Information System
User Manual

External interface CS OTE
BINARY API message format of electricity Intraday market

Document n.: D1.4.X
Doc. version.: A1
Publication date: 12.12.2025

Document name:
D1.4.X_Messages_format_Binary_API_OTE-COM_ELE_A_EN.docx

Content

List of figures

List of tables

History of changes

Date	Version	Change description
12.12.2025	A1	Document creation.

Reference documents

[1] OTE-COM test environment access configuration manual

[2] OTE-COM production environment access configuration manual

[3] Změna_formátu_zpráv_OTE-COM_ELE_protobuf_vs_XML

[4] .PROTO definition [1]

[1] The definition file .proto will be available in a later project phase.

Abbreviations

Zkratka	Popis
BIN API	Binary API
CS OTE	Central System of Mmarket Operator (OTE)
EAN	European Article Number – unambiguous international generic identificator
EIC	Energy Identification Code – unambiguous international energetics identificator
FS	Financial security
IM	Continuous Intraday Market with Electricity
IS OTE	Information system of market operator
MP	Market Participant
Protobuf	Protocol buffers – Language and platform neutral spreadable mechanism of serialization of structured data from Google (https://protobuf.dev)
SFS	System of Finance Security (OTE)
XML	Extensible Markup Language

1. Introduction

The aim of this document is to provide a specification for electricity continuous intraday market with electricity interface including AMQP server and the usage of BINARY API protocol buffers message content format.

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 to their systems, then this document should provide description of necessary changes in the interface for implementation BINARY API protocol buffers message content format.

2. External interface description

By reason of ensuring of high throughput and quick distribution of messages from the IM 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 receiving 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 scripts are described in the following parts.

2.1. Communication protocol

Communication with the MQ server is being processed via the AMQP (Advanced Message Queuing Protocol) protocol. It is a public standard for communication layer of applications based on data exchange through messages. Implementation will be performed via the MQ server RabbitMQ version 4.0.3.

AMQP standard defines basic entities:

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

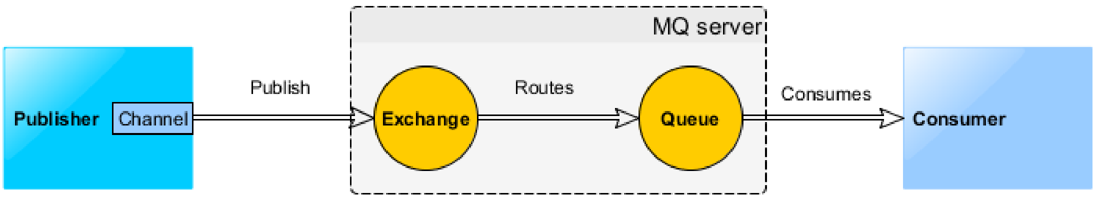


Figure 1 - Communication with MQ server

2.2. Connecting to MQ server

For the connection, an external participant needs to know the following technical information: RabbitMQ server address, port and virtual host identification - this information is specified separately for various OTECOM application environments in the documents "Instruction for the first access to the production/test environment of OTE-COM application" (see http://www.ote-cr.cz/documentation/electricitydocumentation/market-documentation). Description of how to connect a custom client application is available at http://www.rabbitmq.com/api-guide.html.

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.

These channels connect to the individual „queue“ which serve for mutual communication between client and server.

To establish a connection, the external participant must have the following technical details: RabbitMQ server address, port and virtual host identifier – these details are specified individually for each OTE-COM application environment in documents [1], [2]. The description of connection method from a custom client application is available at the following address . External participant provides his own client certificate to OTE.

