D1.5.1 External interface CS OTE

Formats of messages for new IM Gas market

Project number: 00074102

Document number: D1.5.1

Document version: C

Date of issue: 28.6.2018

CONTENT

List of pictures

History of changes

Date	*Subject*	*Revision*
27.11.2017	Final version	A
3.4.2018	Extended technical specification of communication	B
28.6.2018	Extended technical specification of communication	C

1. INTRODUCTION

The aim of this document is to provide description of new interface for IM Gas market through the AMQP server.

If external participants use OTE client´s application then it already contains this interface and communication. In case external participants request connection of new OTE IM Gas to their systems, then this document should provide description of necessary changes in the interface for implementation.

2. DESCRIPTION OF CHANGES IN EXTERNAL INTERFACES

By reason of ensuring of high throughput and quick distribution of messages from the IM/BalM markets, CS OTE expands by another platform supporting the AMQP protocol. At these markets automatic communication will be only performed through communication with the AMQP RabbitMQ server. In comparison with the current automatic communication solution a special setup/permission will not be required by OTE. The interface for AMQP RabbitMQ server will be available to all participants without client identification (identification through certificate)

Participant has to perform implementation of his client which will connect to the MQ server. Participant will use his client for sending of his requests and recieving responses and mass messages. It is possible to use the AMQP client library RabbitMQ – see web site of the product www.rabbitmq.com .

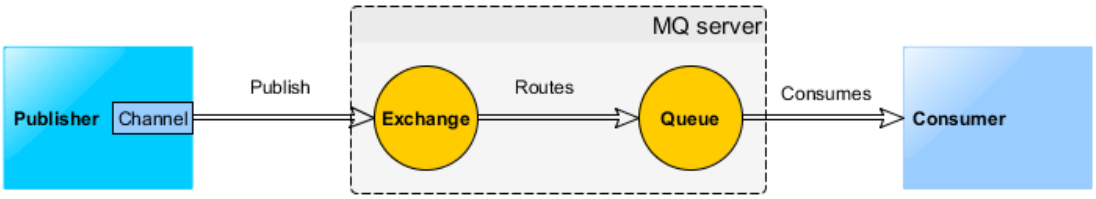
Process of establishing of communication and individual communication sripts are desribed in the following parts.

2.1. Communication protocol

Communication with the MQ server runs through the AMQP protocol (Advanced Message Queuing Protocol). It is open standard for communication layer of applications working on data exchange through messages. Implementation will be performed through the MQ server RabbitMQ, version 3.6.x .

AMQP standard defines basic entities:

- Exchange – input point for message receipt
- Routes – routing (distribution) of message
- Queue – output queue of messages



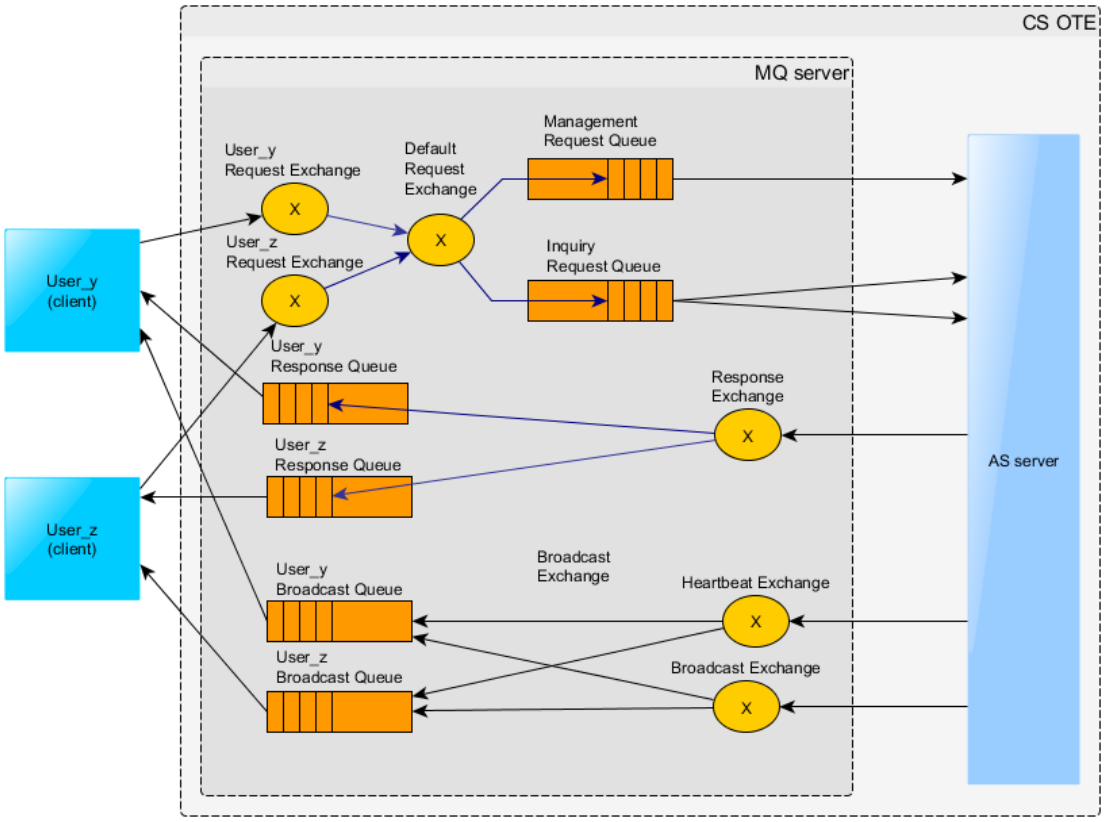
Picture 1 – Communication with MQ server

2.2. Connection to the MQ server

The following technical information is provided to external participant for connection: RabbitMQ server address, port and virtual host identification (see documentation for connection to the RabbitMQ http://www.rabbitmq.com/api-guide.html). External participants provides it's client certificate to OTE.

The first step is to establish connection „connection" to MQ server. Client´s certificate is necessary for creation of „connection". This certificate has to be registered in OTE systems first.

Communication channels „channels" are created on the basis of this connection. These channels connect to the individual „queue" which serve for mutual communication between client and server.



Picture 2 - Connection to the MQ server and architecture of message flow

2.3. Types of message exchange

For communication Client – MQ server are used two basic types of communication:

- Request-response (request – response) – requests initiated by client on which the MQ server will asynchronously respond. The response is sent only to initiator of the communication.
- Mass message (broadcast) – message distribution from the MQ server to clients. Distribution is performed on the basis of defined distributional rules and access rights.

2.3.1. Request-Response communication

Each user has on the RabbitMQ server his privat „Exchange“ with title „*market.exchanges.clientRequest.[USER_ID]“* which serves for request entry from client to the MQ server.Rights for writing into this specific exchange has only a given user.

The response queues used by the user for receiving responses upon requests is not created initially by the AMQP Server but from each client. Therefore at the start of communication the client creates one anonymous response queue with an auto generated name and uses this name within the *reply-to* field of all messages. The queue must be created with those parameters: durable=false, autoDelete = true, exclusive=true.

Types of requests:

- Instruction (Management request) – bid entry, modification, annulation
- Request (Inquiry request) – request for trading data

At request entry of the „Management request“ is immediately sent back to user response by the message „AckResp“ Table 8 – Message structure Acknowledgement Response (AckResp) (distributed into ResponseQueue). After request processing in the system the appropriate response for entered instruction is sent (distributed into BroadcatQueue). If the specific instruction causes change in trading data then mass message will be sent to all users, affected by the change, with appropriate content.

At request entry of the „Inquiry request“ type is sent to user response into his privat queue for responses (ResponseQueue).

2.3.2. Mass messages - Broadcast

System provides 2 basic types of mass messages

- Market data messages – messages about change in trading data and about change of market status. Messages are distributed to all logged in users who have requested permission for the given markets.
- Heartbeat messages – messages for verification of active connection with client.

For each user was created on the RabbitMQ server his private message queue with title „*market.broadcastQueue.[USER_ID]*“ to which is connected and from which user picking up messages. If user doesn´t continuously pick up messages, his queue can be overloaded and new messages will not be put in his queue. Due to this, there is a risk that user will not receive all market information.

2.3.3. Distribution rules

Description of distribution rules shows the following table. Some keys are defined dynamically according to the current market setup and user access rights.

Distribution key	Description
public	public information distributed to all users
public.<marketId>	public information on the given market which is distributed to all users who have access to given market
public.trade.<prodName>	public information on trade,distributed to all user who have access to a given product
PRTC_<particId>	relevant information for particular market participants
<prodName>	relevant information for product
<prodName>. PRTC_< particId>	relevant information only for PARTIC_ID in relation to product
trade	information on trades only for administrator (containing both sides of trade)
halfTrade.<prodName>. PRTC_<particId>	privat information on made trades (containing only half of a trade for a given participant)
USR_<userId>	privat information only for a given user

Table 1 – Summary of distribution rules

By way of illustration , there is the following example of the particular user shown.

User: “123“, Participant: “12“, Access to market: “IMG“, Available products: “Intraday gas“

User will receive messages which will be sent with some of the following distribution keys:

- public
- public.IMG
- public.trade.Intraday gas
- PRTC_12
- Intraday gas
- Intraday gas.PRTC_12
- halfTrade. Intraday gas.PRTC_12
- USR_123

2.3.4. Sequence counting for Broadcast Messages

Sequence number is used to identify the order of the broadcasts and to find out if some broadcasts have been lost. The sequence number is not part of the message payload but it is stored within the header of the AMQP message as an attribute „market-group-sequence“.

The sequence will be always increased by one for the next broadcast. It will be in-memory only (NOT persistent) which means that when the CS OTE system shuts down or terminates, the sequence will be reset to 0. Whenever the client gets a value which is not expected (i.e. value different than last_value+1) it should request the market data from the CS OTE system.

The sequence number is counted based on the routing keys (attribute „market-group-id“ in message header). So for each routing key there will be a different sequence number. All queues bound to the default broadcast exchange with the same routing key will receive the same sequence ID.

2.3.5. Invalid and Unroutable Requests

If the CS OTE system cannot process a request, because the request is incorrect or cannot be fulfilled, it will still send a negative response. The response message contains the details about the reasons why the request could not be processed.

If the CS OTE system cannot process the request because the XML schema version in the request message header is missing or invalid, the system sends a native error response. This response has set the attribute content-type with the value `market/error`. The body contains an error message encoded in UTF-8. Reasons for sending a native error message may be caused by validation errors detected by the CS OTE system. Validation errors may occur because of

- Invalid XML schema
- User ID not set
- ContentType not set
- ReplyTo not set
- CorrelationId not set

If the CS OTE system cannot process the request because it is down, the request message is discarded by the AMQP server and the client is notified about this action via its return listener.

