Arena Gateway 3.1.3

Bucharest Stock Exchange

Version 3.1.3, 2019-06-24

Table of Contents

Revisions

Document Version	Date	Description
1.0.0	June, 28th 2007	Create document
1.0.1	February, 20th 2008	Change password command ErrorDto error codes Cache Order activated message (code 719) ExchangeExplorerDto clarifications `hdv` and `uty` fields from OrdDto
1.0.2	February, 23th 2008	sendCacheAtLogin runtime property
1.0.3	March, 15th 2008	GetOrdersDailyLogCmd changes (`sym`, `mkt`, `dat`, `acc` were removed) GetDailyTradesCmd replaces GetTradesCmd AddOrderSellCmd changes (`ssl` field was added) HalfTrdDto changes (`ssl` and `otk` fields were added) HalfTrdDto `uty`, `din` and `trt` fields clarifications ShortHalfTrdDto was replaced by HalfTrdDto OrdDto changes (`ssl` and `prt` fields were added) SmsInfo (removed `crf` and `foi` fields) Message type 600 was corrected to 603 New reports: GetDailyIndicesCmd, GetDailyPublicTradesCmd and GetSymbolsByNameCmd. New unsolicited messages types: 354, 555, 558 The gateway will receive all trade notifications and order operations confirmations/notifications regardless they were performed by other users from the same participant or by system administrators.

Document Version	Date	Description
1.1.3	May, 8th 2008	PublicTickerMsg label 818 was corrected to 816 Added missing SymDto and MailDto descriptions ShareContentDto, BondContentDto, TBillContentDto, FutureContentDto changes (added `psl` field) New reports: GetStepsCmd, GetPublicParametersCmd New unsolicited messages types: 790 Clarifications and changes SmsInfo (`dfd` field was removed) CommonTickerMsg changes (label 902 was removed)
1.1.4	July, 22th 2008	Runtime upgrade
1.1.5	November 24th 2008	Runtime upgrade
1.1.6	February 28th 2009	`OrdDto.mkp` clarifications CommonTickerMsg label 991 (Default Settlement Flag) clarifications `ExchangeExplorerDto.type` clarifications
1.1.7	August 28th 2009	Runtime upgrade
1.1.8	November 30th 2009	Runtime upgrade
1.1.9	April 27th 2010 This is the last 1.x version	CommonTickerMsg label 991 correction New values for `OrdDto.uty` field (Reject New, Reject Cancel, Reject Replace), please see also Error Processing section for more details An user does not have to load anymore the symbol and subscribe to the symbol-market in order to send order management commands Report pagination clarification, please see Request/Response Processing sesction for details
2.0.0	May 5th 2010	AddOrderBuyCmd and AddOrderSellCmd changes (field `iac` was added) HalfTrdDto changes (fields `clc` and `lqi` were added) OrdDto changes (fields `iac`, `oqy`, `cqy`, `lqy`, `apx`, `lpx`, `ost`, `cli`, `ocl`, `txt` were added)

Document Version	Date	Description
2.0.1	December 14th 2010	`ssl` field from AddOrderSellCmd, OrdDto and HalfTrdDto has a new possible value (2 = Short Sell Exempt) HalfTrdDto changes (field `iac` was added) `tms` field from ChgOrderCmd is no longer mandatory. If missing or null, the trading engine will process the change order command regardless the order's last update time. There is a new parameter named `uptickrule`.
2.0.2	March 14th 2011	Account types (`act`) correction at HalfTrdDto New commands: UpdateMMOrdersCmd and CancelMMOrdersCmd and responses: QuoteResultDto OrdDo changes (field `rol` was added, new `uty` values: Reject Update MMO and Reject Cancel MMO) New parameters: `mmsprdtype`, `mmsprdmin`, `mmsprdmax` Order operations reject codes details (Reject Codes section)
2.0.3	November 9th 2011	StsInfo clarification
2.0.4	February 1st, 2012	Field `sde` was added to UpdateMMOrdersCmd and CancelMMOrdersCmd
2.1.0	May 31st, 2012	UpdateMMOrdersCmd changes (add `bqy`/`sqy` fields and new behavior) ChgOrderCmd changes (add `oqy` field and new behavior)
2.2.0	August, 6th, 2012	Added a new type of ExchangeExplorerDto. Added Underlying Symbols report and the associated data transfer object (UlyDto).
2.2.1	September, 6th, 2012	Added deals confirmations, see Events Processing; reject codes updates
2.2.2	December 4th, 2012	Runtime upgrade
2.3.0	September, 19th, 2013	CommonTickerMsg changes (labels 901 and 992 were added)
2.4.0	October, 29th, 2013	Support for a new market status: trading-at-last

Document Version	Date	Description
2.5.1	July, 3rd, 2014	New command: AddCrossOrdersCmd Changes to commands: AddOrderBuyCmd , AddOrderSellCmd, ChgOrderCmd and UpdateMMOrdersCmd (added new terms and a new trigger type) HalfTrdDto changes (new possible value for `lqi` field) OrdDto changes (new possible values for `rol`, `trm` and `tpa` fields) Reject codes updates
3.0.0	July, 7th, 2014	Added new field `osq` to ChgOrderCmd, OrdDto and HalfTrdDto. Removed `tms` from ChgOrderCmd CommonTickerMsg changes (new possible value for label 901)
3.0.1	September, 10th, 2015	Clarifications related to `osq` field. Updated list of entity parameters. New reject code
3.0.2	March, 30th, 2016	Added section for bonds coupons syntax
3.0.3	January, 3rd, 2017	Volatility Interruption (new SmkDto.sts values, new market and symbol-market params)

Document Version	Date	Description
3.1.0	September, 29th, 2017	MiFID II related changes: • New fields (iwf, ial, ewf, eal, dea) in the orders commands (AddCrossOrdersCmd, AddOrderCmd, CancelMMOrdersCmd, CancelOrderCmd, ChgOrderCmd, ReleaseOrderBuyCmd, ReleaseOrderSellCmd, SuspendOrderCmd, UpdateMMOrdersCmd) • New fields (iwf, ial, ewf, eal, dea, uws) in the order structure (OrdDto) • New fields (neg, alg) in the trade structure (HalfTrdDto) Deals operations support: • AddDealBuyCmd • AddDealSellCmd • CancelDealCmd • ConfirmDealBuyCmd • ConfirmDealSellCmd • RefuseDealCmd New reject codes (89) and new parameters (minxval, minhval, deal.tuty, liq)
3.1.1	October, 10th, 2017	AddCrossOrdersCmd must have two pairs of short codes, new reject codes (90-97)
3.1.2	February, 20th, 2018	Added ExchangeExplorerDto.type 11 (exchange level param)
3.1.3	June, 24th, 2019	Changes in grouping criteria of `GetDailyPublicTradesCmd` response (see TradeTickerDto (trades info) for details), `TradeTickerDto.tim` has millisecond precision

Introduction

Arena Gateway is an application that acts as a message intermediary between the participant systems and the stock exchange central system. It provides request/response services, event based services as well as connectivity. Using a TCP/IP XML based messaging system it will receive commands from the participant systems (gateway client), send them to the central system and provide responses and market data events back to the client.

Arena Gateway requires the Java Runtime Environment version 7 or 8 in order to run.

Components

The gateway
The gateway will be deployed at the member site and will be used as a standalone application.

The gateway client
A gateway client is an application that should be built/bought by the member and that connects to the gateway, sends commands / receive responses from the central system through the gateway. The communication protocol between the gateway client and the gateway is TCP/IP XML schema based. We also provide a simple java based gateway client as a reference implementation.

Communication protocol format

At wire level a message has the following format:

```
|------------------|------------------|------------|-----------------|

   start sequence      message body      checksum      end sequence
```

1. The start sequence is a 9 byte sequence (`0xEF 0x81 0x86 0xE2 0x86 0xA6 0xEF 0x81 0x86`)
2. Message Body - the message payload; it has variable length
3. Checksum - the MD5 hash computed over the message body - 16 bytes
4. The end sequence is a 9 byte sequence (`0xEF 0x81 0x85 0xE2 0x86 0xA4 0xEF 0x81 0x85`)

UTF-8 charset is used to convert between bytes and character sequence representation of the message body. The message body should not contain the start or end sequences in order to prevent a misinterpretation of the real message.

There are two types of messages that flow between the gateway and the client:

- outgoing messages: messages that are sent from the gateway client to the central system through the gateway
- incoming messages: messages received from the central system by the gateway client through the gateway

At application level the message body is interpreted as an XML formatted text. The XML message structure is fully described using an XML Schema file named arena-gateway-messages.xsd. Every outgoing message will be parsed and validated against the provided XML Schema. In the event of an invalidated message the gateway will send an error message to the client formatted against this specification.

Some of the fields of the incoming messages are blank if they refer to confidential information. For example the field `broker code` from an OrdDto structure is filled only for participant own orders or if the exchange rules states that the owner of a quotation is to be public. Otherwise the field `broker code` is an empty string.

Outgoing Messages Description

Any outgoing message has a client sequence and an inner content that represents the actual command with its parameters. An outgoing engine message has the following general structure:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>

<m:outgoingEngineMessage
xmlns:c="http://www.bvb.ro/xml/ns/arena/gw/constraints"
xmlns:m="http://www.bvb.ro/xml/ns/arena/gw/msg"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">

<content xsi:type=...>

...

</content>

<csq>10001</csq>

</m:outgoingEngineMessage>
```

OutgoingEngineMessage fields description follows:

Field name	Description
content	This is the body of the actual command that is sent to the central system. The content is detailed for every type of command in the next chapters.
csq	The client sequence can be used to identify an incoming message as being the response for this outgoing message. The central system will not modify the content of this field. Client sequence is managed by the gateway client.

Incoming Messages Description

Every incoming message has a header and an inner structure that embeds a command confirmation, a report, a market data event or other information. Those embedded structures are called Data Transfer Objects. An incoming message has the following general structure:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>

<m:incomingEngineMessage
xmlns:c="http://www.bvb.ro/xml/ns/arena/gw/constraints"
xmlns:m="http://www.bvb.ro/xml/ns/arena/gw/msg"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">

<content xsi:type=...>

...

</content>

<csq>10003</csq>

<err>false</err>

<id>213</id>

<kmsg>to ADMIN: Hello</kmsg>

<ktime>20070827174849441</ktime>

</m:incomingEngineMessage>
```

The common fields of an incoming engine message are described below:

Field name	Description
id	The type id of the message; based on this id the client knows what to expect in the content
ktime	Time (yyyyMMddHHmmssSSS); the time at witch the message was generated at the central system.
kmsg	comment, message or error description
err	error flag
csq	Client sequence, used to match an incoming message to a command. It is 0 in case of un solicited messages (like market event incoming message).
content	Contains the actual message content. It contains one or more data structures packed together. Also, it can be null. Content is detailed for every type of incoming message in the next chapters.

An incoming message can be a response to a previous command in which case it has a client sequence equal to the client sequence of the command or can represent an event that happened on the market place.

Data processing and message flow

The gateway uses two pipes of communication to talk to the central system. The first pipe line is used to send command messages and the other is used to retrieve reports. Those lines are synchronous by design meaning that after sending a message through a pipe (command or report) one should wait for the response before sending the next message. Any attempt to do so will discard every message until the pending response will arrive.

Session protocol

A gateway client should take the following steps in order to carry out a regular trading session:

- Open a TCP/IP connection to the gateway
- Send a log-in message to the gateway
- Wait for log-in response or error
- Send regular business messages / report requests to the gateway
- Wait for response messages and process response
- Wait for unsolicited events messages and process it
- Send a log-out message to the gateway
- Disconnect from the gateway

Monitor the connection to the gateway

Once a connection is established with the gateway, the client can test the connection's health by sending heartbeat messages (HeartBeatCmd) to the gateway. When the gateway receives such a message it responds with an empty incoming message.

Sending commands and process confirmations

The gateway client can issue commands as outgoing message any time after login.

Any command that arrives at the central system will have a response that maps to an incoming message containing a confirmation that states the command was executed as expected or an incoming message with the error flag marked to true that states that the command could not be executed. In case of an error message the content of the incoming message is null but the field `kmsg` should contain further explanations about the encountered error.

Sending report requests and process responses

The gateway client can issue report requests as outgoing messages. Any report request that reaches the central system will have a response as an incoming message containing a page of records or an incoming message with the error flag marked to true. In case of an error message the content of the incoming message is null but the field `kmsg` should contain further explanations about the encountered error. The number of records in a response page is limited. In order to receive all the pages of a report the gateway client has to issue further report requests, until the last page of records is received.

