicates a change in security trading mode or status.

| Tag | Field name | Required | Data type | Description |
|---|---|---|---|---|
| <messageHeader> | | Y | | |
| 336 | TradingSessionID | Y | SecuritySessionStatus | 'NA' – No trading<br>'O' – Opening period<br>'S' - Opening auction period<br>'C' – Closing period<br>'N' – Normal trading period<br>'L' – Closing auction period<br>'I' – Discrete auction period<br>'D' – Dark pool auction period<br>'E' – Trading at the closing auction price period<br>'A' - Auction: Order entry phase<br>'a' - Auction: Trade conclusion phase<br>'b' - Auction: Bookbuilding phase, orders are locked<br>'p'- Auction: After auction trade phase |
| 625 | TradingSessionSubID | Y | SecuritySessionStatus | If this field value is 'B' , then you can find trading period value in tag 336<br>'NA' – No trading<br>'O' – Opening period<br>'S' - Opening auction period<br>'C' – Closing period<br>'B' – break in trading<br>'N' – Normal trading period<br>'L' – Closing auction period<br>'I' – Discrete auction period<br>'D' – Dark pool auction period<br>'E' – Trading at the closing auction price period<br>'A' - Auction: Order entry phase<br>'a' - Auction: Trade conclusion phase |

| | | | | |
|---|---|---|---|---|
| | | | | 'b' - Auction: Bookbuilding phase, orders are locked<br>'p'- Auction: After auction trade phase |
| 20336 | Board | Y | BoardID | Trading board ID |
| 55 | Symbol | C | SecurityID | Security ID, SECCODE in ASTS trading system. |

### 4.1.10. TradingSessionStatus (msg id=11)

A message indicates trading session start, change of trading session or changes in status of trading session.

| Tag | Field name | Required | Data type | Description |
|---|---|---|---|---|
| <messageHeader> | | Y | | |
| 1301 | MarketID* | Y | MarketID | MIC:<br>'MOEX' - Moscow Exchange. Constant. |
| 1300 | MarketSegmentID* | Y | Char | Market type:<br>'E' Equities market<br>'C' Currency Market |
| 1300 | TradSesStatus* | Y | TradSesStatus | '100' (Connection to MOEX market was established)<br>'105' (Main Trading Session Start)<br>'106' (Main Trading Session End)<br>'107' (Addidional Trading Session Start)<br>'108' (Addidional Trading Session End)<br>'109' (End of Trading Day)<br>'110' (Other)<br>'111' (Morning Trading Session Start)<br>'112' (Morning Trading Session End)<br><br>End of Trading Day status (340=109) indicates that no new data will be published in current date. |

### 4.1.11. MarketDataRequest (msg id=1002)

A message requesting missed incremental messages via TCP Replay.

| Tag | Field name | Required | Data type | Description |
|---|---|---|---|---|
| <messageHeader> | | Y | | |
| 1182 | ApplBegSeqNum | Y | uInt32 | Sequence number of the first requested message |
| 1183 | ApplEndSeqNum | Y | uInt32 | Sequence number of the last requested message |

## 4.2. Examples

### 4.2.1. New order arrived, passive order filled

New order fills passive order and becomes filled too. Best prices change.

before transaction

| bid size | price | ask size |
|---|---|---|
| | | |
| | 77665 | 100 |
| | 77664 | 26 |
| 123 | 77650 | |
| | | |
| | | |

after transaction

| bid size | price | ask size |
|---|---|---|
| | | |
| | | |
| | 77665 | 100 |
| 123 | 77650 | |
| | | |
| | | |

*Figure 3. Orderbook state*

First incremental packet contains BestPrices message with new values of MktBidPx (did not change) and MktOfferPx (changed):

```
{ Packet header:
 MsgSeqNum=105805 MsgSize=N MsgFlags={ LastFragment:0 StartOfSnapshot:0 EndOfSnapshot:0
 IncrementalPacket:1 } SendingTime=20201014070029621
}
{ Incremental packet header:
 TransactTime[60]=70029621508252 ExchangeTradingSessionID[5842]=6144
}
{ SBE Header:
 BlockLength=M TemplateID=3 SchemaID=1 Version=1
}
{ SBE Message:
  Sequence: NoMDEntries[268] = 1 {
   [0]:  Board[20336]=TQBR Symbol[55]=Sample MktBidPx[645]=77650 MktOfferPx[646]=77665 BPFlags[22000]=0x0
}
```