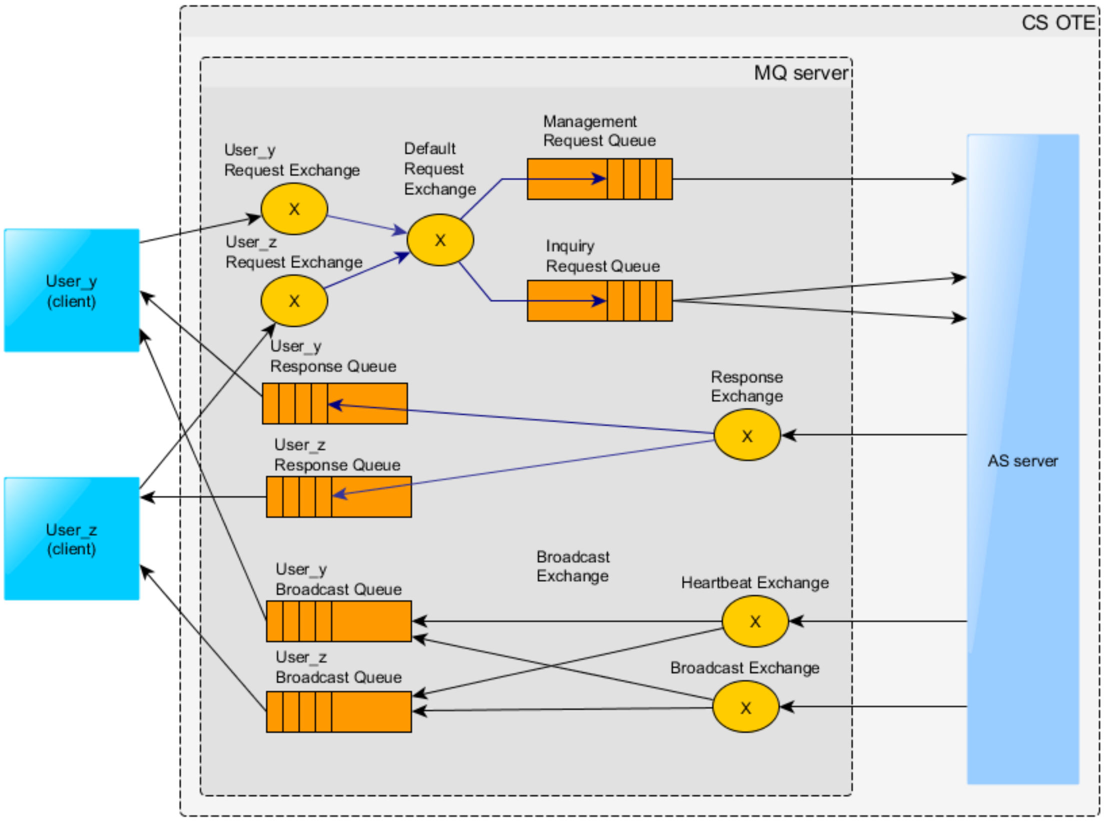


Figure 2 - Connection to MQ server and message flow architecture

2.3. Message exchange types

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

- 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 its own private "*Exchange*" named "*market.exchanges.clientRequest.[USER_ID]*" created in the RabbitMQ server, which is used for sending requests from client to the MQ server. Only the given user has permission to write to the specific exchange.

The user uses the queue named “response queue” for receiving messages addressed to the user, which is not pre-created in AMQP server, but is created by each client individually. The client must create their own anonymous queue during the application start with automatically generated name, whose name is then used in element “*reply-to*” in all messages.

The queue must be created with the following technical details: *durable=false*, *autoDelete=true*, *exclusive=true*.

Types of reqeuests:

- Instruction (Management request) – bid creation, modification, anullation
- Request (Inquiry request) – request for trading data

The response is immediately returned to the user (distributed to ResponseQueue) via message Acknowledgement Response (AckResp) during the*“Management request”* operation. The response to the submitted request is sent (distributed into BroadcastQueue) after the system processes the request. If the request modifies the business data, a broadcast message with relevant content is sent to all users for whom the change is relevant.

The relevant response is sent to the user’s private queue (ResponseQueue) during the “*Inquiry request*” operation.