Load and Subscribe mechanism

In order to receive information about a symbol the gateway client has to *load* a symbol using the LoadSymbolCmd command. Loading is persistent until unload, so that this command has to be issued only once for one symbol.

In order to receive market data for a symbol-market entity the gateway client has to *subscribe* to these symbol-markets using the AddSubscribeCmd command. Subscription is persistent until unsubscribe.

Processing Exchange Entities

After a successful login the client will receive a MarketPictureDto structure that contains a list of exchange entities. Entities can be: the exchange, market, symbol-type, symbol, symbol-market, index, etc. Usually every entity contains two types of information: properties and trading summary. Every time a property of an entity or the summary of the entity changes in the central system the client will receive the changes so it can build an up to date image of the exchange at the participant site.

The exchange entity is the top most entity. For example, the exchange has a status property and the summary contains trading statistics at exchange level.

The Symbol-type entity represents a group of symbols grouped by different criteria (for example bonds are put in different groups than shares)

The Symbol entity represents a financial instrument that is traded on the exchange.

The Market entity represents a trading place where all the symbols are traded using the same general rules.

A Symbol-market entity is a compound entity formed by symbol and the market it is traded on. For a symbol can be defined more than one symbol-market because a symbol can be traded on more than one market, using different trading mechanisms.

The properties of an entity can be delivered by the gateway as ExchangeExplorerDto structures embedded in different incoming messages types. For example one time such information is available is inside a UserEnvDto that arrives at login or after login whenever those properties change during a session inside TickersPack messages.

The summary of an entity (except symbol-market) will be present at login time as a summary structure embedded in a UserEnvDto or during the session as a list of PublicTickerMsg packed inside a TickersPack message.

Structural changes of the exchange entities (a symbol is removed from a certain market) will arrive during the session as ExchangeExplorerDto structures packed inside a TickersPack message.

Processing Level1 Market Data

Level1 market data refers to trading information for symbol-market entities for the current day (best bid/ask, last trade, total volume, total value, variation, open, close, high, low). Level1 market data is delivered by the gateway in two different ways: first, at login time, using symbol-market summary and second, during the session as CommonTickersPack structures embedded inside TickersPack messages

Note that level1 market data will be available only for subscribed symbol-market entities.

When a single order produces trades at different prices the client will receive a number of TickersPack messages that is equal to the number of distinct prices of the trades. All but the last of those messages will have the `partial` flag set to true and the CommonTickerMsg structures will include values only for some of the fields of the symbol-market summary (for example best bid/ask prices and volumes will not be included).

Processing Level2 Market Data

In order to have a full market depth image for a symbol-market, the gateway client has to obtain first a snapshot of the order book and then to update the order book with the changes that will be delivered as ActionTickerPack structures embedded inside incoming messages of type 800.

The snapshot is obtained by issuing the GetMarketByOrderCmd command once per session. The response to this command will be a MboDto structure that contains the order books as a pair of lists of OrdDto structures.

Even if for one book the MboDto contains only one list with orders from both sides (buy and sell) the elements of this list are sorted based on trading priority (descending) for each side. The client should build two lists of orders for every book: one for the buy side and another for the sell side based on the side of each OrdDto object without changing the original sorting order. This is necessary because the changes of the order book arriving as ActionTickersPack objects have to be applied on the original snapshot.

The order book can contain market orders (in pre-open session for example). Those orders are identified by the `mkp` field of the OrdDto structure. For those orders the `prc` field is only informative and should not be displayed (for example the market orders can display M as price and not a numeric value).

Own orders can be identified in the order book using the `own` field which will be set to true. For the rest of the orders only certain fields have relevant information (for example, for an order owned by other participant the `acc` field will be always set to 0).

Gateway Cache

The gateway will keep its own cache with the full order book in order to respond faster to GetMarketByOrder requests. After the UserEnvDto arrives the gateway client will start to receive a series of unsolicited messages that will provide the full order book content. The series will begin with a 762 type (start init cache with a SmkDto payload) message followed by 175 type messages (with MboDto payload for each symbol-market the user is subscribed to) and will end with a 765 type (end init cache) message. The payload of the 762 type message is a SmkDto with only two relevant fields: `sym` and `mkt`. Those fields specify the symbol-market range for which 175 type messages will arrive next. The symbol-market range is filtered again with the list of subscribed symbol-markets.

The range can by specified in 4 combinations:

- `sym=*, mkt=*` (all symbols from all markets),
- `sym=<A_SYMBOL>, mkt=*` (symbol `A_SYMBOL` from all the markets it is traded on),
- `sym=*, mkt=<A_MARKET>` (all symbols from market `A_MARKET`),
- `sym=<A_SYMBOL>, mkt=<A_MARKET>` (symbol `A_SYMBOL` on the `A_MARKET` market).

After login the range will always be `sym=*, mkt=*`. One can expect a different range only if an invalidate cache event occurs for a smaller cache region during the session followed by a cache reload.

For example suppose the user is subscribed to (s1,m1) and (s2,m1) and not subscribed to (s3,m1). After login the client will receive a 762 type message with the SmkDto fields set to `sym=*, mkt=*`. Next it will receive only two 175 type messages containing the order book for (s1, m1) and (s2, m1). The init cache flow will end with a 762 type message.

After end init cache arrives any GetMarketByOrder command will have the response build from the gateway cache if the user is subscribed to the requested symbol-market and the order book is in the cache. If the user is not subscribed to the symbol-market the GetMarketByOrder command will receive a gateway generated error response. If the user is subscribed and the subscription was performed in a prior session but the order book cache was not loaded yet (cache initialization is not finished yet because the end init cache did not arrive) on the gateway the GetMarketByOrder command will receive a gateway generated error asking the client to wait for the order book as it is on it's way. If the user is subscribed but the subscription was done in the current session the GetMarketByOrder command will be passed to the central system.

In case of gateway cache synchronization errors the cache could be invalidated from the central server. In this case the client will receive a 761 type message with a SmkDto content that will specify the cache range that was invalidated. The client should invalidate its own cache. Usually the invalidated cache will be reloaded later from the central system and the client will receive a cache initialization sequence similar to the one described above for the after login event.

One can control if 175 type messages will be transmitted to the client during the cache initialization phase by using the sendCacheAtLogin runtime property. This property can be set in the gateway.bat/gateway.sh launch files. If sendcacheAtLogin is set to true then the 175 type messages will be transmitted to the client and if is set to false during the cache initialization phase the client

will receive only the start init cache and end init cache messages.

Request/Response Processing

Outgoing messages can be split in three categories:

Command Type	Command Name	Description
Framework	LoginCmd	Login command
	LogoutCmd	Logout command
	HeartBeatCmd	Test the connection between the client and the gateway
Business	LoadSymbolCmd	Load a symbol in the client's virtual environment
	AddOrderBuyCmd	Buy order command
	AddOrderSellCmd	Sell order command
	AddCrossOrdersCmd	Add cross orders for immediate execution
	UpdateMMOrdersCmd	Add or change a pair of orders (a quote)
	CancelMMOrdersCmd	Delete a pair of orders (a quote)
	AddSubscribeCmd	Subscribe to a symbol-market
	CancelOrderCmd	Cancel an order
	ChgOrderCmd	Change an order
	AddDealBuyCmd	Sends a buy deal to another user
	AddDealSellCmd	Sends a sell deal to another user
	CancelDealCmd	Cancels a previously sent deal
	ConfirmDealBuyCmd	Confirms a received sell deal
	ConfirmDealSellCmd	Confirms a received buy deal
	RefuseDealCmd	Refuses a received deal
	DelSubscribeCmd	Unsubscribe from a symbol-market
	GetMarketByOrderCmd	Get order book snapshot
	ReleaseOrderBuyCmd	Resume a buy order
	ReleaseOrderSellCmd	Resume a sell order
	SuspendOrderCmd	Suspend an order
	UnloadSymbolCmd	Unload a symbol from the client's virtual environment
	MailCmd	Send message to another user

Command Type	Command Name	Description
Report Request	Find2AccountCmd	Find details about two accounts in a single request
	FindAccountCmd	Find details about an account
	FindFreshOrderReportCmd	Find an order (by order number)
	GetGenericOrderAuditCmd	Get all records related to an order (order audit)
	GetOrdersDailyLogCmd	Get all records related to all orders (for current member)
	GetOutstandingOrdersCmd	Get outstanding orders (for current member)
	GetDailyTradesCmd	Get trades from current date (for current member)
	GetUsersByNameCmd	Get users (connected or by member)
	GetSymbolsByNameCmd	Get list of symbols (based on load status)
	GetDailyPublicTradesCmd	Get all public trades from current date
	GetDailyIndicesCmd	Get the value of indices from current date
	GetStepsCmd	Get all the lists of price steps
	GetPublicParametersCmd	Get the parameters for all markets and symbol-markets
	GetUnderlyingsCmd	Get the current underlying symbols

Framework and business command's responses are listed in the table below:

Command	Response message id	Response message content	Notes
LoginCmd	101	UserEnvDto	It contains the full market picture at the time a client starts the session. The `csq` field of the login confirmation message is always 0 (zero). Every message before it should be discarded as irrelevant (except heart beat messages).
LogoutCmd	102	Null	
HeartBeatCmd	601	Null	

Command	Response message id	Response message content	Notes
LoadSymbolCmd	579	Null	
UnloadSymbolCmd	580	Null	
AddSubscribeCmd	275	SmsDto	
DelSubscribeCmd	276	SmsDto	
GetMarketByOrderCmd	175	MboDto	
MailCmd	213	Null	
ChgPasswordCmd	270	Null	
AddOrderBuyCmd	378	OrdDto	
AddOrderSellCmd	373	OrdDto	
CancelOrderCmd	305	OrdDto	
ChgOrderCmd	309	OrdDto	
SuspendOrderCmd	147	OrdDto	
ReleaseOrderBuyCmd	381	OrdDto	
ReleaseOrderSellCmd	376	OrdDto	
AddCrossOrdersCmd	304	QuoteResultDto	
UpdateMMOrdersCmd	679	QuoteResultDto	
CancelMMOrdersCmd	680	QuoteResultDto	
AddDealBuyCmd	377	OrdDto	
AddDealSellCmd	372	OrdDto	
CancelDealCmd	306	OrdDto	
ConfirmDealBuyCmd	379	OrdDto	
ConfirmDealSellCmd	374	OrdDto	
RefuseDealCmd	319	OrdDto	

The content of a report response message is a ResultPageDto. ResultPageDto has a number of fields that can be used to figure out the *position* of the current page in the request's result set. The records of a response are the elements of a list embedded in the ResultPageDto structure. Below is the list of the report request commands and their corresponding responses.

Command	Rsponse message id	Record type	Notes
FindAccountCmd	196	AccDetailsDto	One record
Find2AccountCmd	576	AccDetailsDto	Two records
FindFreshOrderReportCmd	433	OrdDto	One record
GetGenericOrderAuditCmd	165	OrdDto	
GetOrdersDailyLogCmd	350	OrdDto	
GetOutstandingOrdersCmd	173	OrdDto	

GetDailyTradesCmd	783	HalfTrdDto	
GetUsersByNameCmd	222	UsrDto	
GetSymbolsByNameCmd	360	SylDto	
GetDailyPublicTradesCmd	784	TradeTickerDto	
GetDailyIndicesCmd	785	IdxValueDto	
GetStepsCmd	329	PriceStepDto	
GetPublicParametersCmd	789	ParamsShortDto	
GetUnderlyingsCmd	690	UlyDto	

In order to fetch the entire result set of a report that has more than one page, one has to issue the report request until the end of the result set. In order do that the following pseudo-code can be used:

```
request = new ReportCmd(param1, param2, ...)

//prepare the request for the first page
request.dir = 0
request.str = 0

bottom = false
error = false

while (!bottom and !error) {
  response = client.sendRequest(request)
  error = response.error
  if(!error) {
    //extract the current page
    page = response.content

    //process the records
    process(page.lines)

    //prepare the request for the next page
    request.dir = 1
    request.str = page.start
    bottom = page.bottom
  }
}
```

Events Processing

In case an event happens on the central system a corresponding incoming message should arrive to the client containing the information related to that event.