Second packet contains OrderExecuted message with passive order deletion and trade data. Filled aggressive order is not published in the feed:

```
{ Packet header:
 MsgSeqNum=105806 MsgSize=N MsgFlags={ LastFragment:1 StartOfSnapshot:0 EndOfSnapshot:0
 IncrementalPacket:1 } SendingTime=20201014070029621
}
{ Incremental packet header:
 TransactTime[60]=70029621508252 ExchangeTradingSessionID[5842]=6144
}
{ SBE Header: BlockLength=N TemplateID=6 SchemaID=1 Version=1 }
{ SBE Message:
 MDUpdateAction[279]=2 MDEntryType[269]=1 MDEntryID[278]=18929456 Board[20336]=TQBR Symbol[55]= Sample
 RptSeq[83]=60145 MDEntryPx[270]=77664 LastPx[31]=77664 LastQty[32]=26
 TradeID[1003]=18929456066 MDFlags[20017]=0x00000000000
}
```

### 4.2.2. Recovering of missed packets via a separate TCP session (TCP Replay)

If market data packets are missed, they can be recovered over the TCP historical replay component using the sequence number range. TCP Replay is a low performance recovery option and should be only used if other options are unavailable or for a small-scale data recovery. A number of messages which can be requested by client during TCP connection are limited.

TCP replay recovery include the following:

1. Establish TCP connection with SIMBA ASTS gateway.
2. Send Logon (msg id=1000) message to the server.
3. Server replies with Logon (msg id=1000) message
4. Send MarketDataRequest (msg id=1002) message
5. Server replies with requested packets
6. Server initiates termination of session by sending Logout (msg id=1001) message
7. Confirm termination by sending Logout (msg id=1001) message
8. Server closes TCP connection

TCP connection is closed after receiving confirming logout or after maximum waiting time is reached.

**Note:** closing connection without sending confirming logout is considered as abnormal situation.

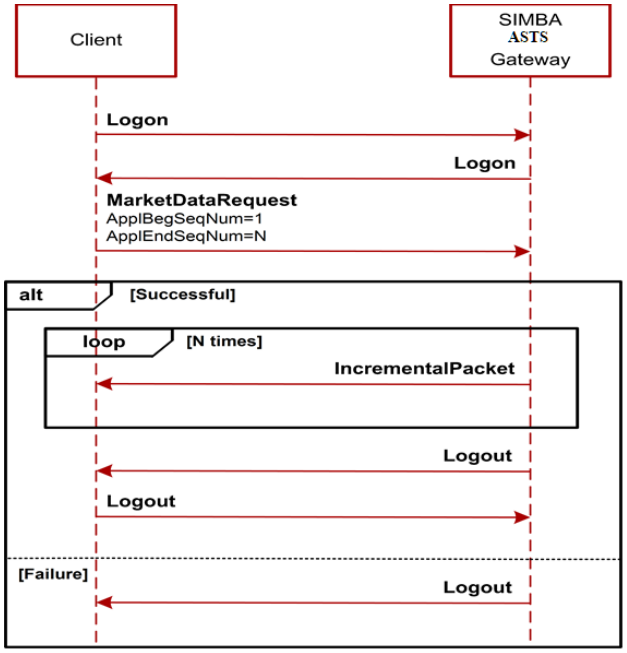


*Figure 4. Diagram of message recovery via TCP replay*

TCP Replay service has the following limitations:

| Parameter | Value | Description |
|---|---|---|
| Maximum number of simultaneously active TCP connections per IP address | 2 | You can establish not more than indicated number of active TCP connection. Excessive connection attempt will be rejected. |
| Maximum number of TCP connections per IP address per day | 1000 | You can establish not more than indicated number of TCP connection per IP address per day. Excessive connection attempt will be rejected. |
| Maximum number of packets in request | 1000 | Marketdata request with higher number of packets will be rejected. |
| Maximum waiting time for expected message, seconds | 1 | Server closes connection if, after indicated number of seconds:<br>Logon message is not received after the TCP connection was established<br>MarketDataRequest message was not received after sending Logon reply.<br>Logout message was not received after sending Logout. |