2.3.6. Failover Processing

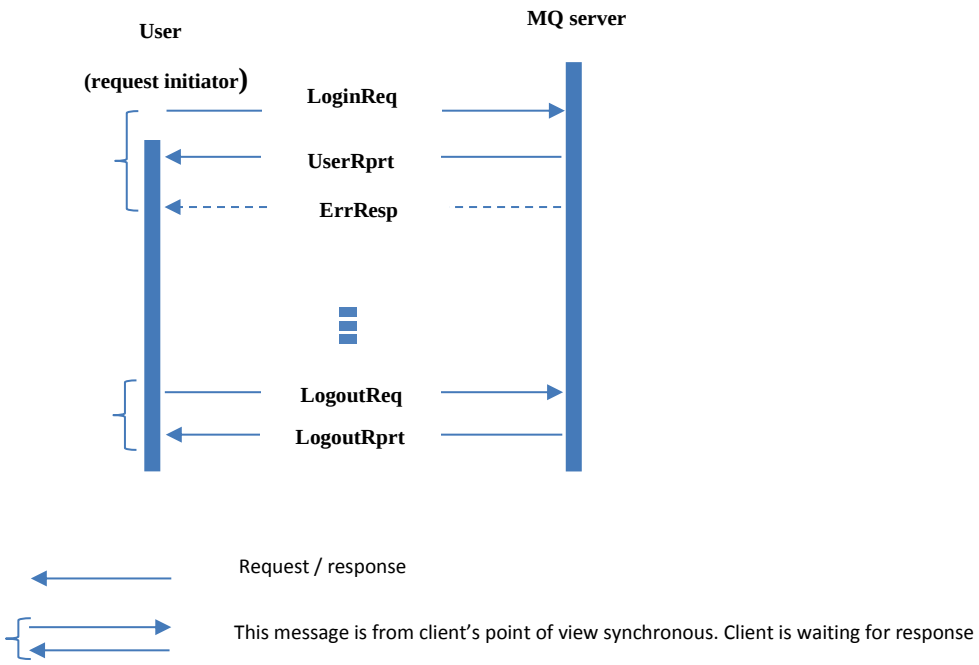
In case of AMQP server shutdown (due to failure or restart), the client subscriptions are lost. If the client has registered a shutdown listener, receives a shutdown notification from AMQP. After successful reconnect to the AMQP server, the client has to re-subscribe.

2.4. Communication scenario

2.4.1. User login, logout

Basic communication scenario for user login, logout to system and request for actual information about system. After establishing of connection with the MQ server user has to start data communication through login request *LoginReq* within 30s otherwise will be disconnected. At successful validation response is the message *UserRprt*, in case of failure message *ErrResp* is sent to client.

At termination of client´s application user is obliged to send logout message *LogoutReq*. If user doesn´t send a request for logout, then user is logged out according to the defined rules applied at loss of connection.



Picture 3 – Sequential scheme user login/logout

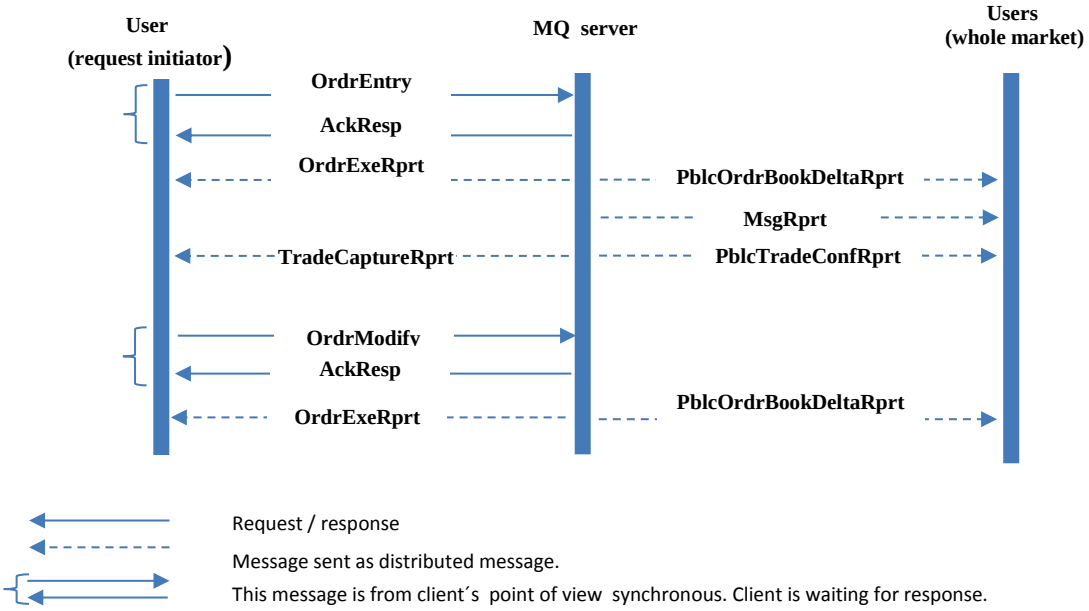
2.4.2. Work with bids

User enters bid by request *OrdrEntry* (alternatively bid modification by *OrdrModify*) and application server will response by *AckResp* that the request was successfully received or will response by *ErrResp* in case of wrong message definition. After bid processing server sends to client message about result of bid implementation/modification by *OrdrExeRprt*.

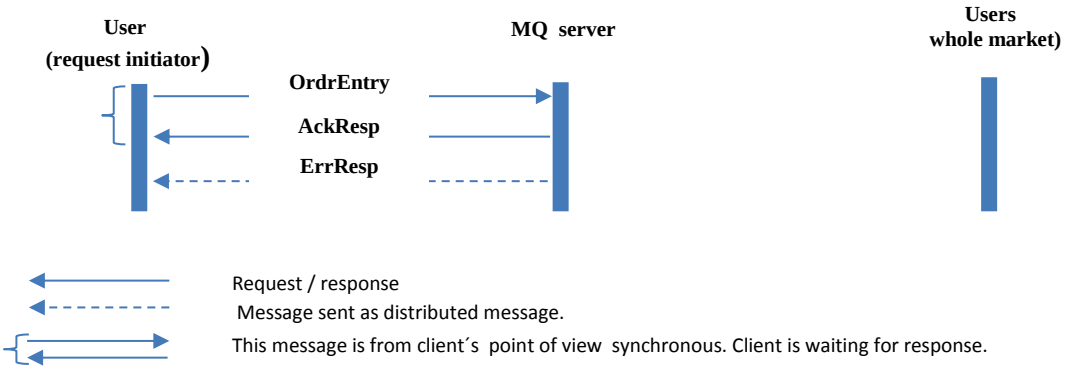
After that it is sent to all users the public message *PblcOrdrBookResp* which informs on notice board change, if bid implementation was successful.

In case a trade is made, the message *TradeCaptureRprt* is sent to bid owner and the public messages *MsgRprt* and *PblcTradeConfRprt* is sent to all user .When a trade is made the messages *OrdrExeRprt* and *TradeCaptureRprt* are sent to counter bid owner.

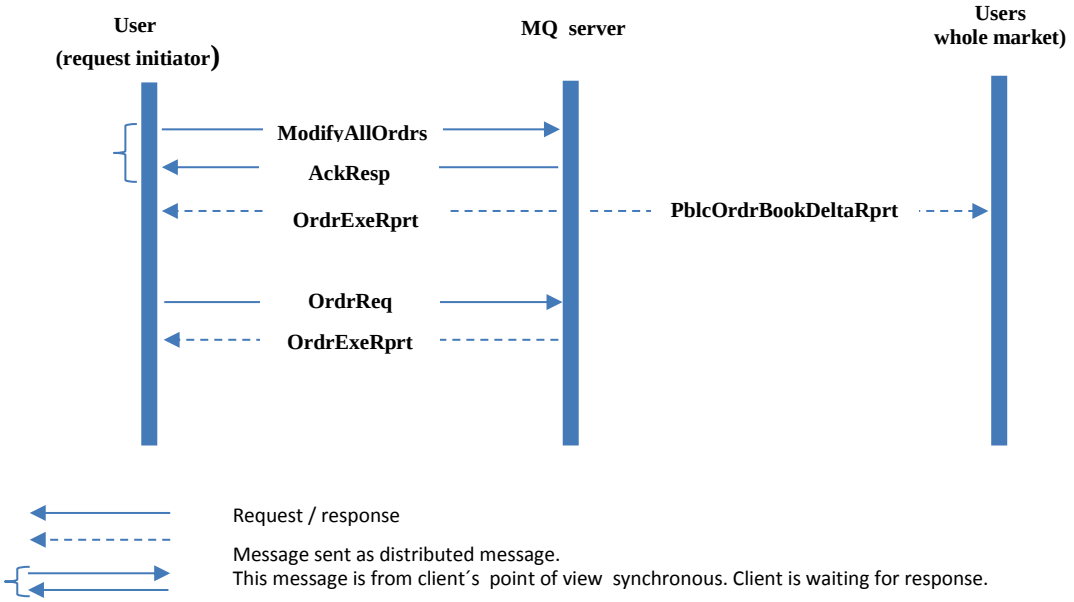
There is also possibility of request for bids through *OrdrReq* shown.



Picture 4 – Sequential scheme of bid entry with its trading and bid modification without trade creation



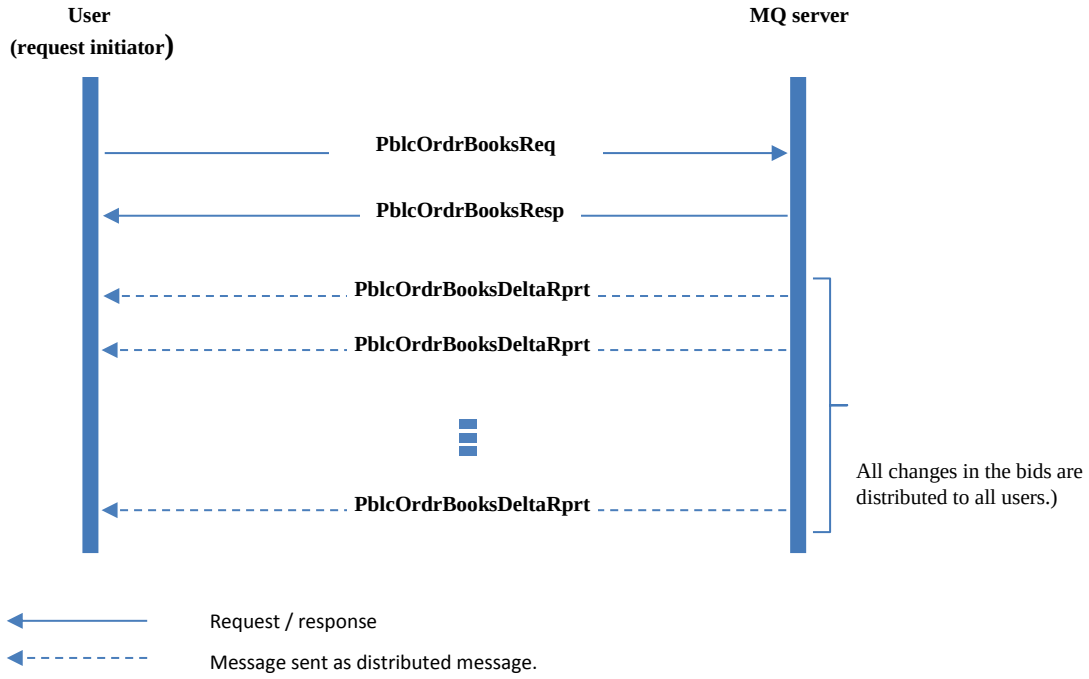
Picture 5 – Sequential scheme of unsuccessful bid entry