Operational events are generated when an exchange entity is added/deleted or one of its properties is changed. For example when the status of a symbol-market is changed a corresponding event is

generated. The message id is always 800 in this case and the content of the message will be a TickersPack structure. Only the `xll` field of the TickersPack will be populated and all the PublicTickersPack elements of this list will have the `type` field set to 0 (zero).

Trading activity performed on the central system will generate public messages with id 800 as well. The message content will be a TickersPack structure populated as follows:

- `cmn` field will contain symbol-market level1 data
- `act` field will contain symbol-market level2 data
- `xll` field will contain a list of changes of the summaries of the exchange entities as PublicTickerMsg structures

The table below lists all the possible incoming messages a client should expect to come unsolicited from the gateway.

Message id	**Event**	**Message content type / comments**
800	Operational events and/or trading activity	TickersPack
802	Trade confirmation	HalfTrdDto The gateway will receive all trade notifications regardless they were performed by other users from the same participant
378, 373, 305, 309, 147, 381, 376, 719, 354, 555, 558	Operations performed on an order by another user (change, delete, etc). Note that order fills will not be transmitted this way.	OrdDto 719 arrives in case of order activation for contingent orders 354, 555 or 558 arrive in case batch operations are performed upon orders by system administrators The gateway will receive all order operations confirmations regardless they were performed by other users from the same participant
377, 372, 379, 374, 323, 375, 380, 306, 695	Operations performed with deals by another user (add deal, confirm deal, refuse deal, etc). Note that deals operations can't be performed through the gateway	OrdDto

Message id	Event	Message content type / comments
679, 680	Updates or cancels of market maker quotes	QuoteResultDto A QuoteResultDto is just a wrapper of a list of OrdDto.
304	Cross orders confirmations	QuoteResultDto
801	A text message sent by another user or other text announcements.	MailDto
102	The gateway was disconnected from the central system.	Null
603	The central system responds to a heartbeat message sent by the gateway. Note that this heartbeat is not related to the HeartBeatCmd.	Null. `ktime` will be a `yyyyMMddHHmmssSSS` timestamp representing the time of the central system.
762	Start init cache. This message arrives after a successful login or after the order book cache was reloaded on the communication server.	SmkDto In this case only `sym` and `mkt` fields are relevant and they can come in 4 different combinations: • sym=*, mkt=* • sym=defined, mkt=* • sym=*, mkt=defined • sym=defined, mkt=defined After this message the client should expect a series of 175 type messages (MboDto payload will have the `symbol` and `market` field from the range defined in the above 4 categories) that could be used to initialize the cache of the client. For example if sym=* and mkt=* the client should expect the 175 type messages for every symbol-market it is subscribed to.

Message id	Event	Message content type / comments
765	End init cache. This message marks the end of the cache initialization process.	Null
761	The order book cache was invalidated from the central system and no longer consistent.	SmkDto. See the comments for 762 above. When this message arrives the client should invalidate its own cache as it is no longer consistent. For example if sym=* and mkt=REGS the client should invalidate the order books for every symbol trades in REGS market.
175	After a successful login or after the order book cache was reloaded on the communication server.	MboDto
790	When a market or symbol-market parameter is added, changed or deleted	ParamsShortDto

Error Processing

The gateway client can receive error messages at different stages of a communication session generated by different components (central system or gateway).

Central system generated error responses are incoming messages filled as follows:

- csq=The csq of the message request that produced the error
- err=true
- id=The regular id to be expected for the message request that produced the error
- content=Null / OrdDto (see order management notes below)
- kmsg=Error description
- ktime=`yyyyMMddHHmmssSSS` formatted central system's timestamp

In case an order or deal management command (AddOrderBuyCmd, AddOrderSellCmd, CancelOrderCmd, ChgOrderCmd, SuspendOrderCmd, ReleaseOrderBuyCmd, ReleaseOrderSellCmd, AddDealBuyCmd, AddDealSellCmd, CancelDealCmd, ConfirmDealBuyCmd, ConfirmDealSellCmd, RefuseDealCmd) is not accepted by the central system an order reject record will be generated, and sent to the user. This order record has the following characteristics:

- A new order identifier (`nmb`) will be generated.
- Field `sts` will be always 0 (Inactive).
- Field `uty` will have one of the following values: 14="Reject New" (incase of AddOrderBuyCmd, AddOrderSellCmd, AddDealBuyCmd, AddDealSellCmd, ConfirmDealBuyCmd, ConfirmDealSellCmd), 15="Reject Cancel" (in case of CancelOrderCmd, CancelDealCmd, RefuseDealCmd) and 16="Reject Replace" (in case of ChgOrderCmd, SuspendOrderCmd, ReleaseOrderBuyCmd or ReleaseOrderSellCmd commands).
- Field `lnk` will be filled with the order identifier from the rejected command if it exists.
- All the other fields will be filled with values taken from the command when possible, otherwise with default values.

The reject order record will be stored to persistent storage and will be available in order audit reports (OrdersDailyLog).

In case a quote management or cross order command (UpdateMMOrdersCmd or CancelMMOrdersCmd, AddCrossOrdersCmd) is rejected the system will generate a single reject record as for single order commands but the response will be send to the user wrapped in a QuoteResultDto structure. The reject record will be also stored to persistent storage and the `uty` field will be 17="Reject Update MMO", 18="Reject Cancel MMO" or 14="Reject New".

Gateway generated error responses can be split further in two categories.

Error messages generated by the I/O between the client and gateway that will be filled as follows:

- csq=-1
- err=true
- id=-1
- content=ErrorDto
- kmsg=Error description
- ktime=`` (empty)

Error messages generated by the gateway as a result of a communication failure between the gateway and central system, application or business protocol violation. They will be filled as follows:

- csq= The csq of the message request that produced the error
- err=true
- id=The regular id to be expected for the message request that produced the error
- content=ErrorDto
- kmsg=Error description
- ktime=`` (empty)

Outgoing Messages Content Fields

Regular Commands

LoginCmd

This command should be the first to be sent to the gateway after the TCP/IP connection is established in order to start a new session. When receiving this command, the gateway will try to connect and perform the login procedure to the communication server of the central system. The response to this command in case of a successful login is a type 101 message.

Field name	Description
user	User name
passwd	The password
host	Host name or ip address of the communication server to be used by the gateway to connect to the central system
port	Port number of the communication server to be used by the gateway to connect to the central system
url	Url of the report service provided by the central system

LogoutCmd

Logout command has no fields and should be used to terminate the session. When this command is sent to the gateway, the gateway will terminate the connection to the central system's communication server.

HeartBeatCmd

HeartBeat command has no fields and should be used to check the connection status between the client and the gateway. When receiving this message the gateway will echo it back to the client.

LoadSymbolCmd

LoadSymbolCmd is used to *load* an instrument in the user's virtual environment. Once an instrument is loaded the client receives information regarding the symbol at login time and during the session.

Field name	Description
sym	Symbol code

UnloadSymbolCmd

UnloadSymbolCmd is used to *unload* an instrument from the user's environment. Once an instrument is unloaded client will not receive information about the symbol. Prior to *unloading* a symbol the user has to be unsubscribed from any symbol-market entities with the same symbol.

Field name	Description
sym	Symbol code

AddSubscribeCmd

AddSubscribeCmd is used to subscribe the user to a symbol-market entity. Once a symbol-market is in the subscription list, the user receives level1 market data for the symbol-market. Prior to subscribe to a symbol-market the client has to have the symbol *loaded* in his environment.

Field name	Description
sym	Symbol code
mkt	Market code

DelSubscribeCmd

DelSubscribeCmd is used to unsubscribe the user from a symbol-market entity.

Field name	Description
sym	Symbol code
mkt	Market code

GetMarketByOrderCmd

GetMarketByOrder is used get a snapshot of the order book for a symbol-market. The user has to be subscribed to the symbol-market.

Field name	Description
sym	Symbol code
mkt	Market code

MailCmd

MailCmd is used to send a message to another user.

Field name	Description
dst	Destination user code
txt	Text of the message

ChgPasswordCmd

This command is used to change the current password. Note that the following rules apply for the new password:

- The length has to be between 8 and 10 characters
- The password can't be one of the last 12 passwords
- The password can contain only alphanumerical characters (so passwords consisting only in numbers or only in letters are not accepted).

Field name	Description
odp	Old password
nwp	New password
npr	New password

AddOrderBuyCmd and AddOrderSellCmd

Those commands are used to send orders to the exchange.

Field name	Description
sym	Symbol code
mkt	Market code
stt	Settlement term type: 1 = standard, 2 = non standard
clr	Standard settlement date; if stt = 1, clr is one of the standard settlement terms as defined at the symbol level; if stt = 2, clr should be 0
std	Non standard settlement date; if stt = 1, std should be 0; if stt = 2 std should be the actual settlement date (`yyyyMMdd`)
trm	Order term; 0 = Fill or Kill, 1 = Day, 2 = Open, 3 = Good Till Date, 4 = Immediate or Cancel, 5 = Valid for Auction, 6 = Valid for Closing, 7 = Valid for Opening
opd	Open date; if trm = 3, it should be the date until which this order should live (`yyyyMMdd`), or 0 otherwise
ref	Comment
acc	Account number
prc	Price; -1 = Market, 0 = Unpriced, a positive numeric for Limit orders
siz	Volume
ver	Special volume restriction; 0 = NONE

Field name	Description
dcv	Disclosed volume; if the order is hidden, it should be the disclosed volume of the hidden order, or 0 otherwise
tpa	Trigger price type; 1 = None, 2 = Stop, 3 = If Touched, 4 = Trading At Last
tgp	Trigger price; a positive number unless tpa = 1, in which case it should be 0 (zero)
ssl	Short sell mark (used only for sell order); 1=Short Sell, 0=N/A, 2=Short Sell Exempt
iac	Internal account; maximum 15 characters
iwf	Investment decision within firm short code; 0 = the decision is not taken within firm, [1, 2, 3] = reserved numbers not to be used, otherwise any integer
ial	Investment decision within firm algo flag; 'Y' = decision taken by algo, 'N' = decision taken by a human
ewf	Execution responsibility within firm short code; 3 = execution responsibility is not within firm, [0, 1, 2] = reserved numbers not to be used, otherwise any integer
eal	Execution responsibility within firm algo flag; 'Y' = execution responsibility of an algo, 'N' = execution responsibility of a human
dea	Direct electronic access flag; 'Y' = direct electronic access, 'N' = not direct electronic access

CancelOrderCmd

CancelOrderCmd command is used to cancel an order.

Field name	Description
tck	Order identifier
sym	Symbol code
mkt	Market code
ref	Comment
iwf, ial, ewf, eal, dea	see AddOrderBuyCmd and AddOrderSellCmd

ChgOrderCmd

ChgOrderCmd command is used to modify an order. It is forbidden to have both `siz` and `oqy` strictly positive in the command. When `siz` > 0 then an attempt is made to change the order's remaining volume and if successful then the order's remaining volume will be `siz`. When `oqy` > 0 then an

attempt is made to change the order's total volume and if successful then the order's total volume will be `oqy`.

For any given order we must have: `oqy` (total quantity) = `cqy` (cumulated quantity) + `siz` (remaining quantity) so if the `oqy` from the replace command is lower than the cumulated quantity of the order (`cqy`) then the command will be rejected.

ChgOrderCmd command can contain the sequence of the order (in the `osq` field) which will be used by the exchange server to validate whether the order has been modified between the moment the command was issued and the moment the request actually reached the central system. If the field is not provided, no integrity check is performed.

The `osq` field needs to be filled with the value received from the exchange server. This value is unique across the entire exchange and will be internally updated by the server when the order is added in the system or whenever the order changes. The latest order sequence will be disseminated in all order confirmations (OrdDto) and trade confirmations (HalfTrdDto). Client applications which want to ensure that data is up-to-date must get the latest `osq` value from OrdDto and HalfTrdDto objects received from server and use it when issuing ChgOrderCmd commands.