Request type “*Management request*” is secured by an electronic signature, wrapped in a SignedMessage structure (see chapter. 3 Using the electronic signature) to ensure integrity and non-repudiation. It is generally required to populate the *type* (see chapter 2.6.1 AMQP attributes) attribute during the client-IM application communication.

2.3.2. Mass messages – Broadcast

The system provides two basic types of mass messages:

- Market data messages – messages that contain changes to business data and market state distributed to all logged-in users who have the appropriate rights for the relevant markets.
- Heartbeat messages – messages used to verify active client connection

Every user is connected to their own private queue, created in RabbitMQ and named “*market.broadcastQueue.[USER_ID]“* to receive messages. If the user does not consume messages regularly, their queue might become full and new messages will not be included in their queue. This poses a risk, that they will not receive all the information from the market.

2.3.3. Distribution rules

Description of distribution rules are described in the following table. Some keys are dynamically defined based on the actual market configuration and user access rights.

Routing key	Description
public	Public information, distributed to all users
public.<market_id>	Public information about the current market, distributed to all users, who have access to that market
public.trade.<product_name>	Public information about trades, distributed to all users, who have access to the relevant product
PRTC_<partic_id>	Relevant current market participant information
<product_name>	Relevant product information
<product_name>.<delivery_area>	Relevant product and distribution area information
<product_name>. PRTC_<partic_id>	Relevant PARTIC_ID information based on product
trade	Trade information for administrators only (consists of both sides of the trade)
halfTrade.<product_name>. PRTC_<partic_id>	Private information about established trades (contains only the participant's side of trade)
USR_<user_id>	Private messages addressed exclusively to the user

Table 1 - Distribution rules overview

There is a user-specific example provided as a showcase:

User: 123, Participant: 12, Market access: INTRADAY, Available products: INTRADAY_1H, Available areas: CZ

User will receive messages, that will be sent with some of the following Routing keys:

- public
- public.INTRADAY
- public.trade.INTRADAY_1H
- PRTC_12
- INTRADAY_1H
- INTRADAY_1H.CZ
- INTRADAY_1H.PRTC_12
- halfTrade.INTRADAY_1H.PRTC_12
- USR_123

2.3.4. Sequence use for Broadcast messages

Sequence number is used to identify the order of mass messages, for the purpose of detecting whether any message has not been lost. Sequence number is not contained directly in the message payload, but it is included in the AMQP message header as an attribute called "*market-group-sequence*".

Sequence is always incremented by one for each mass message. Sequences are kept only in memory (they are not stored), which means they will be set to 0 during a server restart or termination. If client receives unexpected value (different value from the last number + 1), client should request current market data from CS OTE system.

Sequence numbers are calculated individually for Routing keys (routing keys – attribute "*market-group-id*" in message header). Each distribution list has its own sequence numbers. Queues connected to the default mass message exchange with the same Routing key will receive duplicate sequence number.

2.3.4.1. Mass message for sequence numbers reconciliation

It is a OTE-COM's feature ensuring more robust communication with MP. There is a special mass message containing information about last used sequence numbers for each Routing key, which is distributed to each connected client.

SequenceNumbersRprt distribution messages must follow these rules:

1. One and the exact same distribution message is sent to all clients
2. Each client receives the distribution message, which contains the last used sequence numbers for all private as well as public Routing keys, including those that are not relevant
3. Every client selects its relevant Routing keys
4. Each Routing key is published only once
5. A message is distributed every x seconds[2]
    - The sequence number of a Routing key is the last known value of the exact moment of the creation of the SequenceNumbersRprt message
    - Every time the server restarts, the sequence number is reset to null

Message/Field	Type	m/o	No.	Data Type	Short description
SequenceNumbersRprt	MSG			Structure	
standard_header	*FIELD*	*m*		*Structure*	*Standard header of each message. Please see chapter 2.6.7* Standard message header***Error! Reference source not found.***
seq_numbers	FIELD	m	1..n	*Structure*	
routing_key	FIELD	o		String	Routing Key, Each Routing Key is present only once
sequence	FIELD	o		Integer	Latest sequence number of the Routing Key

Table 2 -SequenceNumbersRprt message structure

2.4. Invalid and unrouteable requests

It is important that clients create a valid protobuf (proto3) message content, which they send to CS OTE. Messages with protobuf (proto3) content that CS OTE system can not read will be rejected.