Picture 6 - Sequential scheme of mass bid modification (deactivation) and subsequent request for bids

2.4.3. Request for public data of bids

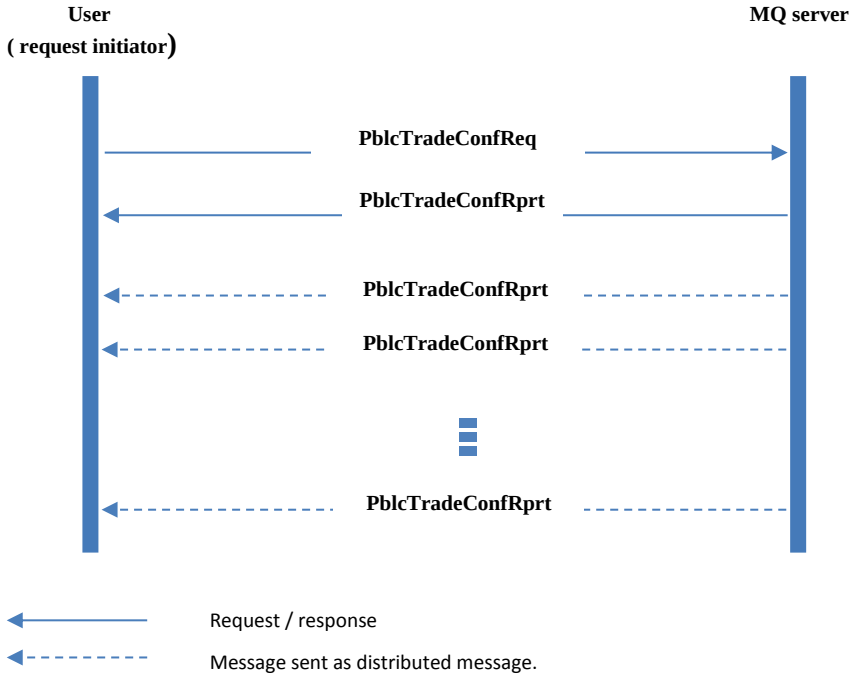
After login user sends one time request for list of active bids on the market through *PblcOrdrBooksReq* and server will respond by bid transcript *PblcOrdrBooksResp*. Thus client will receive full set of bids which are active in the system. If a new bid is entered or modified then the mass message *PblcOrdrBooksDeltaRprt* is sent.



Picture 7 - Sequential scheme of bid request proccesing

2.4.4. Request for public data of trades

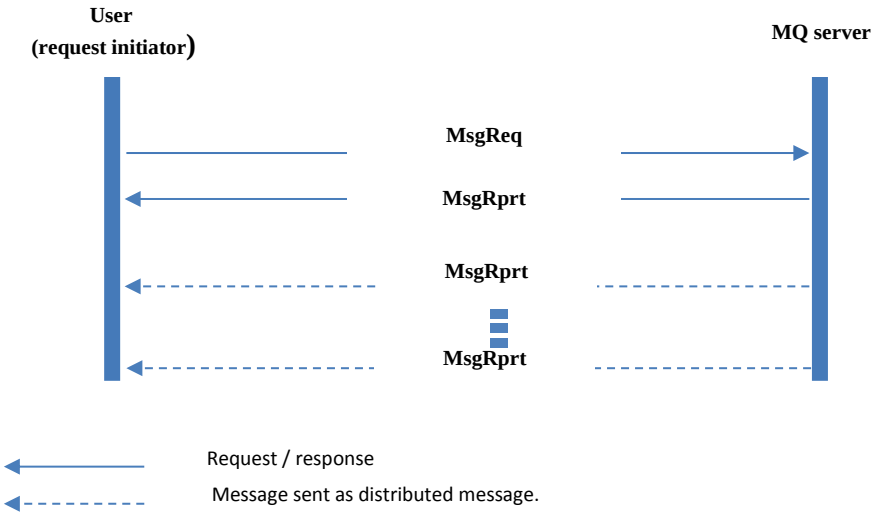
User will send request for trades made on the market by *PblcTradeConfReq* and server will respond by trade transcript *PblcTradeConfRprt.* In case of creation of another trade further messages are sent from the server.



Picture 8 - Sequential scheme of trade request processing

2.4.5. Request for informative messages

After successful login user sends query to the server with request *MsgReq* for list of messages. In the request user can specify whether he wants only privat or public messages. Then user receives given messages for the requested time *MsgRprt* and later distribution of new messages is sent to user automatically.

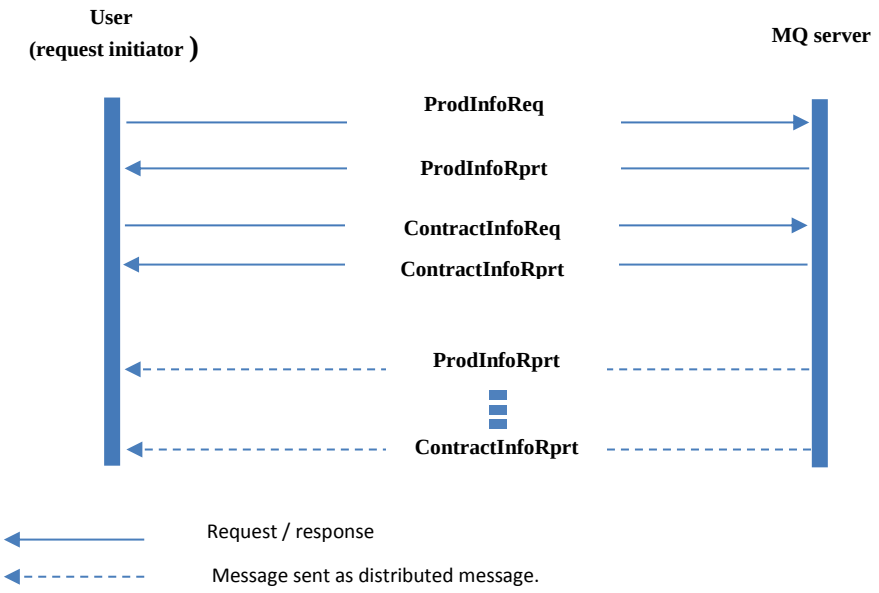


Picture 9 - Sequential scheme of market messages request processing

2.4.6. Request for Products and Contracts of the market

User can request the list of valid products by the request *ProdInfoReq* and response is sent by message *ProdInfoRprt*. In case of product change, update is sent by the message *ProdInfoRprt*.

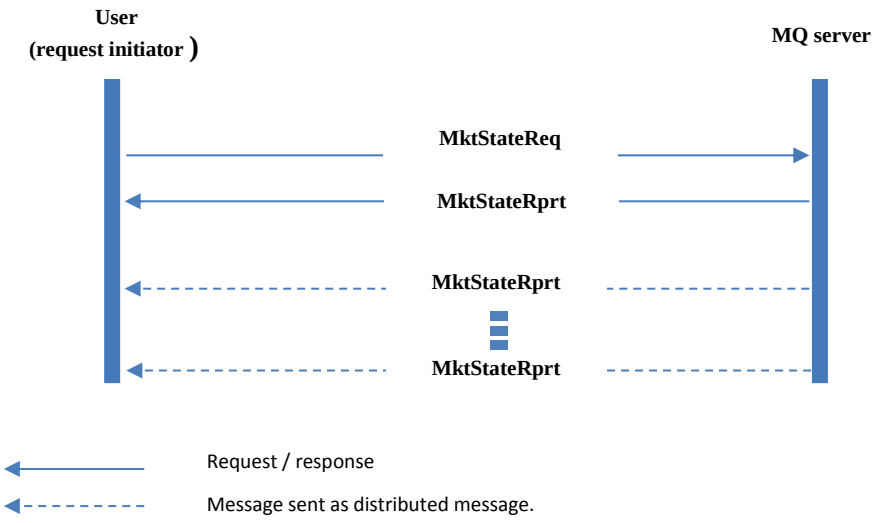
Similarly the messages relating to the Contracts are set.User can request list of valid contracts by the request *ContractInfoReq* and response is sent by the message *ContractInfoRprt*. In case of contract change, update is sent by the message *ContractInfoRprt*.



Picture 10 - Sequential scheme of Products and Contracts request processing

2.4.7. Request for market status

User can request information on the current market status by the request *MktStateReq* and response is sent by the message *MktStateRprt*. In case of change of market status, update is sent by the message *MktStateRprt*. These messages enable to find out current market status, if market is „Deactivated“ – trading is suspended.



Picture 11 - Sequential scheme of market status request processing

2.5. Communication messages

All messages sent between user and IM Gas application have own content of message which is defined through XML format. Description of the individual messages is stated in the following chapters.

2.5.1. General information

2.5.1.1. AMQP atributes

AMQP atributes used for communication between client and IM Gas application.

AMQP Message Attribut	Description
content-type	Contains information about the used XML payload version as well as the used message type. Valid content-type definitions are (version number has to be filled with the used version): ▪ market-gas/request; version=x (Used by the client when sending requests) ▪ market-gas/response; version=x ▪ market-gas/broadcast; version=x ▪ market-gas/heartbeat; version=x ▪ market-gas/error; version=x **Current version of messages is 1.**
reply-to	contains the queue name a response has to be sent to
user-id	contains the login-id of the logged in system
correlation-id	contains the request message id generated by client
expiration	contains an optional entry specifying if the request should be deleted if not executed within the specified time
contentEncoding	contains gzip, if messages are compressed (content is encrypted using gzip method); property is null if messages are not compressed. Message compressing can be activated per message type (e.g. OrdrExeRprt) .
market-group-sequence	Identify the order of the broadcasts counted for „market-group-id“. Only for broadcast message.
market-group-id	Identification of routing key belongs to attribute „market-group-sequence“. Only for broadcast message.
timestamp	Timestamp of distributed message fulfilled by RabbitMQ server. For more information you can see at https://www.rabbitmq.com/releases/rabbitmq-java-client/v3.6.1/rabbitmq-java-client-javadoc-3.6.1/com/rabbitmq/client/AMQP.BasicProperties.html#getTimestamp().