Field name	**Description**
nmb	Order identifier
sym	Symbol code
mkt	Market code
stt	Settlement term type; 1 = standard, 2 = non standard
clr	Standard settlement date; if stt = 1, clr is a number, one of the standard settlement terms as defined in the symbol; if stt = 2, clr should be 0
std	Non standard settlement date; if stt = 1, std should be 0 and if stt = 2 std should be the actual settlement date (`yyyyMMdd`)
trm	Order term; 0 = Fill or Kill, 1 = Day, 2 = Open, 3 = Good Till Date, 4 = Immediate or Cancel, 5 = Valid for Auction, 6 = Valid for Closing, 7 = Valid for Opening
opd	Open date; if trm = 3, it should be the date until which this order should live (`yyyyMMdd`), or 0 otherwise
ref	Comment
prc	Price; -1 = Market, 0 = Unpriced, a positive numeric for Limit orders
tgp	Trigger price
siz	Volume - represents remaining volume
dcv	Disclosed volume; if the order is hidden, it should be the disclosed volume of the hidden order, or 0 otherwise

Field name	Description
oqy	Total volume; this field is not required and has a default value of zero; also note that it's forbidden to have both oqy >0 and size > 0
osq	Order sequence
iwf, ial, ewf, eal, dea	see AddOrderBuyCmd and AddOrderSellCmd

SuspendOrderCmd

SuspendOrderCmd command is used to suspend an active order.

Field name	Description
tck	Order identifier
sym	Symbol code
mkt	Market code
ref	Comment
iwf, ial, ewf, eal, dea	see AddOrderBuyCmd and AddOrderSellCmd

ReleaseOrderBuyCmd and ReleaseOrderSellCmd

Those commands are used to activate a suspended order.

Field name	Description
tck	Order identifier
sym	Symbol code
mkt	Market code
ref	Comment
iwf, ial, ewf, eal, dea	see AddOrderBuyCmd and AddOrderSellCmd

UpdateMMOrdersCmd

This command is used to add/replace a pair of orders (a quote) for an order book and is suitable for market making activities. A quote is in this case a pair of orders, one on the bid side and the other on the ask side and they have the following characteristics:

- The bid price has to be lower that the ask price and the spread has to fall between the configured spread limits.
- A certain trading member can have only one active quote (one bid order and one ask order) in an order book; these orders will have the atribute rol=1
- It's forbidden to have both `sizbuy` and `bqy` strictly positive; `sizbuy` refers to the remaining volume of the order and `bqy` refers to the total volume of the order; the same applies for the sell side
- If an order exists on the buy side with rol=1 we can have one of the following:

- If `sizbuy` > 0 and `bqy` = 0 then the existing order will be replaced and the remaining volume will be `sizbuy`
- If `sizbuy` = 0 and `bqy` > 0 then an attempt will be made to replace the order and set it's total volume to `bqy`; if this is not possible, the command will be rejected with a specific error code

If there is no order on the buy side with rol=1 we can have one of the following situations:

- If `sizbuy` > 0 and `bqy` = 0 then a new order will be placed in the order book
- If `sizbuy` = 0 and `bqy` > 0 then command will be rejected with a specific error code

The above applies for the sell side.

- If one side fails any validations (tick size, block size, price tunnel, etc) the command is rejected.
- Once a quote is accepted the containing orders will be treated as two regular and independent orders. If one order is filled in full, the other will not be removed from the order book.
- If the command is accepted then the response will be a QuoteResultDto with two OrdDto items corresponding to the assigned orders. The accepted orders will have the same `csq` (taken from the command), same update time (`uti`) and will also have rol=1.
- If the command is rejected the response will be a QuoteResultDto with a single reject OrdDto.
- The matching will be performed only after both orders are placed/changed in the order book so no cross trades (at least between two orders from the same trading participant with rol=1) are possible.
- Market Operations or the trading participant can act upon the orders that are part of a quote independently. In this case the user will receive regular unsolicited and unrelated messages that convey the new state of the orders. Also when resumed such an order will have rol=0.
- The gap filling reports will provide plain OrdDto records (they won't be wrapped in QuoteResultDto).

It is possible to specify the side of the quote which will be added or replaced. This is useful particularly when only one of the paired orders, either the buy order or the sell order, needs to be updated. In this case, the issued command will have the side (`sde` field) different that 0 (1 to affect only the Buy side or 2 to affect only the Sell side). If such a command is accepted, the response will be a QuoteResultDto with only one OrdDto item (as opposed to the response of the commands with side 0 which contains two OrdDto items).

The system does not enforce the existence of a quote (a pair of Buy and Sell orders) in the order book at the moment of sending a command with a side different than 0. It is therefore technically possible to place only one order ("a half" of the quote) by using this command (note that the above characteristics are still applicable to the added order). This situation is however less likely to be appear in practice, and the field `sde` will generally have the value 0 when adding new quote orders; and a value different than 0 when there is a need to change only one of the quote orders at a time.

If the field `sde` is not present in the command, or if its value is 0, the command updateMMOrdersCmd will have a regular behavior, as described at the beginning of this section.

Field name	Description
sym	Symbol code
mkt	Market code
stt	Settlement term type: 1 = standard, 2 = non standard
clr	Standard settlement date; if stt = 1, clr is one of the standard settlement terms as defined at the symbol level; if stt = 2, clr should be 0
std	Non-standard settlement date; if stt = 1, std should be 0; if stt = 2 std should be the actual settlement date (`yyyyMMdd`)
trm	Order term; 0 = Fill or Kill, 1 = Day, 2 = Open, 3 = Good Till Date, 4 = Immediate or Cancel, 5 = Valid for Auction, 6 = Valid for Closing, 7 = Valid for Opening
opd	Open date; if trm = 3, it should be the date until which this order should live (`yyyyMMdd`), or 0 otherwise
ref	Comment
acc	Account number
prcbuy	Bid price
prcsell	Ask price
sizbuy	Bid volume; represents remaining buy volume
sizsell	Ask volume; represents remaining sell volume
dcvbuy	Bid disclosed volume; if the order is hidden, it should be the disclosed volume of the hidden order, or 0 otherwise
dcvsell	Ask disclosed volume; if the order is hidden, it should be the disclosed volume of the hidden order, or 0 otherwise
ssl	Short sell mark (used only for sell order); 1=Short Sell, 0=N/A, 2=Short Sell Exempt
iac	Internal account; maximum 15 characters
sde	The side of the quote to place/change; optional field with possible values 0=Both, 1=Buy, 2=Sell and default value of 0 (both sides)
bqy	Buy total volume; not required field, defaults to zero; it's forbidden to have both bqy and sizbuy strictly positive
sqy	Sell total volume; not required field, defaults to zero; it's forbidden to have both sqy and sizsell strictly positive
iwf, ial, ewf, eal, dea	see AddOrderBuyCmd and AddOrderSellCmd

CancelMMOrdersCmd

This command will try to cancel the bid and/or ask orders (the quote) assigned (with rol=1) to the trading participant who sends the command from an order book (identified by the sym/mkt combination). The following assertions apply:

- In case both orders are canceled the QuoteResultDto will contain two OrdDto items corresponding to the canceled orders. Both OrdDto records will have the same csq (taken from the command) and the same update time (uti).
- In case only one order is canceled, the QuoteResultDto will contain only one OrdDto item.
- In case none of the assigned orders is present in the order book or any other error the QuoteResultDto will contain a single reject OrdDto.
- Market Operations or the trading participant can act upon the orders that are part of a quote independently. In this case the user will receive regular unsolicited and unrelated messages that convey the new state of the orders.
- The gap filling reports will provide plain OrdDto records (they won't be wrapped in QuoteResultDto).

It is possible to cancel only one order of the quote by specifying a side (`sde` field different than 0). In this case, the other part of the quote, if it exists in the order book, will remain in the system. If no side is specified or if the `sde` field is 0 then both parts of the quotes will be cancelled.

The response to a canceMMOrdersCmd request with the `sde` field different than 0, will contain at most one OrdDto item.

Field name	Description
sym	Symbol code
mkt	Market code
ref	Comment
sde	The side of the quote to place/change; optional field with possible values 0=Both, 1=Buy, 2=Sell and default value of 0 (both sides)
iwf, ial, ewf, eal, dea	see AddOrderBuyCmd and AddOrderSellCmd

AddCrossOrdersCmd

This command is used to add a pair of orders with the same price and volume but with opposite sides which will be guaranteed to be executed by the exchange. The orders can be entered for the same account or for different accounts. The resulted trade will be publicly marked as a cross trade, but otherwise will be treated as a regular trade (its value will be counted into market statistics, it can trigger contingent orders, etc.)

AddCrossOrdersCmd command will be accepted by the system only during continuous market (Open) or Trading-At-Last sessions and only if the following conditions are met:

- When in Continuous market, the price of the cross orders must be between best bid and best

ask prices (without including them); if there is no best bid price or no best ask price, cross orders will be rejected;

- When in Trading-At-Last, the price of the cross orders must be the same as the TAL price and there must not be any outstanding limit order whose execution priority would be higher than the cross orders' execution priority.

Field name	Description
sym	Symbol code
mkt	Market code
stt	Settlement term type: 1 = standard, 2 = non standard
clr	Standard settlement date; if stt = 1, clr is one of the standard settlement terms as defined at the symbol level; if stt = 2, clr should be 0
std	Non-standard settlement date; if stt = 1, std should be 0; if stt = 2 std should be the actual settlement date (`yyyyMMdd`)
ref	Comment
accbuy	Account number for the buy order
accsell	Account number for the sell order
prc	Price at which the cross orders will be executed
siz	Volume
bia	Internal account for the buy order; maximum 15 characters
sia	Internal account for the sell order; maximum 15 characters
ssl	Short sell flag (used only for sell order); 1=Short Sell, 0=N/A, 2=Short Sell Exempt
biwf, bial, bewf, beal	Buy side short codes for investment decision and execution responsibility, see AddOrderBuyCmd and AddOrderSellCmd for details
siwf, sial, sewf, seal	Sell side short codes for investment decision and execution responsibility, see AddOrderBuyCmd and AddOrderSellCmd for details
dea	Direct Electronic Access, see AddOrderBuyCmd and AddOrderSellCmd

AddDealBuyCmd and AddDealSellCmd

Those commands are used to send deal proposals to the exchange.

Field name	Description
sym	Symbol code

Field name	Description
mkt	Market code
stt	Settlement term type: 1 = standard, 2 = non standard
clr	Standard settlement date; if stt = 1, clr is one of the standard settlement terms as defined at the symbol level; if stt = 2, clr should be 0
std	Non standard settlement date; if stt = 1, std should be 0; if stt = 2 std should be the actual settlement date (`yyyyMMdd`)
trm	Order term; 0 = Fill or Kill, 1 = Day, 2 = Open, 3 = Good Till Date, 4 = Immediate or Cancel, 5 = Valid for Auction, 6 = Valid for Closing, 7 = Valid for Opening
opd	Open date; if trm = 3, it should be the date until which this order should live (`yyyyMMdd`), or 0 otherwise
ref	Comment
acc	Account number
prc	Price; -1 = Market, 0 = Unpriced, a positive numeric for Limit orders
siz	Volume
iac	Internal account; maximum 15 characters
tou	The username of the destination
grs	Gross flag; 1 = Gross, 0 = Net
iwf, ial, ewf, eal, dea	see AddOrderBuyCmd and AddOrderSellCmd

ConfirmDealBuyCmd and ConfirmDealSellCmd

Those commands are used to send deal confirmations to the exchange.

Field name	Description
nmb	The identification of the deal that is to be confirm with this command
sym	Symbol code
mkt	Market code
acc	Account number
iac	Internal account; maximum 15 characters
ref	Comment
grs	Gross flag; 1 = Gross, 0 = Net
iwf, ial, ewf, eal, dea	see AddOrderBuyCmd and AddOrderSellCmd

CancelDealCmd

The command is used to cancel an outstanding deal

Field name	Description
tck	The identification of the deal that is to be canceled
sym	Symbol code
mkt	Market code
ref	Comment
iwf, ial, ewf, eal, dea	see AddOrderBuyCmd and AddOrderSellCmd

RefuseDealCmd

The command is used to refuse a previously received deal

Field name	Description
tck	The identification of the deal that is to be canceled
sym	Symbol code
mkt	Market code
ref	Comment
iwf, ial, ewf, eal, dea	see AddOrderBuyCmd and AddOrderSellCmd

Report Requests

All report requests have 2 fields in common:

Field name	Description
dir	Direction (0 = Refresh, 1 = Forward, -1 = Backward)
str	Start record; if dir = 0, str should be 0 as well

Those two fields are used to control the report's result set navigation. A report response message will have fields that signal if the received page is the last one or not and a pointer that specifies the current number of the first record of the page.

FindAccountCmd

This command is used to retrieve the information associated with a certain account.

Field name	Description
acc	Account number for the account to be found
dir, str	

Find2AccountCmd

This command is used to retrieve the account information for an account pair instead of a single account.

Field name	Description
ac1	Account number for the first account to be found
ac2	Account number for the second account to be found
dir, str	

FindFreshOrderReportCmd

This command is used to get details about an order by order number.