If the CS OTE system is unable to process a request because it is invalid or can not be processed, a negative response will be sent. The response contains details including the reason why the request was not processed.

If the CS OTE system can not process a request due to an invalid or missing message header version, a native error response is sent. In this case, the response has an attribute content-type set to value *market/error*. The message body contains an error message encoded in UTF-8. Discovery of a message validation error within the CS OTE system leads to sending a native error response. These validation errors occur in the following cases:

- Missing AMQP message attribute *user-id*
- Missing AMQP message attribute *content-type*
- Missing AMQP message attribute *reply-to*
- Missing AMQP message attribute *correlation-id*
- Missing AMQP message attribute *type*
- Unknown AMQP protobuf (proto3) message content

[2] The expected interval is 5 seconds, however it will be specified later during the project implementation.

If the CS OTE system can not process a request because it is not running (due to an outage or during a restart), the request will be cancelled on the AMQP server side and the client will be informed via their "return listener".

2.5. Failover processing

In case the AMQP is not running (due to an outage or restart), the client connection is lost. If the client has a registered a "shutdown listener", they will receive an outage notification from the AMQP server. After successfully reconnecting to the AMQP server, the client must log in again.

2.6. General information about communication messages

2.6.1. AMQP attributes

The AMQP attributes used for communication between the client and an IM application.

AMQP Message Atribut	Popis
content-type	Contains information about the used 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/request; version=x (Used by the client when sending requests) ▪ market/response; version=x ▪ market/broadcast; version=x ▪ market/heartbeat; version=x ▪ market/error; version=x **Current version of messages is 5.**
type	Contains fully qualified name of the message
signed-type	Contains name of the signed message (message signing is applicable for AddOrderReq ModifyOrderReq, and ModifyAllOrdersReq). Used only for type=SignedMessage.
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. OrdrExecutionRprt) .
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 3 - Message attributes according to AMQP

2.6.2. Protobuf convention

Messages that contain the AMQP content-type attribute based on message types *market/request, market/response* and *market/broadcast*, include a binary-formatted protobuf (proto3) data section.

The binary format protobuf (proto3) definition uses the following conventions:

- **MSG (MESSAGE):** Used only in custom messages (e.g. *AddOrderReq*)
- **FIELD:** It describes custom value fields within the message (e.g. *price*), as well as value structures (e.g. *order* structure in the *AddOrderReq* message).

2.6.3. Quantity values within messages

Quantity values in all messages are expressed as integer (int32). A custom value is defined by a field group in the message *ProductInfoRprt – decimal_shift_quantity, min_quantity* and *quantity_unit* (see chapter 2.8.4.13 Product Information Report (ProductInfoRprt)). The *decimal_shift_quantity* item defines the position of decimal point within the input integer (e.g. a quantity value of 5200 with field value *decimal_shift_quantity = 3* equals 5.200 value).

The item *min_quantity* ensures the smallest possible step after entering the quantity (e.g. *min_quantity = 100* and *decimal_shift_quantity = 3* mean that it is possible to increase the quantity in steps of 0.1).

The item *quantity_unit* defines quantity unit.

2.6.4. Price values within messages

Price values in all messages are expressed as integer (int64). A custom value is defined by a field group in the message *ProductInfoRprt – decimal_shift_price, tick_size* and *currency* (see chapter 2.8.4.13 Product Information Report (ProductInfoRprt)).

The *decimal_shift_price* item defines the position of decimal point within the input integer (e.g. a price value of 3624 with *decimal_shift_price = 2* equals 36.24 value).