Table 2 – Messaage atributes according to the AMQP

2.5.1.2. XML convention

In message definition are used the following conventions

- Element tags are used for definition of data structure.
- Data is usually stated in atributes.
- Types:
  - Elements are highlighted in bold, but atributes are not highlighted in bold
  - **SE:** Structure Element. Data is not stated among tags but can contain atributes. (grey background in bold)

- **CE:** Content Element. Data is put among tags, but can also contain atributes (in bold).
- **A:** Element atributes.

Element and atribute order is not guaranteed and can be changed.

2.5.1.3. Values of volume in messages

Values of volume are stated in the all messages as integer. Own value is defined by group of atributes in the message ProdInfoRprt - *decShftQty*, *smallestTrdUnit* a *qtyUnit* (see chapter 2.5.4.13).

DecShftQty atribute determines position of the decimal point in the entered integer number (for example the value of volume 5200 with the atribute *decShftQty* = 3, means the value 5,200).

SmallestTrdUnit atribute ditermines the smallest step for volume entry (for example: *smallestTrdUnit* = 100 a *decShftQty* = 3 means that volume is possible to entry with the step 0,1).

QtyUnit atribute defines volume unit

2.5.1.4. Values of price in messages

Values relating to the prices are stated in all messages as integer. Own value is defined by group of atributes in the message ProdInfoRprt - *decShftPx*, *tickSize* and *currency* (see chapter 2.5.4.13).

DecShftPx atribute determines position of the decimal point in the entered integer number (for example: the value of volume 3624 with the atribute *decShftPx* = 2, means the value 36,24).

TickSize atribute determines the smallest step for price entry (for example: *tickSize* = 1 and *decShftPx* = 2 means that price is possible to entry with the step 0,01)

Currency atribute defines currency for trading.

2.5.1.5. Format of date items in messages

Date items are defined as "DateTime" types. Format of these items in XML messages is the following:

`YYYY-MM-DD`**`T`**`hh:mm:ss`**`Z`** `(2016-03-18T16:32:03Z)`

Symbol	Desription	Example
YYYY	Year	2016
MM	Month	03
DD	Day	18
T	Separation symbol of date and time section	T
hh	Hour (0-23 h)	16
mm	Minute	32
ss	Second	03
Z	Zero time zone =UTC time	Z

All dates and times are stated only in UTC.

2.5.1.6. Heartbeat

The heartbeat contains the text message with the attribute "server-timestamp" and „interval-length". Both attributes are in milliseconds, first one represents the difference between between the current time and midnight, January 1, 1970 UTC.

Message example: server-timestamp=1468251175238;interval-length=30000

2.5.1.7. Standard message header

Each message contains standard header with the following atributes.

XML Tag	Type	m/o	No.	Data Type	Short description
StandardHeader	SE	m		Structure	
marketID	A	m		Char(4)	Market Identification Code (MIC) of the market to which the request is sent or from which the request originates. The following values are allowed: "IMG": Intraday gas market.
clientData	SE	o	0..1	Structure	
clientDataInt	A	o		Integer	The client data fields in this section can be used by the client to store information or meta-data about a request. The content in these fields is not used by CS OTE system. Content is send back to client in response.
clientDataString	A	o		String	
clientCorrelationId	A	o		String	

Table 3 – Message header

2.5.1.8. Parameter description of the individual messages

In the next chapters are defined the following message parameters:

- Type – message type
  - Inquiry Request –request for data
  - Management request –executive instruction
  - Broadcast –mass message
- Role – message accessibility according to the role
- Routing key – message routing to the MQ server
- Message limit – max. number of messages of given title within defined time which will be processed for particular user by the server without being rejected. Definition of the formats a/b. „a“ represents max. number of messages received in one minute. „b“ represents max. number of messages received in one hour. If limit is not stated, then number of messages is not limited. Limit is counted separetely for each marketID.

2.5.2. General requests and responses

2.5.2.1. Login Request (LoginReq)

LoginReq	
Type:	Inquiry Request
Roles:	<All>
Routing Keys:	`market.request.inquiry`
Request Limits:	3/20

Request for login to the system. The system will respond by message „UserReport“.

XML Tag	Type	m/o	No.	Data Type	Short description
LoginReq	SE			Structure	
StandardHeader	*SE*	*m*		*Structure*	*Standard header of each message. Please see chapter2.5.1.7.*
user	A	m		String	Login ID of the user that want to login to the CS OTE system.
force	A	m		Boolean	Flag that indicates if this user want to force a login even if a user with the same credentials is already logged in into the CS OTE system.

disconnectAction	A	m		String	Action that will be executed in case of an unexpected connection loss between user and CS OTE system, irrespective of where the connection loss will be (user – AMQP – CS OTE system). The following values are allowed: "**NO**": No action is executed. "**DEACT_USER_ORDRS**": All orders of this user will be deactivated.

Table 4 – Message structure of the Login Request

2.5.2.2. User Report (UserRprt)

UserRprt	
Type:	Management Response, Broadcast
Response to:	LoginReq (sent to the user-generated private response queue or a broadcast to `market.broadcastQueue.<login-id>`)
Broadcasted:	Yes
Broadcast Routing Keys:	`USR_<login-id>`
Roles:	<All>

The message contains basic atributes of user. „User Report“ is sent back as response to „Login Request“ and it is also distributed at change of configuration regarding assigning of user to products.

XML Tag	Type	m/o	No.	Data Type	Short description
UserRprt	SE	m		Structure	
StandardHeader	*SE*	*m*		*Structure*	*Standard header of each message. Please see chapte 2.5.1.7.*
usrId	A	m		Integer	The unique identifier of a user.
sessionId	A	m		Long	The current session id of the user given after login to the system.
state	A	m		Char(4)	Current state of the User. The following values are allowed: "**ACTI**": User is active. It is possible to trade using this User. "**DELE**": User is deleted. Trading using this User is not possible. "**SUSP**": User is suspended. Trading using this User is not possible.
prtcId	A	m		Integer	The participant id the user belongs to.
name	A	m		String	Name of the user.
connectionLossMsg	A	o		String	In case of a connection loss for the previous user session, this field is filled with a connection loss message, indicating the connection loss event with date and time and the logout action executed by the CS OTE system.
Assgs	SE	m	1	Structure	
usrRole	CE	o	0..n	String	Contains the user roles assigned to the user
prdAssg	CE	o	0..n	String	Contains the products for the user
DlvryArea	SE	o	0..n	Structure	Delivery Area
dlvryAreaId	A	m		String	Delivery Area Id.
revisionNo	A	m		Long	Revision number. With every change of the delivery area this value is increased by one.
state	A	m		Char(4)	Current state of the delivery area. The following values are allowed: "**IACT**": Delivery area is inactive and thus not tradable. "**ACTI**": Delivery area is active. It is possible to trade in that area.
name	A	m		String	Name of the delivery area usually used for display purposes.
longName	A	m		String	Long name of the delivery area usually.
prodName	CE	o	0..n	String	List of assigned products to the delivery area.

Table 5 –Message structure of the User Report

2.5.2.3. Logout Request (LogoutReq)

LogoutReq	
Type:	Inquiry Request
Roles:	<All>
Routing Keys:	`market.request.inquiry`
Request Limits:	3/20

Request for user logout from the system

XML Tag	Type	m/o	No.	Data Type	Short description
LogoutReq	SE			Structure	
StandardHeader	*SE*	*m*		*Structure*	*Standard header of each message. Please see chapter 2.5.1.7.*
sessionId	A	m		Long	Session id of the PX session passed to the PX on login.

Table 6 –Message structure of the Logout Request

2.5.2.4. Logout Report (LogoutRprt)

LogoutRprt	
Type:	Inquiry Response, Broadcast
Response to:	LogoutReq (sent to the user-generated private response queue or a broadcast to `market.broadcastQueue.<login-id>`)
Broadcast:	Yes
Broadcast Routing Keys:	`USR_<login-id>`
Roles:	<All>

Message about user logout from system is sent as response to the request for logout ”Logout Request” or as a braodcast message as consequence of competitive login of the same user with forced login (force=true).

XML Tag	Type	m/o	No.	Data Type	Short description
LogoutRprt	SE			Structure	
StandardHeader	*SE*	*m*		*Structure*	*Standard header of each message. Please see chapter 2.5.1.7.*
sessionId	A	m		Long	Session id of the PX session passed to the PX on login.
usrId	A	m		Integer	User ID identification.
txt	A	o		String	Text field containing information about the reason of the logout.

Table 7 – Message structure of the Logout Report

2.5.2.5. Acknowledgement Response (AckResp)

AckResp	
Type:	Management Response
Response to:	OrdrEntry; OrderModify; ModifyAllOrdrs: (sent to the user-generated private response queue)
Broadcast:	No
Routing Keys:	---
Roles:	<All>

Message confirming receipt of instruction to processing

XML Tag	Type	m/o	No.	Data Type	Short description
AckResp	*SE*			Structure	
StandardHeader	*SE*	*m*		*Structure*	*Standard header of each message. Please see chapter 2.5.1.7.*

Table 8 – Message structure of the Acknowledgement Report

2.5.2.6. Error Response (ErrResp)

ErrResp	
Type:	Inquiry Response; Management Response; Broadcast
Response to:	<All> (sent to the user-generated private response queue or a broadcast to `market.broadcastQueue.<login-id>`)
Broadcast:	Yes

Broadcast Routing Keys:	USR_<login-id>
Roles:	<All>

Error message distributed in case of unsuccessful execution of instruction/ request

XML Tag	Type	m/o	No.	Data Type	Short description
ErrResp	SE			Structure	
StandardHeader	*SE*	*m*		*Structure*	*Standard header of each message. Please see chapter 2.5.1.7.*
Error	SE	m	1..n	Structure	
errCode	A	m		Integer	Predefined error codes. Some error messages do not have a specific error code. In this case the value is 0.
errEn	A	m		String	The error message for this error – English version.
errCz	A	m		String	The error message for this error – Czech version.
clOrdrId	A	o		Char(40)	Client order ID.

Table 9 – Message structure of the Error Report

2.5.3. Entry and management of bids

2.5.3.1. Order Entry (OrdrEntry)

OrdrEntry	
Type:	Management Request
Roles:	EmtasGImTsMod
Routing Keys:	market.request.management

Submission of one or more bids. Max. number of bids within one message is 25.