Field name	Description
tck	Order identifier
dir, str	

GetGenericOrderAuditCmd

This command is used to get the audit of a particular order.

Field name	Description
nmb	Order identifier
dir, str	

GetOrdersDailyLogCmd

This command is used to get the daily activity log for the current member.

Field name	Description
tms	Start time (hh:mm:ss)
tme	End time (hh:mm:ss)
dir, str	

GetOutstandingOrdersCmd

This command is used to get all the outstanding orders for the current member.

Field name	Description
sts	Status of orders to be retrieved; 1=Active, 2=Suspended, 0=All
sym	A particular symbol code or * for all symbols

mkt	A particular market code or * for all markets
sde	Side of the orders to be retrieved; 1=Buy, 2=Sell, 0=All
acc	A particular account number or 0 (zero) for all accounts
dir, str	

GetDailyTradesCmd

This command is used to get all the trades from the current date for the current member.

Field name	Description
tms	Start time (hh:mm:ss)
tme	End time (hh:mm:ss)
dir, str	

GetUsersByNameCmd

This command is used to get the connected users or users defined at a certain member.

Field name	Description
mbr	A particular member code or * for all connected users
dir, str	

GetSymbolsByNameCmd

This command is used to get the symbols from the central system based on whether or not they are loaded into the user's virtual environment.

Field name	Description
ldd	Load selector; 0 - Unloaded, 1 - Loaded, 2 - All
dir, str	

GetDailyPublicTradesCmd

This command is used to get all the public trades from the current date.

Field name	Description
tms	Start time (hh:mm:ss)
tme	End time (hh:mm:ss)
dir, str	

GetDailyIndicesCmd

This command is used to get the intraday values of the indices for the current date.

Field name	Description
tms	Start time (hh:mm:ss)
tme	End time (hh:mm:ss)
dir, str	

GetPublicParametersCmd

This command is used to get the parameters for all markets as well as for specific symbol-market entities. It does not have any fields except dir and str.

GetStepsCmd

This command is used to get all price steps lists. It does not have any fields except dir and str.

GetUnderlyingsCmd

This command is used to get all current underlying symbols. It does not have any fields except dir and str.

Data Transfer Objects

ResultPageDto (report page wrapper)

Field name	Description
lines	List of records
start	Start record number
end	End record number
crtpage	Current page number
bottom	If true it means the page is the last of the result set

UserEnvDto (login exchange snapshot)

Field name	Description
username	User name
brokercode	Broker code
brokername	Broker name

Field name	Description
exp	Password expired flag; if it is true the user has to change the password in 5 minutes otherwise it will be disconnected from the system
tmw	Automatic timeout in milliseconds; the client should send the next command only after tmw timeout elapsed from the time the response to a previous command arrived.
sysname	Central system's name
sbs	Subscription mechanism flag; 0=Off (in this case it is not required to subscribe to a symbol market in order to receive live market data), 1 = On
mp	A snapshot of the exchange entities at the time the user logged in as an MarketPictureDto object
loginmessage	This is the first message every user may receive upon login (usually important announcements)

MarketPictureDto (full exchange picture)

Using the information contained in this object the client can build a *virtual* exchange structure. The building blocks of this structure are exchange entities delivered as ExchangeExplorerDto objects.

Field name	Description
eeml	A list of ExchangeExplorerDto objects

ExchangeExplorerDto (exchange structure element)

Field name	Description
action	This field is used to identify what kind of operation should be applied to the *virtual* exchange structure. 1=Add: add the element present in the `value` field to the exchange structure -1=Delete: delete the element present in the `value` field from the exchange structure 0=Update: the element present in the `value` field should replace the element already present in the *virtual* exchange structure kept at the client site. At login time MarketPictureDto will contain only ExchangeExplorerDto objects with action=1 so the client can build the *virtual* structure of the exchange. After login when TickersPack objects can arrive, ExchangeExplorerDto objects embedded inside them can have the `action` field filled with any of the above values.

Field name	Description
type	The type of the field `value`; possible values of this field are: • 0 = MPExcDto/ExcDto (Exchange entity) • 1 = MPStyDto/StyDto (Symbol-type entity) • 2 = MPIdxDto/IdxDto (Index entity) • 3 = MPSmkDto/SmkDto (Symbol-market entity) • 4 = MPMktDto/MktDto (Market entity) • 5 = CnvDto (Currency entity) • 6 = MPSymDto/SymDto (Symbol entity) • 10 = UlyDto * (Underlying symbol entity) • 11 = ParamsShortDto (Exchange level parameter) * Underlying symbol entities (ExchangeExplorerDto of type 10) will not be sent at login, in the market picture. They are only received after login, during the session, when underlying symbol changes occur (at addition, update, and deletion). The list of all current underlying symbols can be retrieved on demand, after login, through the command GetUnderlyingsCmd. IMPORTANT: The type will be a MP* object at login (MarketPictureDto) but during the session, if changes occur, the type will be plain dto objects, except that, if during the session an ExchangeExplorerDto is received with action=1 and type=6, the type of the `value` field will be a MPSymDto.
value	Depends on the `type` field, see above

MPExcDto (exchange entity)

Field name	Description
sum	Exchange summary as a ExsInfo object
bdy	Exchange properties as a ExcDto object

ExsInfo (exchange summary)

Field name	Description
efd	Effective date (`yyyyMMdd`)
tva	Total value traded for spot markets
tvo	Total volume for spot markets
ttr	Total number of trades for spot markets
tup	Number of up symbols for spot market
tdw	Number of down symbols for spot market

tst	Number of unchanged symbols for spot market
fva	Total notional value for futures markets
fvo	Toal volume for futures markets
ftr	Total number of trades for futures markets
fup	Number of up symbols up for futures markets
fdw	Number of down symbols for futures markets
fst	Number of unchanged symbols for futures markets
foi	Open interest
uti	Last update time

ExcDto (exchange properties)

Field name	Description
sts	Exchange Status 1=Opened 2=Suspended 2=Closed 4=Maintenance
ref	Comment
uty	Update Type 1=New 0=Deleted 2=Changed
rky	Exchange Code
chs	Clearing holidays array; a semicolon separated list of dates in `yyyyMMdd` format
sbs	Subscribe mechanism flag 1=On 0=Off
nam	Exchange name
ovn	Overwrite NIN permit 1=Yes 0=No

Field name	Description
unm	Single account for a person at one member 1=Yes 0=No
rio	Internal registry operations permission flag 2=Suspended for regular users 0=Suspended for all users (including batch operations) 1=Permitted for all users
reo	External registry operations permission flag 2=Suspended for regular users 0=Suspended for all users (including batch operations) 1=Permitted for all users

MPMktDto (market entity)

Field name	Description
sum	Market summary as MksInfo structure
bdy	Market properties as MktDto structure

MksInfo (market summary)

Field name	Description
efd	Effective date (`yyyyMMdd`)
tva	Trading value
tvo	Trading volume
ttr	Trades count
tup	Number of up symbols
tdw	Number of down symbols
tst	Number of unchanged symbols
uti	Update time

MktDto (market properties)

Field name	Description

sts	Market status 1=Opened 2=Suspended 3=Closed 4=Maintenance 5=Preopened 6=Preclosed 7=Trading-at-last
ref	Comment
uty	Update type 1=New 0=Deleted 2=Changed
uti	Update time
mkt	Market code
nam	Market name
typ	Market type A comma separated list of values (orders, deals, quotes)
atp	Account type priority 1=Yes 0=No
cls	Instrument type traded on this market Possible values: share, bond, bill or future

MPSmkDto (symbol-market entity)

Field name	**Description**
sum	Symbol-market summary as a SmsInfo object
bdy	Symbol-market properties as a SmkDto object
sbs	Subscription flag: 1=Yes, 0=No

SmsInfo (symbol-market summary)

Even if the user is subscribed to a symbol-market (sbs=1), if there is no market activity on that

symbol-market (all the fields except uti and efd are zero) then the whole object will be null (in order to save traffic). The client can build an empty SmsInfo (with all fields set to zero) and take the efd and uti from the ExsInfo.

If the user is not subscribed to the symbol-market (sbs=0) than one should not make any assumptions regarding the contents of the SmsInfo.

Field name	Description
efd	Effective date (`yyyyMMdd`)
vol	Traded Volume
val	Trading value (in trading currency of the symbol)
vad	Clearing value (in clearing currency of the symbol)
mny	Trading value (in system's national currency)
trd	Trades count
opn	Open price
cls	Close price
avg	Average price
low	Low price
hig	High price
bbp	Best buy price
bbv	Best buy volume
bsp	Best sell price
bsv	Best sell volume
dir	Direction (+1,-1, 0) from previous trade
clv	Close volume (last trade volume)
omp	Fixing price
omv	Fixing volume
uti	Update time

SmsDto (extended symbol-market summary)

This structure extends the SmsInfo structure by adding two fields that can identify the symbol-market it represents. It comes as a response to an AddSubscribeCmd so that the client has the first snapshot for the symbol-market summary. Note that if the client was subscribed to a symbol-market prior to login, the symbol-market summary will come as a SmsInfo object at login and it is not necessary to issue the subscribe after login.

Field name	Description
sym	Symbol code

mkt	Market code

SmkDto (symbol-market properties)

Field name	Description
sts	Symbol-market status 1=Opened 2=Suspended 3=Closed 4=Opening/Closing/Fixing 5=Preopened 6=Preclosed 7=Trading-at-last 8=Preopened-XT 9=Preclosed-XT 10=Volatility-Interruption 11=Volatility-Interruption-XT
ref	Comment
sym	Symbol code
mkt	Market code
pmk	Primary market flag 1=Yes 0=No

MPStyDto (symbol-type entity)

Field name	Description
sum	Symbol-type summary as a StsInfo object
bdy	symbol-type properties as a StyDto object

StsInfo (symbol-type summary)

Field name	Description
efd	Effective date (`yyyyMMdd`)
tva	Trading value
tvd	Clearing value
mny	Trading value (in system's national currency)
tvo	Trading volume
ttr	Trades count

tup	Symbols up
tdw	Symbols down
tst	Symbols unchanged
uti	Update time
foi	Open interest (relevant only for futures)

StyDto (symbol-type properties)

Field name	Description
sty	Symbol-type code
nam	Symbol-type name
sts	Symbol-type status 1=Active 0=Inactive
ref	Comment
uty	Update type 1=New 0=Deleted 2=Changed
uti	Update time
cls	Symbol class Possible values: bill, bond, share, future
trc	Trading currency
clc	Clearing currency

MPSymDto (symbol entity)

Field name	Description
key	Symbol code
sty	Symbol-type code
sum	Symbol summary as a SysInfo structure
bdy	Symbol properties; the type of this field depends of the symbol class

SysInfo (symbol summary)

Field name	Description
crf	Current reference price; for futures this is the settle price
foi	Open interest (relevant only for futures)

SymDto (symbol entity)

Field name	Description
key	Symbol code
sty	Symbol-type code
nam	Symbol name
bdy	Symbol properties; the type of this field depends of the symbol class

ShareContentDto (properties for shares)

Field name	Description
cat	Symbol category
cls	Symbol class (always `share`)
cpd	Price decimals
dst	Default settlement term
pcd	Percentage change decimals
pcp	Nominal value
pnd	Pending flag 1=Yes 0=No
psl	Price step list code
reo	External registry operations flag 2=Suspended for regular users 0=Suspended for all users 1=Permitted for all users
rgl	Registry link 1=On 0=Off

Field name	Description
rio	Internal registry operations flag 2=Suspended for regular users 0=Suspended for all users 1=Permitted for all users
rky	Symbol code
rpr	Reference price
rpt	Reference price type 1=Average 2=Close
sct	Standard clearing terms - a comma separated list of settlement terms as integers
sts	Symbol status 1=Ready 2=Suspended 3=Delisted
sty	Symbol-type code
uti	Update time
uty	Update type 1=New 0=Deleted 2=Changed
uuv	Use Unit Value 1=Yes 0=No
vld	Trading value decimals
vsd	Clearing value decimals

BillContentDto (properties for bills)

Field name	Description
cat	Symbol category
cls	Symbol class (always `bill`)
cpd	Clean price decimals
dpd	Dirty price decimals