The item *tick_size* defines the smallest possible step after entering the price (e.g. *tick_size = 1* and *decimal_shift_price = 2* mean that it is possible to increase the price in steps of 0.01).

The item *currency* defines trade currency.

2.6.5. Format of date items within messages

Date items within messages use the Timestamp data type (native google.protobuf.Timestamp), which uses the unix UTC timestamp as the base value (for more information see https://www.unixtimestamp.com/).

2.6.6. Heartbeat message

The Heartbeat message contains text with the attributes “server-timestamp” and “interal-length”. Both attributes use milliseconds. The first one represents the difference between an actual time and the date 1.1.1970 0:00:00 UTC.

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

2.6.7. Standard message header

Every message contains the standard message header with the following items:

Message/Field	Type	m/o	No.	Data Type	Short description
standard_header	FIELD	m		Structure	
market_id	FIELD	m		Enum	Market Identification Code (MIC) of the market to which the request is sent or from which the request originates. The following values are allowed: **"MARKET_ID_TYPE_XBID"**: XBID Intraday market **"MARKET_ID_TYPE_IM"**: OTE secondary Intraday market (fallback to XBID).
client_correlation_id	FIELD	o		String	The client data field in this section can be used by the client to store information or meta-data about a request. The content in this field is not used by CS OTE system. Content is sent back to client in response.

Table 4 - Message header

2.6.8. Individual message parameter descriptions

The following chapters define the following message parameters:

- Type – message type
  - Inquiry Request – data request
  - Management request – performance request
  - Broadcast – mass message
- Role – role based accessibility
- Routing key –message routing to MQ server
- Message limit – max. number of name-specific messages that can be processed by the server for each user within the defined time limit, without being rejected by the server. The format is defined as a/b, where the first "a" represents max. number of messages allowed within a 1-minute time limit and "b" represents max. number of messages within a 1-hour time limit. If no time limit is set, the number of messages is unlimited. Limit is calculated separately for each *market_id*.

2.7. Communication scenarios

2.7.1. User log-in, log-out

The base communication scenario is used for user log-in, log-out access to the system as well as actual system information requests. The user must intiate data communication with MQ server by sending a log-in request *LoginReq* within a 30-second time limit, otherwise the connection will be terminated. The response is *UserRprt* returned as a result of a successful validation, otherwise *ErrResp* is sent to the client in case of an unsuccessful validation.

The user must send a *LogoutReq* message during client application termination. If the user does not send a log-out request, they will be logged out according to the rules that define a lost connection.