XML Tag		Type	m/o	No.	Data Type	Short description
OrdrEntry		SE	m	1	Structure	
StandardHeader		*SE*	m		*Structure*	*Standard header of each message. Please see chapter* ***Error! Reference source not found.***.
OrdrList		SE	m	1	Structure	List of all orders contained in the basket.
	Ordr	SE	m	1.. 25	Structure	
	state	A	o		Char(4)	**"ACTI"**: The order is entered and immediately exposed to the market for execution. This is the default value. **"HIBE"**: The order is entered into the CS OTE system but not exposed to the market.
	validityRes	A	o		Char(3)	Validity restriction of the order. If this field is omitted, the order will be treated as a "Good for Session" order. Valid values: **"GFS" (**Good for trading session): The order rests in the order book until it is either executed, removed by the user or the current trading session (trading phase) of the underlying contract ends. **"GTD"**. The order rests in the order book until the date specified in the validityDate field. **"NON"** (No validity restriction): Mandatory for orders with the execution restriction "FOK" or "IOC".
	validityDate	A	o		DateTime	This field is mandatory in case of validityRes equals "GTD". It is used to define the date until which the order is valid. The remaining part of the order will be removed from the order book after this point in time.
	txt	A	o		String	Comment entered by the user. Maximum possible length is 250 characters.
	type	A	m		Char(1)	Order type. Valid values: **"O":** Regular limit order (for all predefined contracts). **"I":** Iceberg order.

XML Tag		Type	m/o	No.	Data Type	Short description
	dlvryAreaId	A	m		String	Defines the delivery area of the order.Valid value is “CZ”.
	ordrExeRestriction	A	o		Char(3)	Execution restriction of the order. Valid values: **“NON”:** No restriction. This is the default. **“FOK”** (Fill or Kill): The order is immediately fully executed or deleted. **“IOC” (**Immediate and cancel): The order is executed immediately to its maximum extend. In case of a partial execution, the remaining volume is removed from the order book.
	qty	A	m		Integer	Contains the total quantity of the order. In case of an Iceberg order this field corresponds to the hidden quantity + display quantity.
	displayQty	A	o		Integer	Used to define display quantity of an Iceberg Order. This field is required only in the case of type=’I’.
	px	A	o		Long	Limit price of the order in currency defined by contracts. Value is multiplied by 100, e.g. 1 Euro = 100.
	ppd	A	o		Long	Peak price delta for Iceberg orders. • The ppd of buy orders must be smaller or equal than zero. • The ppd of sell orders must be greater or equal than zero. If it is omitted the system will assume a value of “0,00”.
	side	A	m		String	Defines on which side of the market the order is entered (“BUY”, “SELL”).
	contract	A	m		String	Contract code identifier. Applicable for orders for pre-defined contracts only.
	clOrdrId	A	o		String	Client Order Id with a maximum length of 40 characters.

Table 10 – Message structure of the Order Entry Message

2.5.3.2. Order Modify (OrdrModify)

OrdrModify	
Type:	Management Request
Roles:	EmtasGImTsMod
Routing Keys:	`market.request.management`

Message for modification of one or more bids. Max. number of bids within one message is 25.

XML Tag	Type	m/o	No.	Data Type	Short description
OrdrModify	SE	m	1	Structure	
StandardHeader	*SE*	*m*		*Structure*	*Standard header of each message. Please see chapter**Error! Reference source not found.**.*
ordrModType	A	m		Char(5)	Offers the possibility to activate, deactivate, modify or delete all orders contained in the basket. **"ACTI"**: Activate all orders contained in this basket. Already active orders are ignored. **"HIBE"**: Deactivates (hibernates) all orders contained in the basket. Hibernated orders are removed from the order book but are still available for modification or activation in the own orders list. **"MODI"**: Modifies all orders in the basket. **"DELE"**: Deletes all orders in the basket.
OrdrList	SE	m	1	Structure	List of all orders contained in the basket.
Ordr	SE	m	1..25	Structure	Definition of a single order.
validityRes	A	o		Char(3)	Validity restriction of the order. If this field is omitted, the order will be treated as a "Good for Session" order. Valid values: **"GFS" (**Good for trading session): The order rests in the order book until it is either executed, removed by the user or the current trading session (trading phase) of the underlying contract ends. **"GTD"** (Good till date): The order rests in the order book until the date specified in the validityDate field. **"NON"** (No validity restriction): Mandatory for orders with the execution restriction "FOK" or "IOC".
validityDate	A	o		DateTime	This field is mandatory in case of validityRes equals "GTD". It is used to define the date until which the order is valid. The remaining part of the order will be removed from the order book after this point in time.
type	A	m		Char(1)	Order type. It must be the same as order type of original order. Order type can't be changed by modification. Valid values: **"O":** Regular limit order (for all predefined contracts). **"I":** Iceberg order.
txt	A	o		String	Comment entered by the user. Maximum possible length is 250 characters.
ordrExeRestriction	A	o		Char(3)	Execution restriction of the order. Valid values: **"FOK"** (Fill or Kill): The order is immediately fully executed or deleted. **"IOC" (**Immediate and cancel): The order is executed immediately to its maximum extend. In case of a partial execution, the remaining volume is removed from the order book. **"NON":** No restriction. This is the default.
qty	A	m		Integer	Contains the total quantity of the order. In case of an Iceberg order this field corresponds to the hidden quantity + display quantity.
displayQty	A	o		Integer	Used to define display quantity of an Iceberg Order.
px	A	o		Long	Limit price of the order in currency defined by contract. Value is multiplied by 100, e.g. 1 Euro = 100.
ppd	A	o		Long	Peak price delta for Iceberg orders. • The ppd of buy orders must be smaller or equal than zero. • The ppd of sell orders must be greater or equal than zero. If it is omitted the system will assume a value of "0,00".
ordrId	A	m		Long	Order Id as returned by the CS OTE system. This value is used to identify the order to be modified.
revisionNo	A	m		Long	The latest revision number of the order must be provided by the user. In case the CS OTE has another revision number of currently valid order, it will reject the request with an ErrResp.
clOrdrId	A	o		String	Client Order Id with a maximum length of 40 characters.

Table 11 – Message structure of the Order Modify Message

2.5.3.3. Order Request (OrdrReq)

OrdrReq	
Type:	Inquiry Request

Roles:	EmtasGImTsAcc
Routing Keys:	`market.request.inquiry`
Request Limits:	1/10

Request for status of own bids

XML Tag	Type	m/o	No.	Data Type	Short description
OrdrReq	SE	m	1	Structure	
StandardHeader	SE	m		Structure	*Standard header of each message. Please see chapter2.5.1.7.*
contract	CE	o	0.. 1000	String	List of contract codes If no contract code is given, the own orders for all contracts assigned to the requesting user are returned.

Table 12 – Message request of the Order Request

2.5.3.4. Order Execution Report (OrdrExeRprt)

OrdrExeRprt	
Type:	Management Response; Broadcast
Response to:	OrdrEntry; OrdrModify; OrdrReq; ModifyAllOrdrs; (sent to the user-generated private response queue or a broadcast to `market.broadcastQueue.<login-id>`)
Broadcast:	Yes
Broadcast Routing Keys:	`<prodName>.<particId>`
Roles:	EmtasGImTsAcc

Message on the successful bid modification. The message is sent to market participants in the following cases:

- successful bid entry
- successful bid modification
- bid partially or totally traded
- as response of the request for bid (only in this case is sent to the privat queue for responses, in other case is sent to the queue for the mass messages)

XML Tag	Type	m/o	No.	Data Type	Short description
OrdrExeRprt	SE	m	1	Structure	
StandardHeader	*SE*	*m*		*Structure*	*Standard header of each message. Please see chapter* ***Error! Reference source not found.***.
OrdrList	SE	o	0..1	Structure	
Ordr	SE	o	0..n	Structure	
action	A	m		String	Code of the last action provided on the order. Valid values are: **"UADD"**: Order added by user. **"UHIB"**: Order hibernated by user. **"UMOD"**: Order modified by user. **"UDEL"**: Order deleted by user. **"SHIB"**: Order hibernated by the system. **"SMOD"**: Order modified by the system. **"SDEL"**: Order deleted by the system. **"FEXE"**: Order is fully executed. If an order comes into the system and gets executed immediately by matching an already existing order only one OrdrExeRprt for this order is sent with action FEXE or PEXE. If an order comes into the system and gets executed by a later entered order two messages are sent. One for the order entry with UADD and later one for the execution with either FEXE or PEXE. **"PEXE"**: Partial execution of order. **"IADD"**: A new slice of an Iceberg order was added to the service.
validityRes	A	o		Char(4)	Validity restriction of the order. If this field is omitted, the order will be treated as a "Good for Session" order. Valid values: **"GFS" (**Good for trading session): The order rests in the order book until it is

XML Tag	Type	m/o	No.	Data Type	Short description
					either executed, removed by the user or the current trading session (trading phase) of the underlying contract ends. **"GTD"** (Good till date): The order rests in the order book until the date specified in the vldtyDate field. **"NON"** (No validity restriction): Mandatory for orders with the execution restriction "FOK" or "IOC".
validityDate	A	o		DateTime	This field is mandatory in case of validityRes equals "GTD". It is used to define the date until which the order is valid. The remaining part of the order will be removed from the order book after this point in time.
timestmp	A	m		DateTime	Timestamp of the order entry as determined by the CS OTE system. This timestamp determines the execution priority in case of identical limit prices.
revisionNo	A	m		Long	This value is increased in case of a partial execution, hibernation, modification without execution priority change.
usrCode	A	m		String	User code of the user who entered the order.
state	A	m		Char(4)	The current state of the order in the system. Valid values: **"HIBE":** The order is entered into the XBID SOB system but not exposed to the market. **"ACTI":** The order is entered and immediately exposed to the market for execution **"IACT":** The order is inactive due time validity or fully executed. "**DELE**": The order is deleted
type	A	m		Char(1)	Order type. Valid values: **"O":** Regular limit order (for all predefined contracts). **"I":** Iceberg order.
dlrvyAreaId	A	m		String	Defines the delivery area of the order. Valid value is "CZ".
txt	A	o		String	Comment entered by the user. Maximum possible length is 250 characters.
ordrExeRestriction	A	o		Char(3)	Execution restriction of the order. Valid values: **"FOK"** (Fill or Kill): The order is immediately fully executed or deleted. **"IOC" (**Immediate and cancel): The order is executed immediately to its maximum extend. In case of a partial execution, the remaining volume is removed from the order book. **"NON":** No restriction.
totalQty	A	m		Integer	The total quantity entered with this order. If the order is partially matched, the totalQty still contains the original quantity value.
qty	A	m		Integer	Contains the quantity exposed to the market. In case of an Iceberg Order this is the rest of the display quantity.
hiddenQty	A	o		Integer	Contains the hidden quantity of the Iceberg order. The total executable quantity may be calculated by adding the hiddenQty to the qty.
displayQty	A	o		Integer	Used to define display quantity of an Iceberg Order.
px	A	o		Long	Limit price of the order in currency defined by contract. Value is multiplied by 100, e.g. 1 Euro = 100.
ppd	A	o		Long	Peak price delta for Iceberg orders.
side	A	m		String	Defines on which side of the market the order is entered. Valid values: **"BUY":** Buy order. **"SELL"**: Sell order.
contract	A	m		String	Contract code identifier.
ordrId	A	m		Long	Order Id as returned by the CS OTE system.
lastUpdateUsrCode	A	m		String	Information about the user who last updated the order
clOrdrId	A	o		String	Client Order Id with a maximum length of 40 characters. This value is not modified by the CS OTE system and may be used by Client applications to identify orders.