Field name	Description
dst	Default settlement term
dys	No. of days in a year
isd	Issue date (`yyyyMMdd`)
lsd	Last settlement date (`yyyyMMdd`)
mtd	Maturity date (`yyyyMMdd`)
pcd	Percentage change decimals
pcp	Principal value
pnd	Pending flag 1=Yes 0=No
psl	Price step list code
reo	External registry operations flag 2=Suspended for regular users 0=Suspended for all users 1=Permitted for all users
rgl	Registry link 1=On 0=Off
rio	Internal registry operations flag 2=Suspended for regular users 0=Suspended for all users 1=Permitted for all users
rky	Symbol code
rpr	Reference price
rpt	Reference price type 1=Average 2=Close
sct	Standard clearing terms - a comma separated list of settlement terms as integers

Field name	Description
sts	Symbol status 1=Ready 2=Suspended 3=Delisted
sty	Symbol-type code
uti	Update time
uty	Update type 1=New 0=Deleted 2=Changed
uuv	Use Unit Value 1=Yes 0=No
vld	Trading value decimals
vsd	Clearing value decimals

BondContentDto (properties for bonds)

Field name	Description
cat	Symbol category
cls	Symbol class (always `bond`)
cpd	Clean price decimals
cps	Coupons list. See the coupons syntax for more details
dpd	Dirty price decimals
dst	Default settlement term
isd	Issue date (`yyyyMMdd`)
lsd	Last settlement date (`yyyyMMdd`)
pcd	Percentage change decimals
pnd	Pending flag 1=Yes 0=No
psl	Price step list code

Field name	Description
reo	External registry operations flag 2=Suspended for regular users 0=Suspended for all users 1=Permitted for all users
rgl	Registry link 1=On 0=Off
rio	Internal registry operations flag 2=Suspended for regular users 0=Suspended for all users 1=Permitted for all users
rky	Symbol code
rpr	Reference price
rpt	Reference price type 1=Average 2=Close
sct	Standard clearing terms - a comma separated list of settlement terms as integers
sts	Symbol status 1=Ready 2=Suspended 3=Delisted
sty	Symbol-type code
uex	Use ex-coupon flag 1=Yes 0=No
uti	Update time
uty	Update type 1=New 0=Deleted 2=Changed

Field name	Description
uuv	Use Unit Value 1=Yes 0=No
vld	Trading value decimals
vsd	Clearing value decimals

FutureContentDto (properties for futures)

Field name	Description
cat	Symbol category
cls	Symbol class (always `future`)
cpd	Price decimals
dst	Default settlement term
mtd	Maturity date (`yyyyMMdd`)
pcd	Percentage change decimals
cps	Underlying symbol
pcp	Multiplier
pnd	Compute Open Interest 1=Yes 0=No
psl	Price step list code
rgl	Clearing house link 1=On 0=Off
rky	Symbol code
rpr	Reference price (previous settle)
rpt	Settle price type 1=Average 2=Close
sct	Standard clearing terms - a semicolon separated list of settlement terms

Field name	Description
sts	Symbol status 1=Ready 2=Suspended 3=Delisted
sty	Symbol-type code
uti	Update time
uty	Update type 1=New 0=Deleted 2=Changed
uuv	Use Unit Value 1=Yes 0=No
vld	Trading value decimals
vsd	Clearing value decimals

MPIdxDto (index entity)

Field name	Description
sum	Index summary as an IdxInfo object
bdy	Index properties as an IdxInfo object

IxsInfo (index summary)

Field name	Description
cla	Absolute change
clp	Percentage change
cls	Close value (last value)
efd	Effective date
hig	High value
low	Low value
opn	Open value (previous close)
uti	Update time

IdxDto (index properties)

Field name	Description
idx	Index code
mlt	Multiplier
nam	Index name
pcd	Percentage change decimals
ref	Comment
sts	Index status 1=Ready 2=Suspended
uti	Update time
uty	Update type 1=New 0=Deleted 2=Changed
vld	Absolute value decimals

CnvDto (currency properties)

Field name	Description
sym	Currency code
rat	Rate against RON
ref	Comment
uti	Update time

ErrorDto

Field name	Description
errorStatus	Error code 100=Only one client can connect to the gateway 101=The gateway is not connected 102=The gateway is already connected 103=The gateway is busy 104=Response timed out 105=Exception on message decoding / parsing / schema validation 106=Unknown command type 107=GetMarketByOrderCmd was issued for a subscribed symbol-market but the cache initialization process did not finished. Issue the command after the end init cache message arrives 108=GetMarketByOrderCmd was issued for an unsubscribed symbol-market. Subscribe first to the symbol-market 109=The client sequence should be greater than zero

MailDto

Field name	Description
src	From user
txt	Text message

HalfTrdDto (trade)

Field name	Description
ext	External trade flag 1=Yes 0=No
sty	Symbol-type code
cls	Symbol class (share, bond, bill or future)
sde	Side 1=Buy 2=Sell

Field name	Description
sts	Trade status 1=Active 0=Canceled
mkt	Market code
tss	Allocated trade sequence
tcs	Clear trade sequence
tck	Ticket number
trt	Trade time. If uty = 1 (new trade) it will be identical with the `uti` field of the orders that participated to this trade.
sym	Symbol code
prc	Trade price
dtp	Trade dirty price
siz	Trade size
val	Trade value (clearing currency)
vlt	Trade value (trading currency)
brk	Broker code
mbr	Member code
acc	Account number
act	Account type 1=Client 2=Financial 3=House 4=Staff 5=Insider 6=Mixed
grp	Group account number
alv	Allocated volume
ava	Allocated value
clv	Cleared volume
ord	Order number
uid	User id of the last user that changed the order
ini	User id of the user that created the order
bnk	Settlement bank code
bka	Settlement bank account
std	Settlement date (`yyyyMMdd`)

Field name	Description
din	Deal indicator >0 = the number of the order that initiated the trade 0=No
fst	Settlement flag 1=Yes (the trade will be cleared by Arena) 0=No (the trade will be cleared on another clearing system)
clt	Cleared flag 1=Yes 0=No
grs	Gross settlement flag 1=Yes 0=No
ssl	Short sell flag 1=Short Sell 0=N/A 2=Short Sell Exempt
fal	Allocation flag 1=Yes (the trade has to be allocated) 0=No
ald	Allocated flag 1=Yes (trade was allocated) 0=No
okt	Old ticket - greater than zero if this trade was introduced as a correction of a trade and represents the ticket of the corrected trade
ref	Comment

Field name	Description
uty	Update type 1=New 2=Changed 0=Deleted 3=Allocate 4=Settle 5=Deallocate 6=Discrete settle
uti	Update time
clc	Clearing currency
lqi	Liquidity indicator 0=N/A 1=Aggressive Buy 2=Aggressive Sell 3=Route Out 4=Auction 5=Cross
iac	Internal account
osq	Order sequence
neg	Negotiated trade indicator 'N' = Negotiated trade '1' = Negotiated trade in liquid financial instruments '2' = Negotiated trade in illiquid financial instruments '3' = Negotiated trade subject to conditions other than the current market price 'A' = No negotiated trade or field not present '-' = N/A
alg	Algo flag 'Y' = Algorithmic trade 'N' = No algorithmic trade '-' = N/A

OrdDto (order)

Field name	Description
own	Owner flag 1=Yes 0=No
sts	Order status 1=Active 0=Inactive 2=Suspended
	Indicative flag 1=Yes 0=No
hdi	Hidden flag 0=No - normal order -1=hidden order indicator >0=hidden value for an order
ver	Volume execution restriction 1=Minimum fill 2=Minimum block 3=All of None 0=None
tpa	Trigger type 1=None 2=Stop 3=If Touched 4=TAL
bok	Order book 1=Regular book 2=Special order book 3=Contingent order book
hdv	Visible volume of a hidden order 0 for normal order Positive value for an hidden order(hdi <>0)

Field name	Description
nmb	Order number
csq	Client sequence
lnk	Link to order If this order is a side of a quote, lnk is the number of the other side of the quote If it is a confirm deal, lnk is the target deal. Otherwise lnk=0
eft	Effective time; used to determine the priority in the order book
sde	Order side 1=Buy side 2=Sell side
ssl	Short sell flag 1=Short Sell 0=N/A 2=Short Sell Exempt
trm	Order term 2=Open 1=Day 0=Fill or Kill 3=Good Till Date 4=Immediate or Cancel 5=Valid for Auction 6=Valid for Closing 7=Valid for Opening
opd	Last date of availability (`yyyyMMdd`)
prt	Input Price Type 0=NA (Not available for records created before this field was introduced) 1=Market 2=Unpriced 3=Limit

Field name	Description
odt	Order type 1=Regular 2=Cross 3=Quote 4=Deal
rgr	Registry reference
sym	Symbol code
sty	Symbol-type code
mkt	Market code
clr	Standard Settlement term
std	Settlement date (`yyyyMMdd`)
stt	Settlement type 1=Standard 2=Non standard
brk	Broker code (trading member)
mbr	Member code (clearing member)
trd	Last trade number this order was a part of
acc	Account number
act	Account type
mkp	Market order flag 1=Yes 0=No
tob	Sent to broker code (only for deals)
tou	Sent to user (only for deals)
prc	Price
tgp	Trigger price
siz	Size
dcv	Disclosed volume; for normal orders will be 0 or a positive value for hidden orders
shv	Volume to be publicly displayed
bnk	Clearing bank code
bka	Clearing bank account

Field name	Description
grs	Gross settlement flag 1=Gross settlement 0=Net settlement
ref	Comment
uty	Update type 1=New 0=Deleted 2=Changed 3=Filled 4=Rejected 5=Confirmed 6=Released 7=Suspended 8=Activated 9=Rejected FOK 10=Rejected Odd Lot FOK 11=Rejected Out of Term 12=Rejected Out of Price 13=Rejected Cross Account 14=Reject New 15=Reject Cancel 16=Reject Replace 17=Reject Update MMO 18=Reject Cancel MMO
uti	Update time
uui	Update user
ini	Initiator user
iac	Internal account
oqy	Order Quantity
cqy	Cumulated Quantity
lqy	Last Quantity
apx	Average Price (volume weighted)
lpx	Last Price

Field name	Description
ost	Order Status (FIX) -1=N/A 0=New 1=Partial Fill 2=Fill 3=Done 4=Canceled 7=Stopped 8=Rejected 9=Suspended 10=Pending New 12=Expired
cli	Client Order Id (FIX)
ocl	Original Client Order Id (FIX)
txt	Comment (reject reason), see Reject Codes section
rol	Order role 0=N/A 1=MM (Market Maker); orders placed with UpdateMMOrdersCmd will have rol=1 2=Cross (Cross Order); orders placed with AddCrossOrders will have rol=2
osq	Order sequence, a non-negative unique number assigned by the central system at order creation and at each subsequent order update (either requested or unsolicited). The combination of fields *[nmb, uti, osq]* is unique across all order updates.
iwf	Investment decision within firm short code 0 = the decision is not taken within firm or N/A integer = the short code identifying the human or algorithm within firm that took the decision
ial	Investment decision within firm algo flag 'Y' = decision taken by algo 'N' = decision taken by a human '-' = N/A

Field name	Description
ewf	Execution responsibility within firm short code 0 = N/A 3 = execution responsibility is not within firm integer = the short code identifying the human or algorithm within firm responsible with the execution
eal	Execution responsibility within firm algo flag 'Y' = execution responsibility of an algo 'N' = execution responsibility of a human '-' = N/A
dea	Direct electronic access flag 'Y' = direct electronic access 'N' = not direct electronic access '-' = N/A
uws	The source of modification; in case the order was changed by the matching engine this field will be set to `system` otherwise it will contain the IP address from which the user designated in `uui` was connected

QuoteResultDto

This is just a wrapper for a list of OrdDto objects and is used to convey the response for UpdateMMOrdersCmd and CancelMMOrdersCmd.