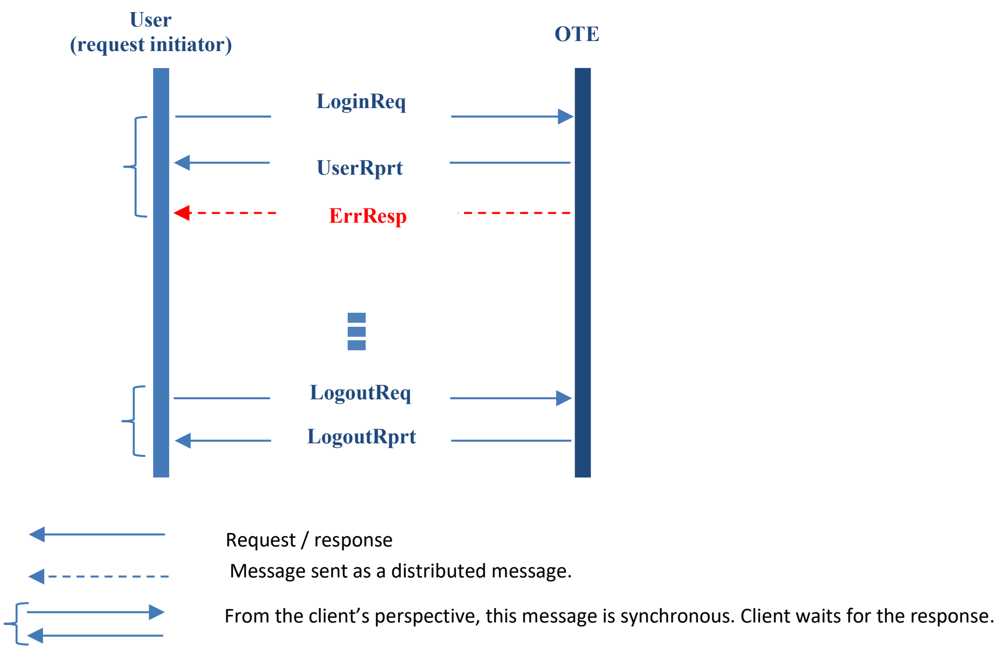

User
(request initiator)
OTE
LoginReq
UserRprt
ErrResp
LogoutReq
LogoutRprt
Request / response
Message sent as a distributed message.
From the client’s perspective, this message is synchronous. Client waits for the response.

Figure 3 - User login/logout sequence diagram

2.7.2. Base scenario of bid submission – error processing

Bid creation or modification requests are formally validated and according to the control results, the user is either notified via a formal message receipt confirmation (*AckResp*) or formal error message (*ErrResp*). In case of a successful receipt (after sending *AckResp* to the user), the request validation follows on the OTE side and the XBID central side. In case of an unsuccessful validation, an *ErrResp* error notification is generated for the user, which is then sent as a distributed message.

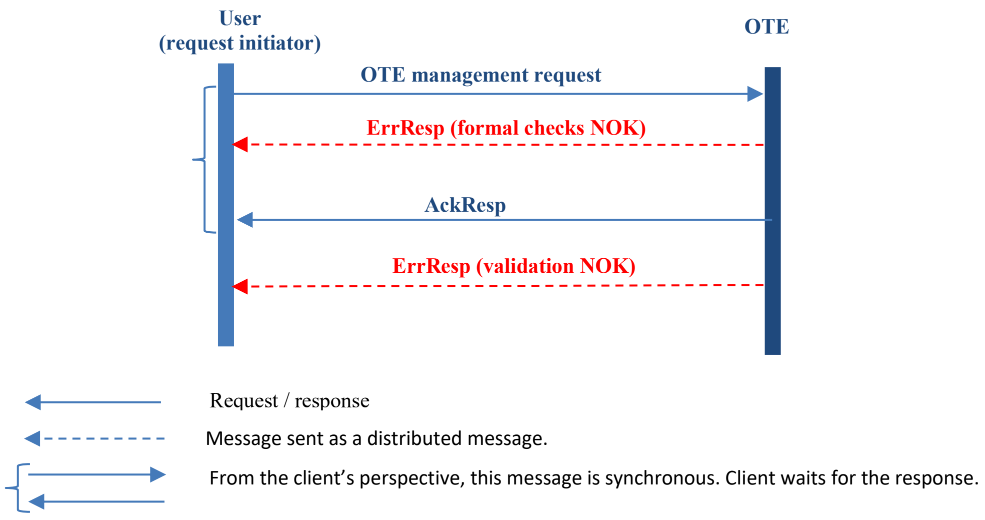

User
(request initiator)
OTE
OTE management request
ErrResp (formal checks NOK)
AckResp
ErrResp (validation NOK)
Request / response
Message sent as a distributed message.
From the client’s perspective, this message is synchronous. Client waits for the response.

Figure 4 - General request sequence diagram

2.7.3. Bid manipulation

The user submits an bid using the *AddOrderReq* (or modifies the bid using the *ModifyOrderReq*) and the application server responds with an *AckResp* message, either confirming that the request was successfully received or informing the user about the error message definition via the *ErrResp* message. The server sends the submission/modification result message through a private *OrderExecutionRprt* message as well as via a *MessageRprt* private message.