Table 13 – Message structure of the Order Execution Report

2.5.3.5. Modify All Orders (ModifyAllOrdrs)

ModifyAllOrdrs	
Type:	Management Request
Roles:	EmtasGImTsMod
Routing Keys:	`market.request.management`

Message for mass bid activation, deactivation and cancellation.

XML Tag	Type	m/o	No.	Data Type	Short description

ModifyAllOrders	SE			Structure		
StandardHeader	*SE*	*m*		*Structure*	*Standard header of each message. Please see chapter 2.5.1.7.*	
prtcId	A	o		String	Unique identifier of a partic.	One and only one of these attributes must be supplied.
usrId	A	o		Integer	Unique identifier of a user.	
ordrModType	A	m		Char(4)	Modification type for the orders: **"ACTI"**: Activate all orders. Already active orders are ignored. **"HIBE"**: Deactivates (hibernates) all orders. Hibernated orders are removed from the order book but are still available for modification or activation in the own orders list. " **"DELE"**: Deletes all orders.	
contract	CE	o	0.. 1000	String	List of contract codes If no contract code is given, the own orders for all contracts assigned to the specified participant or user are changed.	

Table 14 – Message structure of the Modify All Orders Message

2.5.4. Market Information

2.5.4.1. Public Order Books Request (PblcOrdrBooksReq)

PblcOrdrBooksReq	
Type:	Inquiry Request
Roles:	EmtasGImTsAcc
Routing Keys:	`market.request.inquiry`
Request Limits:	2/20

Request for notice board of the requested contract.

XML Tag	Type	m/o	No.	Data Type	Short description
PblcOrdrBooksReq	SE		1	Structure	
StandardHeader	*SE*	*m*		*Structure*	*Standard header of each message. Please see chapter 2.5.1.7.*
contractType	A	(m)		Char(3)	Defines which kind of contracts should be retrieved: Possible values are: "ALL" – All kind of contracts (pre-defined and user-defined) "PDC" – Only pre-defined contracts "UDC" – Only user-defined contracts This attribute is ignored when contractId is specified.
prodName	CE	(m)	0.. 1000	String	List of product names. All order books for these products are returned. Delivery area may be specified to filter the result. **Please note:** If no product name is given, at least one contract (see below) must be provided.
contract	CE	(m)	0.. 1000	String	List of contract codes. **Please note:** If no contract is given, at least one product name (see above) must be provided. If both values are given the contract is taken.
dlvryAreaId	CE	o	0.. 1000	String	Delivery areas for which the order book(s) should be retrieved.

Table 15 –Message structure of the Public Order Books Request

2.5.4.2. Public Order Books Response (PblcOrdrBooksResp)

PblcOrdrBooksResp	
Type:	Inquiry Response
Response to:	PblcOrdrBooksReq (sent to the user-generated private response queue)
Broadcast:	No
Broadcast Routing Keys:	`---`
Roles:	EmtasGImTsAcc

Public information on the current bids of given contract.

Message is distributed as response to the request „Public Order Book Request”.

XML Tag	Type	m/o	No.	Data Type	Short description
PblcOrdrBooksResp	SE	m	1	Structure	
StandardHeader	*SE*	*m*		*Structure*	*Standard header of each message. Please see chapter **Error! Reference source not found.**.*
OrdrbookList	SE	o	0..1		
OrdrBook	SE	o	0..n	Structure	
revisionNo	A	m		Long	This value is increased in case of any change in the order book. **Please note**: revision numbers of order book are stored in memory only (not persistent) on CS OTE system. After a restart of CS OTE system, the revision numbers of order books will start from 0 again.
contract	A	m		String	Contract code identifier.
dlvryAreaId	A	m		String	Delivery Area to which the attached order books refer to.
lastPx	A	o		Long	Last traded price.
pxDir	A	o		Integer	Defines the direction of the price movement with regard to the last 2 trades happened and that are relevant for this orderbook. Valid values are: -1: Price decreased 0: Price unchanged 1: Price increased
lastQty	A	o		Integer	Last traded quantity.
totalQty	A	o		Long	The total quantity traded during this trading session.
lastTradeTime	A	o		DateTime	Timestamp of the last execution.
highPx	A	o		Long	Highest traded price since the start of the trading period.
lowPx	A	o		Long	Lowest traded price since the start of the trading period.
SellOrdrList	SE	o	0..1	Structure	
OrdrBookEntry	SE	o	0..n	Structure	
ordrId	A	m		Long	Order Id as determined by the CS OTE system.
qty	A	m		Integer	The quantity of the order which is exposed in that delivery area.
px	A	m		Long	Limit price of the order in currency defined by contract. Value is multiplied by 100, e.g. 1 Euro = 100.
ordrEntryTime	A	m		DateTime	Timestamp of the order.
ordrType	A	o		Char(1)	**“O”:** Regular limit order. **“I”:** Iceberg order.
BuyOrdrList	SE	o	0..1	Structure	
OrdrBookEntry	SE	0	0..n	Structure	
ordrId	A	m		Long	Order Id as determined by the CS OTE system.
qty	A	m		Integer	The quantity of the order which is exposed in that delivery area.
px	A	m		Long	Limit price of the order in currency defined by contract. Value is multiplied by 100, e.g. 1 Euro = 100.
ordrEntryTime	A	m		DateTime	Timestamp of the order.
ordrType	A	o		Char(1)	**“O”:** Regular limit order. **“I”:** Iceberg order.

Table 16 – Message structure of the Public Order Books Report

2.5.4.3. Public Order Books Delta Report (PblcOrdrBooksDeltaRprt)

PblcOrdrBooksDeltaRprt	
Type:	Broadcast
Response to:	n/a
Broadcast:	Yes
Broadcast Routing Keys:	`<prodName>`
Roles:	EmtasGImTsAcc

The message Public Order Book Delta Report is sent at bid entry or at change of the active bid. The message contains all changed bids from the moment when was distributed the previous message *PblcOrdrBooksDeltaRprt* for the given contract.

The message format is the same with the message *PblcOrdrBooksResp*.

2.5.4.4. Message Request (MsgReq)

MsgReq	
Type:	Inquiry Request
Roles:	<ALL>
Routing Keys:	`market.request.inquiry`
Request Limits:	1/10

Request for messages of the trading system which were created by the trading system in the past. It is possible to enquire for the messages created in the past 2 days.

XML Tag	Type	m/o	No.	Data Type	Short description
MsgReq	SE		1	Structure	
StandardHeader	*SE*	*m*		*Structure*	*Standard header of each message. Please see chapter 2.5.1.7.*
type	A	m		Char(7)	Defines what kinds of messages are returned, allowing filtering the messages on a request level. Valid Values: **"ALL":** Return all messages. **"PUBLIC":** Return only public messages. **"PRIVATE":** Return only private messages.
endDate	A	m		DateTime	Timestamp defining to which point in time the messages should be retrieved.
startDate	A	m		DateTime	Timestamp defining from which point in time the messages should be retrieved. It is possible only to retrieve messages from the last 2 days.

Table 17 – Message structure of the Message Request

2.5.4.5. Message Report (MsgRprt)

MsgRprt	
Type:	Inquiry Response, Broadcast
Response to:	MsgReq (sent to the user-generated private response queue or a broadcast to `market.broadcastQueue.<login-id>`)
Broadcasted:	Yes

Broadcast Routing Keys:	`PRTC_<particId>`
	`<prodName>`
	`<prodName>.PRTC_<particId>`
	`public`

Roles:	`<All>`

Messages from the trading system are sent as response of the request for messages "Message Request" and then are distributed at creation of a new message in the trading system.

XML Tag		Type	m/o	No.	Data Type	Short description
MsgRprt		SE	m	1	Structure	
StandardHeader		*SE*	*m*		*Structure*	*Standard header of each message. Please see chapter 2.5.1.7.*
MsgList		SE	o	0..1		
	Msg	SE	o	0..n		
	msgId	A	m		Integer	The message Id as assigned by the CS OTE system.
	type	A	m			Defines the message type. Valid Values: **"PUBLIC":** The message is a public message. **"PRIVATE":** The message is a private message.
	contract	A	o		String	Underlying contract.
	timestmp	A	m		DateTime	Timestamp of the message as assigned by theCS OTE system.
	svrty	A	m		String	Severity of the message: **"URG":** Urgent message. **"ERR":** Error. **"HIG":** High prioritized message. **"MED":** Medium prioritized message. **"LOW":** Low priority message.

	mrktSupervisionMsg	A	m		Boolean	Determines if the message has been send by market supervision
	txtEn	A	m		String	Message text. – English version.
	txtCz	A	m		String	Message text – Czech version.
	sellDlvryAreaId	A	o		String	In case of an order execution, this field contains the delivery area of the sell side.
	buyDlvryAreaId	A	o		String	In case of an order execution, this field contains the delivery area of the buy side.

Table 18 –Message structure of the Message Report

2.5.4.6. Trade Capture Request (TradeCaptureReq)

TradeCaptureReq	
Type:	Inquiry Request
Roles:	EmtasGImTsAcc
Routing Keys:	`market.request.inquiry`
Request Limits:	7/35

The request for own trades. It is possible to enquire max. up to 7 days retroactively with the max. date time interval 48 hours (the values of limitation can be adjusted by the system). In case of incorrect entry parameters the response „ErrResp“ is sent back.

XML Tag	Type	m/o	No.	Data Type	Short description
TradeCaptureReq	SE				
StandardHeader	*SE*	*m*		*Structure*	*Standard header of each message. Please see chapter 2.5.1.7.*
startDate	A	m		DateTime	Start of the period for which the trades are retrieved. This value must fulfil the following conditions: • endDate – startDate <= 48 hours
endDate	A	o		DateTime	End of the period for which the trades are retrieved. The following condition must be fulfilled: • endDate – startDate <= 48 hours If no end date is given, the CS OTE system will return all trades until midnight of the start date. In case of invalid value Error Message is returned stating that diff is bigger than max value.