Field name	Description
execs	A list of execution (OrdDto object) reports

MboDto (level2 snapshot)

Field name	Description
sym	Symbol code
mkt	Market code
reg	Regular order book; a list of OrdDto objects
spc	Special order book; a list of OrdDto objects
opp	Potential fixing price
opv	Potential fixing volume

ixb	Fixing depth on buy side
ixs	Fixing depth on sell side

AccDetailsDto (account and person details)

Field name	Description
acc	Account number
sts	Account status 3=Locked 4=Closed 1=Opened 5=Locked for buy 6=Locked for sell
atp	Account type 1=Client 2=Financial 3=House 4=Staff 5=Mixed
fnc	Function 0=Individual 1=Group 2=Aggregate
alc	Allocation type 0=None 1=Fifo 2=Prorata
acl	Account is eligible for access lists (for custodians only) 1=Yes 0=No
fav	Favorite account flag 1=Yes 0=No

brk	Member Code
nin	NIN (person identifier)
nam	Name
ref	Record reference (account reference)
uti	Update time
uty	Update type 1=New 0=Deleted 2=Changed
clt	Client type 1=Natural person 2=Legal person 0=Generated identifier (only for groups)
csp	Citizenship code
cty	Country code
cit	City
dst	District
zip	Zip code
ad1	Address line 1
ad2	Address line 2
phn	Phone
fax	Fax
gsm	Mobile phone
eml	Email
ctc	Contact person name
bnk	Bank code
bka	Bank account
nrf	Record reference (NIN reference)

AccDto (account details - see AccDetailsDto)

NinDto (person details - see AccDetailsDto)

UsrDto (account and person details)

Field name	**Description**

rky	User code
bky	Member code
nam	User name
cst	Connected flag (1=Connected, 0=Disconnected)

SylDto (symbol info)

Field name	Description
sts	Status (1 = Ready, 2 = Suspended, 3 = Delisted)
ldd	Loaded flag (1 = Loaded, 0 = Not loaded)
sty	Symbol type
cls	Symbols class (share, bond, tbill, future)
sym	Symbol code
den	Symbol name
isi	Symbol's ISIN

PriceStepDto (price step)

Field name	Description
sym	Price step code
bas	Price step value
sup	Superior limit
npp	Number of protection price steps

ParamsShortDto (entity parameter)

For this structure the (sym, mkt) combination defines the scope of the parameter. A (*, *) combo is applicable at exchange level. A (*, MARKET) combo applies the parameter at market level and a (SYMBOL,MARKET) combo applies to the symbol-market level and overrides the market level. For example if we have priceup=15 at REGS market level (sym=*, mkt=REGS) then priceup=15 will be applied to any symbol that is traded in the REGS market, but in the same time if we have also priceup=0 at TLV/REGS symbol-market level (sym=TLV,mkt=REGS) then for TLV priceup will be 0 for REGS market. If a parameter is missing then the default value applies.

Market and symbol-market level parameters snapshots are available through the GetPublicParametersCmd report and updates are transmitted as `790` messages carrying ParamsShortDto.

Exchange level parameters snapshots are delivered embedded in the market picture at login time and updates are transmitted as ExchangeExplorerDto messages (see Processing Exchange Entities for details).

Field name	Description
uty	Update type (0 - delete , 1 - add, 2 - change)
uti	Update time
sym	Symbol code
mkt	Market code
rky	Parameter code (see next table for an enumeration of possible codes and their default values).
bdy	Parameter value

Below is the list of common market and symbol-market level params.

Code	Name	Values	Default	Description
priceup	price up limit	number	0	Upper limit for price tunnel; if pricelimit=percentage and priceup=0 then there is no upper limit
pricedown	price down limit	number	0	Lower limit for price tunnel; if pricelimit=percentage and pricedown=0 then there is no lower limit
blocksize	block size	number	0	Trading block size
blocksmax	maximum number of blocks	number	0	Maximum number of blocks per order
openmaxage	open orders max age	number	0	Maximum number of days until an open order is deleted from the system if it was not changed
blocksmin	minimum number of blocks	number	0	Minimum number of blocks per order
prcdecs	input price decimals	number	-1	number of permitted decimals;if -1 is not effective
open	open terms permission	y/n	n	Permits open (GTC) orders
onlyopen	only open terms permissio	y/n	n	Permits only open orders
hidden	hidden order permission	y/n	n	Permits hidden orders
special	special volume restriction	y/n	n	Permits orders with special volume restrictions (AON, etc)

Code	Name	Values	Default	Description
contingent	contingent order permission	y/n	n	Permits contingent orders (Stop, If Touched)
fok	fill or kill term permission	y/n	y	Permits FOK orders
pricelimit	price limit type	percentage/ absolute/ fixed	percentage	Price tunnel computation mode: percentage=[ReferencePrice*(1-pricedown/100), ReferencePrice*(1+priceup/100)], absolute=[ReferencePrice-pricedown, ReferencePrice+priceup], fixed=[pricedown, priceup]
qmatch	quotes maching permission	y/n	n	Permits matching between quotes
crossacc	cross account permision	y/n	y	Permits same account trades
dlevel	duplicate level price permision	y/n	n	Permits orders on the same account, same side and same price
singleq	unique quote per member restiction	y/n	n	If set to y(es) a member can post enter only one quote
dealp	deal permission	y/n	n	Permits deals
quotep	quote permission	y/n	n	Permits quotes
orderp	order permission	y/n	n	Permits orders
targetup	active connection for deal target	y/n	n	If set to y(es) a deal counterparty has to be connected
hymtype	hybrid matching type	plain/ discret/ none	plain	If none hybrid matching (orders with quotes) is no permitted; if plain an order can match with multiple quotes; if discret - an order can match with a single quote
oddlotside	odd lot target side	buy/sell/ none	none	The side for odd lot markets
usestep	use price steps list	y/n	y	Check the order price against the symbol’s price step list
indq	indicative quotes	y/n	n	If set to y(es) quotes and orders are indicative (as opposed to ferm)

Code	Name	Values	Default	Description
qhouse	house account restriction for quotes	y/n	n	Quotes can be entered only for house accounts
maxnetval	maximum netvalue per order	number	0	Maximum value for a deal entered for gross settlement
maxval	maximum value per order	number	0	Maximum value for an order or deal
minval	minimum value per order	number	0	Minimum value for an order or deal
qdfprcmn	quote prices minimum diference	number	0	Quotes minimum spread
qdfprcmx	quote prices maximum diference	number	0	Quotes maximum spread
usedefst	use only default settlement term	y/n	y	Permits only orders with default setttlement term
tdypermit	today settlement permit	y/n	y	Permits orders with settlement date se to T (or the current date)
dscmax	maximum disclosed %	number	0	Maximum disclosed volume (in percentage) for a hidden order; if 0, no restriction applies
dscmin	minimum disclosed %	number	0	Minimum disclosed volume (in percentage) for a hidden order; if 0, no restriction applies
extgross	extended (full) gross interval	y/n	n	If set to y(es) the value of a gross deal is inside [minval, maxval], else [maxnetval, maxval]
usegroup	use group account	y/n	n	Permits the use of group accounts
usemixed	use mixed group account	y/n	n	Permits the use of mixed group accounts

Code	Name	Values	Default	Description
enabled	enable trading	y/n/p	y	If set to n(o) orders operations will not be accepted regardless the market status; if set to p only delete and suspend order operations are permitted if the market or symbol-market status permits it; if set to y(es) any order operations are permitted if the market or symbol-market status permits it
allprices	unpriced order matches all prices	y/n	n	If set to y(es) an unpriced order will execute at multiple price levels until it's volume is't depleted
useagg	use aggregate account	y/n	n	Use only aggregate and house accounts
inspread	confirm deal in spread	y/n	n	The price of a deal can be only inside the spred of the orders from the main market
fullfok	full fok	y/n	y	If set to y(es) a FOK order will be executed only if it's volume can be traded in full
uprcmode	unpriced order mode	none/ fulltime/ inopen	none	Market phase in witch the unpriced order can be entered
mktmode	mkt order mode	none/ fulltime/ inopen	none	Market phase in witch the market order can be entered
fxcls	use last price for fixing	y/n	n	If set to y(es) the fixing algorithm will use the last price if there is at least one trade
linkedsts	linked status	y/n	y	If set to n(o) the symbol-market will not follow the status change of the market
includetl	include in trading limit computation	y/n	n	If set to n(o) trades carried out on the market or symbol/market will not be taken into account when the daily spot exposure is computed

Code	Name	Values	Default	Description
uptickrule	Zero uptick rule enablement	y/n	y	If set to y(es) zero uptick rule will be enforced against SSH sell orders. If set to n(o) then zero uptick rule is disabled.
mmsprdtype	Spread type for market makers	percentage/ absolute/ none	none	Spread computation mode: if percentage spread = round_half_up[(ask-bid)/bid, 6 decimals]*100; if absolute spread = ask - bid; if none then no checks are performed.
mmsprdmin	Miminum spread for market makers	number	0	If zero there is no minimum spread
mmsprdmax	Maximum spread for market makers	number	0	If zero there is no maximum spread
vfo	permit vfo order type	y/n	n	Permits orders with VFO term
vfc	permit vfc order type	y/n	n	Permits orders with VFC term
vfa	permit vfa order type	y/n	n	Permits orders with VFA term
cross	permit cross order type	y/n	n	Permits cross orders (orders added through AddCrossOrdersCmd)
ioc	permit ioc order type	y/n	n	Permit orders with IOC term
tgttal	permit tgttal order type	y/n	n	Permits orders with TAL trigger type
mktal	permit market order in TAL	y/n	n	Permits market orders when status is TAL
extal	use extended TAL price determination	y/n	n	Use extended price determination mode, which allows markets to enter TAL phase even if there was no fixing price in the closing auction.
tmode	trading mode	auction/continuous	continuous	Trading mode
poxt.dur	preopen-xt duration	number	10	The duration in seconds of the Preopen-XT market state
pcxt.dur	preclose-xt duration	number	10	The duration in seconds of the Preclose-XT market state

Code	Name	Values	Default	Description
vixt.dur	vixt duration	number	10	The duration in seconds of the ViXT market state
vi.dur	vi duration	number	10	The duration in seconds of the Vi market state
rnd.dur	vi random duration	number	1	The maximum duration is seconds that will be added to the vi durations above; the effective added duration will be a random number between 0 and rnd.dur
prcup.co	dynamic tunnel price up during Open	number	0	The price up leg of the dynamic tunnel during Open
prcdn.co	dynamic tunnel price down during Open	number	0	The price down leg of the dynamic tunnel during Open
prcup.po	dynamic tunnel price up during PreOpen	number	0	The price up leg of the dynamic tunnel during PreOpen
prcdn.po	dynamic tunnel price down during PreOpen	number	0	The price down leg of the dynamic tunnel during PreOpen
prcup.pc	dynamic tunnel price up during PreClose	number	0	The price up leg of the dynamic tunnel during PreClose
prcdn.pc	dynamic tunnel price down during PreClose	number	0	The price down leg of the dynamic tunnel during PreClose
prcup.vi	dynamic tunnel price up during Vi	number	0	The price up leg of the dynamic tunnel during Vi
prcdn.vi	dynamic tunnel price down during Vi	number	0	The price down leg of the dynamic tunnel during Vi
prcup.st	static tunnel price up	number	0	The price up leg of the static tunnel
prcdn.st	static tunnel price down	number	0	The price down leg of the static tunnel
lastfx	the reference for the static tunnel	number	-	The reference price for the static tunnel; if missing the reference for the static tunnel will be the reference price

Code	Name	Values	Default	Description
lastdyn	the reference for the `Vi` dynamic tunnel	number	-	The reference price for the dynamic tunnel during `Vi`
minhval	Minimum value for iceberg orders	number	0	The value refers to the whole order including the hidden portion
minxval	Minimum value for cross orders	number	0	-
deal.tuty	deal tunnel type	hard/static/vwap	hard	The tunnel type applicable to deals: • hard - the usual tunnel computed out of `pricelimit`, `priceup` and `pricedown` parameters • static - the static tunnel used in volatility interruption computed from `lastfx`, `prcup.st` and `prcdn.st` params • vwap - volume weighted average spread given by the visible volumes of main order book of the underlying instrument
liq	liquidity class	LIQ/ILQ	ILQ	the ESMA liquidity indicator - it will be put on the main market of each instrument

Below is the list of common exchange level params.

Code	Name	Values	Default	Description
chs.EUR	settlement holidays for EUR	text		A semicolon separated list of settlement holidays for EUR in `yyyyMMdd` format
chs.wk.EUR	settlement weekends for EUR	y/n	y	y - whether to add or not add weekends to the holidays list

TradeTickerDto (trades info)

Field name	Description
sym	Symbol code
mkt	Market code
tim	Timestamp (millisecond precision)
prc	Price
siz	Accumulated volume
cnt	Number of trades

Accumulated volume and Number of trades are computed over the trades performed on the same symbol-market, at the same price and in the same second but with the following notes:

- for a trade to be in the same record as the previous trade it must have the same symbol, market, price and `trade time` belonging to the same second
- `tim` is the last `trade time` of the group of trades represented by one record
- the records will be sorted chronologically
- (sym, mkt, prc, tim) tuples are not unique

IdxValueDto (index info)

Field name	Description
idx	Index code
uti	Timestamp
cls	Index value

Usually index values are computed, stored and transmitted only if the last value is different than the previous.