Next, in case of a successful bid submission, the *PublicOrderBooksDeltaRprt* message, including the public order book modifications, is sent to all OTE users. This also applies in the case of a bid modification (or a set of bids), but only if such modification changes its timestamp.

In case of a trade establishment, the *TradeCaptureRprt* private message is sent to the bid owner, and the public *MessageRprt* message is sent to all OTE users. In case of a trade establishment between OTE users (intrastate Czech trade), private messages *OrderExecutionRprt* and *TradeCaptureRprt* are also sent to the counter-bid owner. In case of interstate trade the XBID central side ensures notifying the counter-bid owner outside of the Czech area.

There is also an option to submit a bid request via the *OrderReq*, where the response is sent in the form of a distributed private message *OrderExecutionRprt* from OTE back to the request initiator.

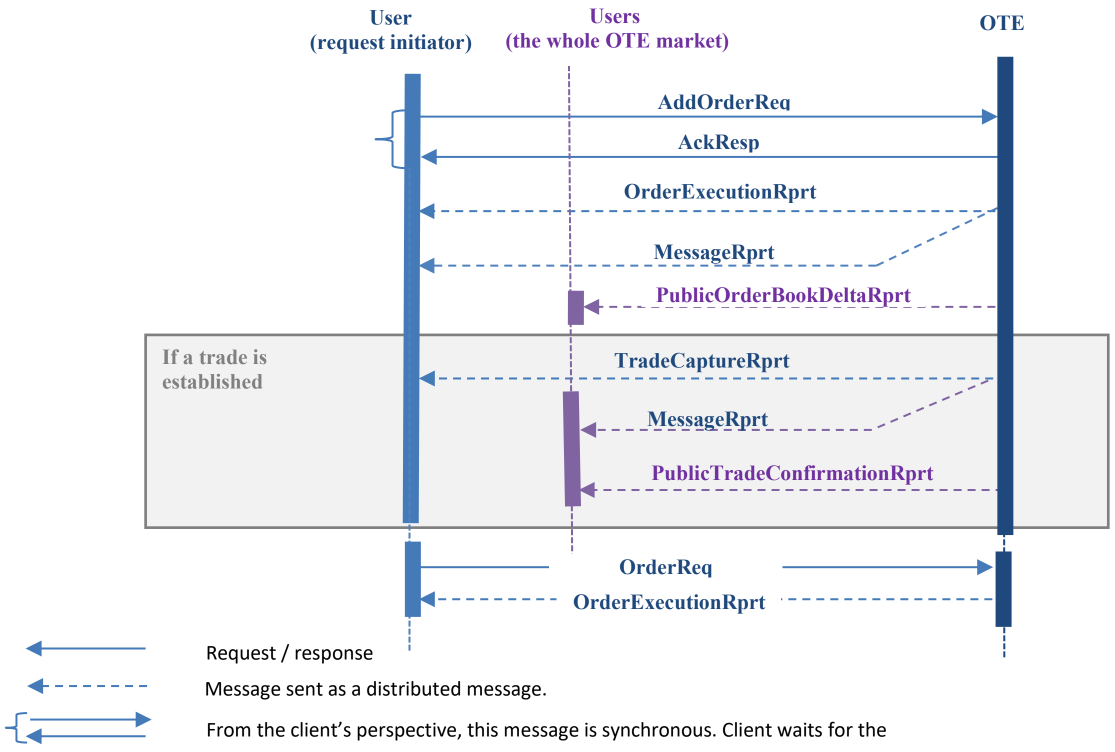

User (request initiator)
Users (the whole OTE market)
OTE
AddOrderReq
AckResp
OrderExecutionRprt
MessageRprt
PublicOrderBookDeltaRprt
If a trade is established
TradeCaptureRprt
MessageRprt
PublicTradeConfirmationRprt
OrderReq
OrderExecutionRprt
Request / response
Message sent as a distributed message.
From the client's perspective, this message is synchronous. Client waits for the