Table 19 – Message structure of the Trade Capture Request

2.5.4.7. Trade Capture Report (TradeCaptureRprt)

TradeCaptureRprt	
Type:	Inquiry Response, Broadcast
Response to:	TradeCaptureReq (sent to the user-generated private response queue or a broadcast to `market.broadcastQueue.<login-id>`)
Broadcasted:	Yes
Broadcast Routing Keys:	`halftrade.<prodName>.PRTC_<particId>`
Roles:	EmtasGImTsAcc

Message on trade creation is sent to the both participants of the given trade and for each of them is fulfilled only the part of the trade which relates to them. The message is also sent as response to „Trade Capture Request“.

XML Tag	Type	m/o	No.	Data Type	Short description
TradeCaptureRprt	SE	m	1	Structure	
StandardHeader	*SE*	*m*		*Structure*	*Standard header of each message. Please see chapter 2.5.1.7.*
TradeList	SE	o	0..1		
Trade	SE	o	0..n	Structure	
tradeId	A	m		Long	Trade ID of the trade.
state	A	m		Char(4)	Current state of the trade. Valid values are: **"CNCL":** Trade was cancelled by Central Admin. **"RREJ":** Requested Recall was rejected by Central Admin **"RGRA":** Requested Recall was granted by Central Admin. **"RREQ":** Recall of this trade was requested. **"ACTI":** Trade is active (this is the default value).
contract	A	m		String	Contract code
qty	A	m		Integer	Executed quantity.
px	A	m		Long	Execution price in currency defined by contract. Value is multiplied by 100, e.g. 1 Euro = 100.
execTime	A	m		DateTime	Execution date as assigned by the CS OTE system.
Buy	SE	o	0..1	Structure	
ordrId	A	m		Long	Order Id of the buy side order.
dlvryAreaId	A	m		String	Delivery Area to which the attached order books refer to.
prtcId	A	m		String	Participant who entered the buy side order.
usrCode	A	m		String	User code of the user who entered the buy side order.
clOrdrId	A	o		String	Client’s identification of order.
txt	A	o		String	Text of the buy side order.
Sell	SE	o	0..1	Structure	
ordrId	A	m		Long	Order Id of the sell side order.
dlvryAreaId	A	m		String	Delivery Area to which the attached order books refer to.
prtcId	A	m		String	Participant who entered the sell side order.
usrCode	A	m		String	User code of the user who entered the sell side order.
clOrdrId	A	o		String	Client’s identification of order.
txt	A	o		String	Text of the sell side order.

Table 20 – Message structure of the Trade Capture Report

2.5.4.8. Public Trade Confirmation Request (PblcTradeConfReq)

PblcTradeConfReq	
Type:	Inquiry Request
Roles:	EmtasGImTsAcc
Routing Keys:	`market.request.inquiry`
Request Limits:	7/35

Request for public information about created trades. It is possible to enquire max. up to 7 days retroactively with the max. date time interval 48 hours (the values of limitation can be adjusted by the system). In case of incorrect entry parameters the response „ErrResp“ is sent back.

XML Tag	Type	m/o	No.	Data Type	Short description
PblcTradeConfReq	SE	m		Structure	
StandardHeader	*SE*	*m*		*Structure*	*Standard header of each message. Please see chapter 2.5.1.7.*
startDate	A	m		DateTime	Start of the period for which the trades are retrieved. This value must fulfil the following conditions: endDate – startDate <= 48 hours
endDate	A	o		DateTime	End of the period for which the trades are retrieved. The following condition must be fulfilled: endDate – startDate <= 48 hours If no end date is given, the system will return all trades until midnight of the start date. In case of invalid value Error Message is returned stating that diff is bigger than max value.
prodName	CE	o	0.. 1000	String	Products for which the public trade confirmations are requested. If not supplied all products for which the user has access rights are returned

Table 21 – Message structure of Public Trade Confirmation Request

2.5.4.9. Public Trade Confirmation Report (PblcTradeConfRprt)

PblcTradeConfRprt	
Type:	Inquiry Response, Broadcast
Response to:	PblcTradeConfReq (sent to the user-generated private response queue or a broadcast to `market.broadcastQueue.<login-id>`)
Broadcasted:	Yes
Broadcast Routing Keys:	`public.trade.<prodName>`
Roles:	EmtasGImTsAcc

Message on trade creation. The message is distributed to all users who have assigned contract on which the trade was created. The message is also sent as response to „Public Trade Confirmation Request“.

XML Tag		Type	m/o	No.	Data Type	Short description
PblcTradeConfRprt		SE	m	1	Structure	
StandardHeader		*SE*	*m*		*Structure*	*Standard header of each message. Please see chapter 2.5.1.7.*
TradeList		SE	m	1	Structure	
	PblcTradeConf	SE	o	0..n	Structure	
	tradeId	A	m		Long	Trade Id of the underlying trade.
	state	A	m		Char(4)	Current state of the trade. Valid values are: **"ACTI":** Trade is active (this is the default value).
	contract	A	m		String	Contract code of the trade.
	px	A	m		Long	Execution price in currency defined by contract. Value is multiplied by 100, e.g. 1 Euro = 100.
	qty	A	m		Integer	Traded quantity.
	sellDlvryAreaId	A	m		String	Delivery area of the sell side. Valid value is “CZ”.
	buyDlvryAreaId	A	m		String	Delivery area of the buy side. Valid value is “CZ”.
	tradeExecTime	A	m		DateTime	Trade execution time.

Table 22 – Message structure of the Public Trade Confirmation Report

2.5.4.10. Contract Information Request (ContractInfoReq)

ContractInfoReq	
Type:	Inquiry Request
Roles:	EmtasGImTsAcc
Routing Keys:	`market.request.inquiry`
Request Limits:	2/20

Request for contract. It is possible to enquire max. up to 7 days retroactively. In case of incorrect entry parameters the response „ErrResp“ is sent back.

XML Tag	Type	m/o	No.	Data Type	Short description
ContractInfoReq	SE			Structure	
StandardHeader	*SE*	*m*		*Structure*	*Standard header of each message. Please see chapter 2.5.1.7.*
startDate	A	o		Date	Start date for which the contract information is requested. Notes: • if contract is specified this attribute is ignored • if prodName is specified or neither contract nor prodName are specified, this attribute becomes mandatory.
endDate	A	o		Date	End date for which the contract information is requested. Notes: • if contract is specified this attribute is ignored • if prodName is specified or neither contractId nor prodName are specified, this attribute becomes mandatory.
prodName	CE	o	0.. 1000	String	The contract information for all contratcs belonging to the given products is requested. If prodName is specified, the contract element cannot be specified and the startDate and endDate attributes are mandatory.
contract	CE	o	0..1	String	If contract is specified, the prodName element cannot be specified and the startDate and endDate attributes are ignored.

Table 23 – Message structure of the Contract Information Request

2.5.4.11. Contract Information Report (ContractInfoRprt)

ContractInfoRprt	
Type:	Inquiry Response, Broadcast
Response to:	ContractInfoReq (sent to the user-generated private response queue or a broadcast to `market.broadcastQueue.<login-id>`)
Broadcasted:	Yes
Routing Keys:	`<prodName>`
Roles:	EmtasGImTsAcc

Information on contracts. The message is distributed in case of the atribute change at contract or as response to the request “Contract Information Request”.

XML Tag	Type	m/o	No.	Data Type	Short description
ContractInfoRprt	SE	m	1	Structure	
StandardHeader	*SE*	*m*		*Structure*	*Standard header of each message. Please see chapter **Error! Reference source not found.**.*
ContractList	SE	o	0..1	Structure	
Contract	SE	o	0..n		
contract	A	m		Integer	Contract code
revisionNo	A	m		Long	Revision number of the contract.
prod	A	m		String	Underlying product.
prodRevisionNo	A	m		Long	Revision number of the underlying product.
name	A	m		String	Contract name. This is used for display purposes.
longName	A	m		String	Contract long name, containing additional information.
dlvryStart	A	m		DateTime	Start of delivery.

XML Tag	Type	m/o	No.	Data Type	Short description
dlvryEnd	A	m		DateTime	End of delivery.
duration	A	o		Double	A contract would have value 24 (or 23/25 in case of short/long clock change).
predefined	A	m		Boolean	Flag that indicates, if a contract has been automatically created by the system. 1 = automatically generated
state	A	m		String	Current state of the contract. The following values are allowed: **"HIBE":** Hibernated, the contract was manually deactivated by Central Admin. **"ISSUED":** The contract is issued, but not available for trading. **"OPEN":** Contract is active and available for trading. **"CLOSE"**: Contract is closed and not available for trading. **"TERM"**: Contract is terminated and not available for trading. **"NOT_ISSD":** The contract is not issued and there is not possible to trade on this contract at all.
tradingPhaseStart	A	m		DateTime	Start date and time of the current/next trading phase. When "NOT_ISSD" state is distributed then contains timestamp of the "Not issued" event.
tradingPhaseEnd	A	o		DateTime	End date and time of the current/next trading phase. When "NOT_ISSD" state is distributed then contains timestamp of the "Not issued" event.

Table 24 – Message structure of the Contract Information Report

2.5.4.12. Product Information Request (ProdInfoReq)

ProdInfoReq	
Type:	Inquiry Request
Roles:	EmtasGImTsAcc
Routing Keys:	`market.request.inquiry`
Request Limits:	2/20

Request for detailed information on products

XML Tag	Type	m/o	No.	Data Type	Short description
ProdInfoReq	SE	m	1	Structure	
StandardHeader	*SE*	*m*		*Structure*	*Standard header of each message. Please see chapter 2.5.1.7.*
prodName	CE	o	0.. 1000	String	

Table 25 – Message structure of the Product Information Request

2.5.4.13. Product Information Report (ProdInfoRprt)

ProdInfoResp	
Type:	Inquiry Response, Broadcast
Response to:	ProdInfoReq (sent to the user-generated private response queue or a broadcast to `market.broadcastQueue.<login-id>`)
Broadcasted:	Yes
Broadcast Routing Keys:	---
Roles:	EmtasGImTsAcc

Detailed information on product as response to "Product Information Request".

XML Tag	Type	m/o	No.	Data Type	Short description
ProdInfoRprt	SE	m	1	Structure	
StandardHeader	*SE*	*m*		*Structure*	*Standard header of each message. Please see chapter **Error! Reference source not found.**.*
ProdList	SE	o	0..1	Structure	
Prod	SE	o	0..n	Structure	

XML Tag	Type	m/o	No.	Data Type	Short description
prodName	A	m		String	Unique identifier name of the product.
dsplName	A	m		String	String used to display the product.
currency	A	m		Char(3)	The currency of the product (e.g. "EUR").
revisionNo	A	m		Long	Revision number of the product. This value is increased by one every time the product is modified by the system.
qtyUnit	A	m		String	Defines the quantity unit.
smallestTradableUnit	A	m		Integer	Defines the smallest tradable unit of the product.
minDsplQty	A	o		Integer	Minimal display quantity.
decShftQty	A	m		Integer	Decimal shift of the quantity information. A value of 2 results in a display of 100 Kw.
maxQty	A	m		Integer	Maximal allowed quantity for orders entered in contracts belonging to this product.
minPx	A	m		Long	Minimal price allowed for orders entered in contracts belonging to this product.
maxPx	A	m		Long	Maximal price allowed for orders entered in contracts belonging to this product.
decShftPx	A	m		Integer	Decimal shift of the price information. A value of 2 results in a display in Eurocents.
tickSize	A	m		Integer	Defines the minimum increment for limit prices for this product. The value is entered as an integer, but the decimal price shift is applied.
contractNamePattern	A	o		String	Format string for the contract name.
ProdCfgs	SE	o	0..n	Structure	
cfgKey	A	m		String	Exchange specific product attribute name.
cfgVal	A	m		String	Exchange specific product attribute value.