UlyDto (underlying symbol properties)

Field name	Description
cls	Close value
eti	Effective time
isi	ISIN
name	Underlying symbol name
sym	Underlying symbol code

TickersPack Structure

TickersPack is always embedded in an incoming message of type 800. The role of this structure is to

provide in real time information about the trading activity and changes regarding exchange entities properties or summaries or about structural changes of the exchange (entities are added/removed, etc).

Field name	Description
partial	Partial flag
xll	Exchange entity properties or entity summary changes as a list of PublicTickersPack objects
cmn	Level1 market data as a CommonTickersPack object
act	Level2 market data as an ActionTickersPack object

CommonTickersPack (level1 market data)

This structure contains a list of label-value items each representing a field of symbol-market summary that was changed and the new value for the field. The client should overwrite the old value of the field with the new value.

Field name	Description
sym	Symbol code
mkt	Market code
tck	A list of CommonTickersMsg objects

CommonTickerMsg

Field name	Description
label	A number that identifies the field of symbol-market summary that was changed (target)
value	The new value of the target. This field is a String so the value has to be converted to the type of the target before applying it over the old target value.

The next table provides a map between the label and the target it represents (see SmsInfo).

Label	Corresponding field from symbol-market summary (target)	Target description
907	trd	Number of trades at symbol-market level
917	bbp	Best buy price
918	bbv	Best buy volume
919	bsp	Best sell price

Label	Corresponding field from symbol-market summary (target)	Target description
920	bsv	Best sell volume
903	sym	Symbol code
904	mkt	Market code
905	vol	Trading volume
906	val	Trading value
908	opn	Open price
909	cls	Close price
910	avg	Average price
911	low	Low
912	hig	High
922	vad	Clearing value
923	clv	Last trade volume
925	omp	Potential fixing price
926	omv	Potential fixing volume
927	*	Fixing depth on sell side1
928	*	Fixing depth on buy side1
990	*	Last trade price2
991	*	Default settlement flag 0=the trade has the default settlement term (in this case Last trade price is the same as close price) `yyyyMMdd`=the trade has a settlement term other then the default settlement term
924	*	Number of trades executed at last price (multiple trades)
929	*	Accumulated volume of the trades (multiple trades)
921	dir	Direction of last trade
900	uti	Update time (`yyyyMMddHHmmssSSSS`)

Label	Corresponding field from symbol-market summary (target)	Target description
901	*	Liquidity indicator of the last trade 0=Unknown 1=Aggressive Buy 2=Aggressive Sell 3=Route Out 4=Auction 5=Cross
992	*	Refresh summary statistics flag 1=a refresh was issued by market control

* These fields do not have a direct correspondent to the symbol-market summary

NOTE

- 927 and 928 labels are related to level2 market data (see MboDto's `ixs` and `ixb` fields)
- last trade price can be different from close price if the trades have a different settlement term than the default one; close price for a symbol-market is the price of the last trade with a default settlement term

PublicTickersPack

Field name	Description
type	Identifies the role of the `tck` field. In case this field is 0 (zero) the `tck` field will be a list of ExchangeExplorerDto objects representing structural changes at exchange level entities or exchange entity properties changes. In case this field is 1,2,3,4 or 5, the `tck` field will contain a list of PublicTickerMsg objects representing changes of the trading summary at different levels: 1=Exchange level statistics 2=Market level statistics 3=Index level statistics 4=Symbol level statistics 5=Symbol-type level statistics
id	The code of the entity. The actual meaning of this field is different depending on the value of the `type` field. 0 - N.A. 1 - Exchange code 2 - Market code 3 - Index code 4 - Symbol code 5 - Symbol-type code
tck	In case `type` field is zero `tck` will be a list of ExchangeExplorerDto otherwise a list of PublicTickerMsg.

PublicTickerMsg

Field name	Description
label	A number that identifies the field of the exchange entity summary that was changed (target)
value	The new value of the target. This field is a String so the value has to be converted to the type of the target before applying it over the old target value.

PublicTickerMsg at exchange level (type =1)

Label	Exchange summary field (ExsInfo)	Description of Value
930	uti	Last Update time (yyyyMMddHHmmssSSS)
931	*	Exchange code
932	efd	Effective date (yyyyMMdd)
933	tva	Total value for spot markets (default currency)
934	tvo	Total volume for spot markets
935	ttr	Total trades for spot markets
936	tup	Number of up symbols for spot markets
937	tdw	Number of down symbols for spot markets
938	tst	Number of unchanged symbols for spot markets
817	fva	Total value for derivative markets
816	fvo	Total volume for derivative markets
815	ftr	Total trades for derivative markets
814	fup	Number of up symbols for derivative markets
813	fdw	Number of down symbols for derivative markets
812	fst	Number of unchanged symbols for derivative markets
811	foi	Open interest for derivative markets

* These fields do not have a direct correspondent in the symbol-market summary

PublicTickerMsg at Market level (type =2)

Label	Market summary field (MksInfo)	Description of Value
940	uti	Update time (yyyyMMddHHmmssSSS)
941	*	Market code
942	efd	Effective date (yyyyMMdd)
943	tva	Total value for this market (system's default currency)

944	tvo	Total volume for this market
945	ttr	Total trades for this market
946	tup	Number of up symbols for this markets
947	tdw	Number of down symbols for this markets
948	tst	Number of unchanged symbols for this markets

* These fields do not have a direct correspondent in the market summary

PublicTickerMsg at Symbol level (type =4)

Label	Symbol summary field (SysInfo)	Description of Value
987	crf	Current reference price
801	foi	Open Interest

PublicTickerMsg at Index level (type =3)

Label	Index summary field (IxsInfo)	Description of Value
960	uti	Update time (`yyyyMMddHHmmssSSS`)
961	efd	Effective date (`yyyyMMdd`)
962	*	Index code
963	opn	Open (previous close)
964	cls	Close(last)
965	low	Low
966	hig	High
967	cla	Net change
968	clp	Percentage change

* These fields do not have a direct correspondent in the index summary

PublicTickerMsg at Symbol-Type (type =5)

Label	Symbol-type summary field (StsInfo)	Description of Value
950	uti	Update time (`yyyyMMddHHmmssSSS`)
952	efd	Effective date (`yyyyMMdd`)
953	tva	Total trading value for this symbol-type (see trading currency for this symbol-type)

Label	Symbol-type summary field (StsInfo)	Description of Value
954	tvo	Total volume for this symbol-type
955	ttr	Total trades for this symbol-type
956	tup	Number of up symbols for this symbol-type
957	tdw	Number of down symbols for this symbol-type
958	tst	Number of unchanged symbols for this symbol-type
959	tvd	Total value for clearing this symbol-type (see clearing currency for this symbol-type)
802	foi	Open interest

ActionTickersPack

This structure contains level 2 market data regarding a certain symbol-market. In fact is contains a list of operations that should be applied to the existing client side order book in order to rebuild the correct image or the order book as it is in the central system. Before interpreting this information the client should take a snapshot (once per session) of the order book as the base image.

Field name	Description
sym	symbol
mkt	market
tck	a list of ActionTickersMsg

ActionTickerMsg

Field name	Description
shm	Show member flag. If true the owner of the order should be public
action	Action type +1=Add at the specified position into the order book an order with the specified value 0=Update the content of the order book at specified position with the new value -1=Delete the order at the specified position from the order book

Field name	Description
position	Position in order book
book	Order book identifier 1=regular 2=special
side	Side (1=Buy, 2=Sell)
value	An OrdDto object that should be applied at the specified position in the order book as the action type says

In case the action specifies invalid positions the client should discard the order book it is keeping and request again an order book snapshot.

NOTE `yyyyMMddHHmmssSSS` formatted strings refer to literal representation of a timestamp but they do not have explicit time zone information inside. For those timestamps the time zone is implicit Europe/Bucharest.

Appendix A: Reject Codes

In case an order operation is rejected, the OrdDto.txt field contains the reject reason formatted as REJECT_CODE/REJECT_DESCRIPTION. The following table contains the reject codes for order management operations. Be aware that if reject records are retrieved with recovery reports the OrdDto.txt field has maximum 40 characters so the reject description may be incomplete.

Reject code	Reject description
1	order not found
2	order not permitted
3	deal not permitted
4	quote not permitted
5	unpriced order not permitted
6	market order not permitted
7	cross order type not permitted
8	invalid timestamp
9	compute market price error
10	invalid price step
11	price computing error
12	price is out of range
13	duplicate level price
14	only one of order quantity fields could be filled (siz/oty)

Reject code	Reject description
15	invalid order size
16	invalid order quantity
17	order value exceeds maximum value
18	order value is lower than minimum value
19	invalid order type
20	invalid price spread for quote
21	price balancing error for quote
30	contingent order not permitted
31	hidden order not permitted for contingent orders
32	special execution not permitted for contingent orders
33	invalid term and trigger type combination
34	unpriced order not permitted for contingent orders
35	invalid trigger price
36	invalid tgp/lastprice relation
37	price is out of best bid-ask spread
40	hidden order not permitted
41	invalid disclosed order size
42	special terms order not permitted
43	order volume must be multiple of disclosed volume
50	use only default settlement term
51	use only standard settlement term
52	invalid date in term field
53	please use only open-term for order
54	open order not permitted
55	only day orders permitted
56	cannot change account
57	operation not permitted on oddlot market
58	use only permitted settlement term
59	invalid settlement date
61	fill or kill order not permitted
62	fill conditions are not met
63	invalid symbol-market status for FOK order

Reject code	Reject description
64	<fill or kill> conditions are not met because cross account
65	wrong volume for this matching type
66	IOC order not permitted
67	IOC conditions are not met
70	short-selling not permitted
71	cross account
72	out of term
73	uptick rule restriction
74	order type mismatch
80	cannot activate non-contingent order
82	order book change is not permitted
83	VFO order is not permitted
84	VFA order is not permitted
85	VFC order is not permitted
86	Trigger type TAL order is not permitted
87	invalid trigger type TAL conditions
88	Cancel on disconnect/Bulk order cancellation
89	Maximum order book depth exceeded
90	Invalid usage of short code
91	Client code is mandatory
92	T settlement not permitted
93	Target deal user not connected
94	Deal value exceed maximum net value
95	Deal value lower than minimun value
96	Deal price is out of spread
97	deal not found
99	business error
9999	check access error

Appendix B: Bonds coupons syntax

The encoding syntax for a bond's coupons is as follows:

```
[days-convention](coupon-1)[coupon-2]...[coupon-n]
```

NOTE	`[]` represents optional elements, `()` represents mandatory elements, ⋯ represents repetable elements

where

- `days-convention` is a digit representing the convention for counting the number of days between two calendar dates and at the moment we can have only `0 - for Actual/Actual convention` and `1 - for 30E/360 convention`

  The encoding of the number of days convention into the coupons string was introduced as of Arena 3.0.4. The coupons string representation before this version is missing this first digit the default convention being Actual/Actual

- `coupon-x` represents a single coupon and has in turn the following syntax (elements are separated by a comma):

  ```
  c(defined),(principal-value),(ex-coupon-date),(date-end-coupon),(subcoupon-1)[subcoupon-2]...[subcoupon-m]
  ```

  where

  - `c` is the first character of a coupon signaling the beginning of it's definition
  - `defined` cand be 1 if the coupon is defined or 0 if it's not
  - `principal-value` represents the principal value of the current coupon
  - `ex-coupon-date` represents the calendar date of the coupon's ex-date and is encoded with `yyyyMMdd` format
  - `date-end-coupon` represents the calendar date of the coupon's end date and is encoded with `yyyyMMdd` format
  - `subcoupon-x` represents a subcoupon and has in turn the following syntax (elements are separated by a semicolon):

    ```
    s(days-in-year);(subcupon-end-date);(rate);
    ```

    where

    - `days-in-year` number of days in an year
    - `subcoupon-end-date` represents the calendar date of the subcoupon's end date and is encoded with `yyyyMMdd` format; if we have a single subcoupon the end date is equal with the coupon end date.
    - `rate` subcoupon interest rate per year

NOTE	A bond must have at least one coupon and a coupon will have at least one sub-coupon; in fact we never had coupons with more than one subcoupon