Table 26 – Message structure of the Product Information Report

2.5.4.14. Market State Request (MktStateReq)

MktStateReq	
Type:	Inquiry Request
Roles:	EmtasGImTsAcc
Routing Keys:	`market.request.inquiry`
Request Limits:	1/10

Request for the current market status. The reqeusted market is specified in the header of message "StandardHeader"

XML Tag	Type	m/o	No.	Data Type	Short description
MktStateReq	SE	m	1	Structure	
StandardHeader	*SE*			*Structure*	*Standard header of each message. Please see chapter 2.5.1.7.*

Table 27 – Message structure of the Market State Request

2.5.4.15. Market State Report (MktStateRprt)

MktStateRprt	
Type:	Inquiry Response, Broadcast
Response to:	MktStateReq (sent to the user generated private response queue or a broadcast to `market.broadcastQueue.<login-id>`)
Broadcasted:	Yes
Broadcast Routing Keys:	`public.<marketId>`
Roles:	EmtasGImTsAcc

Current information on about trading status on market. The message is distributed in case of change of the market status and then as response to the request "Market State Request".

XML Tag	Type	m/o	No.	Data Type	Short description
MktStateRprt	SE	m	1	Structure	

StandardHeader	*SE*	*m*		*Structure*	*Standard header of each message. Please see chapter 2.5.1.7.*
state	A	m		Char(4)	Contains the current market state. The following values are allowed: **"HIBE":** Hibernated; no trading is possible and order books are empty. Done on WebGui by Admin. **"ACTI":** Market is active and trading is possible.
revisionNo	A	m		Long	Revision number of the market. With every change of the market state this value is increased by one.

Table 28 – Message structure of the Market State Report

2.5.4.16. Last Trade Price Request (LastTradePriceReq)

LastTradePriceReq	
Type:	Inquiry Request
Roles:	NominationTransport, NominationStorage
Routing Keys:	`market.request.inquiry`
Request Limits:	4/20

Dotaz na cenu posledního realizovaného obchodu daného kontraktu na VDP dle PTP. V případě chybných vstupních parametrů je vrácena odpověď „ErrResp“.

XML Tag	Type	m/o	No.	Data Type	Short description
LastTradePriceReq	SE	m		Structure	
StandardHeader	*SE*	*m*		*Structure*	*Standard header of each message. Please see chapter* ***Error! Reference source not found.***.
contract	A	m		String	Contract code for which the last known trade price is requested.

Table 29 – Message structure Last Known Trade Price Request

2.5.4.17. Last Trade Price Report (LastTradePriceRprt)

PblcTradeConfRprt	
Type:	Inquiry Response
Response to:	LastTradePriceReq (sent to the user-generated private response queue)
Broadcasted:	No
Broadcast Routing Keys:	
Roles:	NominationTransport, NominationStorage

Zpráva je odeslána jako odpověď na „Last Trade Price Request“.

XML Tag	Type	m/o	No.	Data Type	Short description
LastTradePriceRprt	SE	m	1	Structure	
StandardHeader	*SE*	*m*		*Structure*	*Standard header of each message. Please see chapter* ***Error! Reference source not found.***.
contract	A	m		String	Contract code of the trade.
tradeExecTime	A	m		DateTime	Trade execution time.
px	A	m		Long	Last known price in currency defined by contract. Value is multiplied by 100, e.g. 1 Euro = 100.

Table 30 – Message structure Last Known Trade Price Report

2.6. New scenarios for the current way of automatic communication through the communication server KSP/KSM

2.6.1. Setup/change/response to the new offline limit

Current status of offline limit, including other values, will return adjusted report of current status of the limits in the present structure SFVOTLIMITS.

The new structure SFVOTSETTINGS will serve for the offline limit setup through automatic communication (KSP). Except a standard header and receiver/sender indentification will contain:

SFVOTSETTINGS/Setting –main encasing data element

SFVOTSETTINGS/Limit –main element for limit setup

SFVOTSETTINGS/Limit@type –limit type, enumerative type, at present only at the IM/BaIM markets.

SFVOTSETTINGS/Limit@value –new value for the given limit in CZK

Example of the limit setup for 20 000 CZK:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<SFVOTSETTINGS answer-required="false" date-time="2015-06-24T12:41:08+02:00" dtd-release="1" dtd-version="1" id="123" message-code="475" xmlns="http://www.ote-cr.cz/schema/sfvot/settings">
    <SenderIdentification id="8591824000007" coding-scheme="14"/>
    <ReceiverIdentification id="8591824000007" coding-scheme="14"/>
    <Setting>
        <Limit type="VDT/VT" value="20000"/>
    </Setting>
</SFVOTSETTINGS>
```

The response will contain the structure RESPONSE with the message code 477 and in case of successful performing also data transcript in the form of current status of limits (SFVOTLIMITS with the message code 476). The current return codes from financial report area will be partially used.

RESPONSE/Reason@code	***Description***
S09000	Request successfully processed, set up changed
S09008	Participants doesn´t have necessary setting (undefined limits)
S09009	Participant doesn´t have a permission to a change
S09010	Lack of free financial resources
S09011	Invalid value.
S09012	Not expected error.

2.6.2. Message on transfer of part of the offline limit into online

At changing of part of the offline limit to online as stated in the chapter 8.2.2.2. Trade creation – all traded – online FS security for BKO side, utilization of instruction in offline < trade utilization in online for B, it will

be necessary to inform participant on this status through the automatic communication as well. Information sent to the participants will be the following:

- Moved financial amount from the IM limit into online (CZK)
- Remaining amount of the IM limit (CZK)
- Remaining free financial resources at the IM security (CZK)
- ID of a trade which caused this transfer
- Trade delivery date

For this purposes will serve the new structure SFVOTLIMITCHANGE. It will be sent in an unsolicited way through the KSP. Beside a standard header and receiver/sender identification will contain:

SFVOTLIMITCHANGE/Limits –main encasing data element

SFVOTLIMITCHANGE/Limits@trade-date –trade delivery date

SFVOTLIMITCHANGE/Limits@trade-id –trade id

SFVOTLIMITCHANGE/Limit –main limit element

SFVOTLIMITCHANGE/Limit@type –limit type, enumerative type, at present only at the IM/BaIM markets.

SFVOTLIMITCHANGE/Limit@value –new value for a given limit in CZK

SFVOTLIMITCHANGE/Limit@moved –Resources moved to another type in CZK (for IM/BaIM markets until the online utilization of short – term trades)

SFVOTLIMITCHANGE/Limit@free – free resources for a given limit in CZK

Example:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<SFVOTLIMITCHANGE answer-required="false" date-time="2015-06-24T12:41:08+02:00" dtd-release="1" dtd-version="1" id="123" message-code="478" xmlns="http://www.ote-cr.cz/schema/sfvot/limitchange">
    <SenderIdentification id="8591824000007" coding-scheme="14"/>
    <ReceiverIdentification id="8591824000007" coding-scheme="14"/>
    <Limits trade-id="237445" trade-date="2015-08-31">
        <Limit type="VDT/VT" value="15000" moved="5000" free="1280"/>
    </Limits>
</SFVOTLIMITCHANGE>
```

3. USE OF AN ELECTRONIC SIGNATURE

Messages are transfered between client´s application and backend system in the XML form and are by reason of integrity and incontestability ensuring secured by the electronic signature.

Electronic signature is inserted into following messages (see XSD templates in chapter 4)

- OrdrModify
- OrdrEntry
- ModifyAllOrdrs

Electronic signature is created in the form of the Enveloped XML signature (http://www.ietf.org/rfc/rfc3275.txt), i.e. the Signature element is inserted at the end of message under root element of XML message.

XML Signature **has to contain client´s certificate** either in the form of BinarySecurityToken (i.e. through a link in the SecurityTokenReference element) or has to be encrypted in the X509Data section. Other forms such as SKI are not supported.

3.1. Example of message using electronic signature

The message before signing

```
<?xml version="1.0" encoding="UTF-8"?>
<OrdrModify ordrModType="ACTI" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <StandardHeader marketID="IM"/>
  <OrdrList>
    <Ordr ordrId="0" qty="100" revisionNo="0" type="O"/>
  </OrdrList>
</OrdrModify>
```

will have after the XML signature application the following form

```
<?xml version="1.0" encoding="UTF-8"?>
<OrdrModify ordrModType="ACTI" xmlns:ds="http://www.w3.org/2000/09/xmldsig#"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
        <StandardHeader marketID="IM" />
        <OrdrList>
                <Ordr ordrId="0" qty="100" revisionNo="0" type="O" />
        </OrdrList>
        <ds:Signature xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
                <ds:SignedInfo>
                        <ds:CanonicalizationMethod
                                Algorithm="http://www.w3.org/TR/2001/REC-xml-c14n-20010315" />
                        <ds:SignatureMethod Algorithm="http://www.w3.org/2000/09/xmldsig#rsa-sha1" />
                        <ds:Reference URI="">
                                <ds:Transforms>
                                        <ds:Transform Algorithm="http://www.w3.org/2000/09/xmldsig#enveloped-signature" />
                                </ds:Transforms>
                                <ds:DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1" />
                                <ds:DigestValue>vx6g0IKv5Qw1nwqOM4hGmn5igXY=</ds:DigestValue>
                        </ds:Reference>
                </ds:SignedInfo>
        <ds:SignatureValue>UJUfISXST2D9FNBah…</ds:SignatureValue>
                <ds:KeyInfo>
                        <ds:X509Data>
                                <ds:X509Certificate>MIIElTCCA32gAwIBAgIDIA+.....</ds:X509Certificate>
                        </ds:X509Data>
                        <ds:KeyValue>
                                <ds:RSAKeyValue>
                                        <ds:Modulus>xnm5U6RIswp0aRV9ab…</ds:Modulus>
                                        <ds:Exponent>AQAB</ds:Exponent>
                                </ds:RSAKeyValue>
                        </ds:KeyValue>
                </ds:KeyInfo>
        </ds:Signature>
</OrdrModify>
```

4. XSD TEMPLATES

The XSD templates are shown in the attached file: XSD_IMG.zip