Figure 5 - Order submission and trade establishment sequence diagram

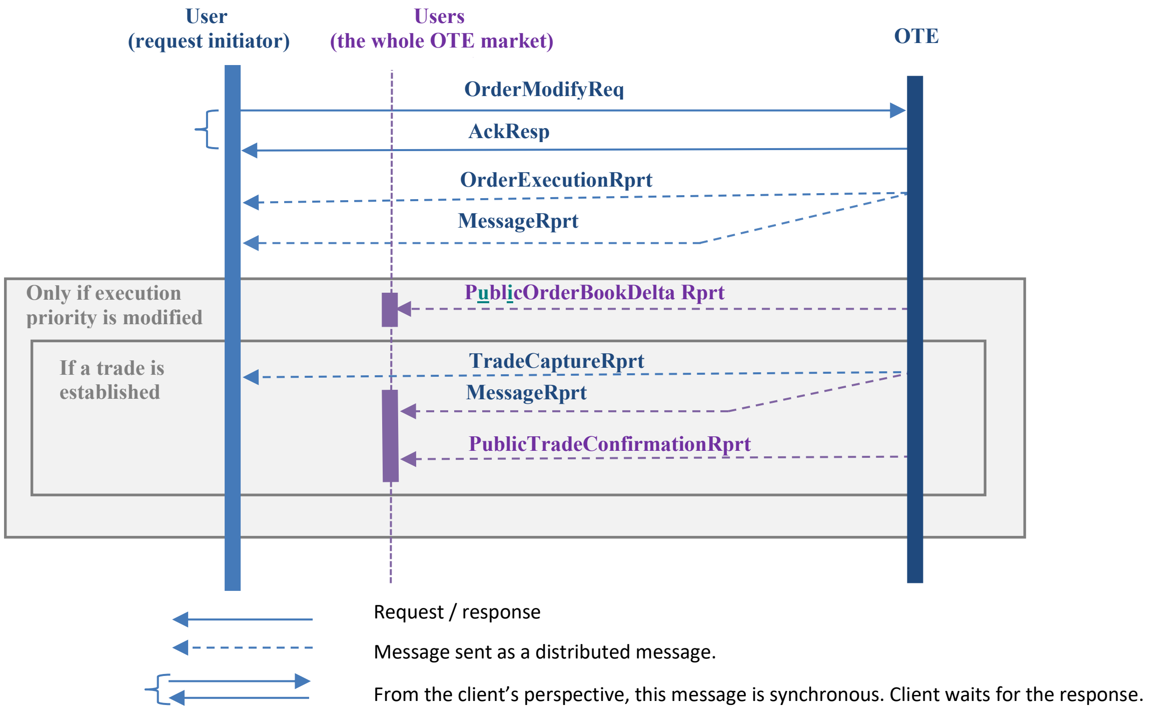

User
(request initiator)
Users
(the whole OTE market)
OTE
OrderModifyReq
AckResp
OrderExecutionRprt
MessageRprt
Only if execution priority is modified
PublicOrderBookDelta Rprt
If a trade is established
TradeCaptureRprt
MessageRprt
PublicTradeConfirmationRprt
Request / response
Message sent as a distributed message.
From the client's perspective, this message is synchronous. Client waits for the response.

Figure 6 - Order modification and trade establishment sequence diagram

If the submitted creation or modification request is invalid, the *ErrResp* message is sent as a response to the request initiator (see chapt. 2.7.2 Base scenario of bid submission – error).

During a bulk bid modification (such as activation, deactivation and annulment) based on the *ModifyAllOrdersReq* user request, a message sequence similar to the one in case of bid submission or modification is applied, as described in the introduction section of this chapter, with the information that these messages contain information about the set of modified *n* bids. If there is a *m* amount of trades established (where the number of trades $m \leq$ number of bids n), then the relevant messages concerning trade establishment contain information about a set of *m* trades. If a set of *m* trades is created (where the number of trades $m \leq$ number of bids n), then the system will provide a set of *m* relevant trade establishment messages.

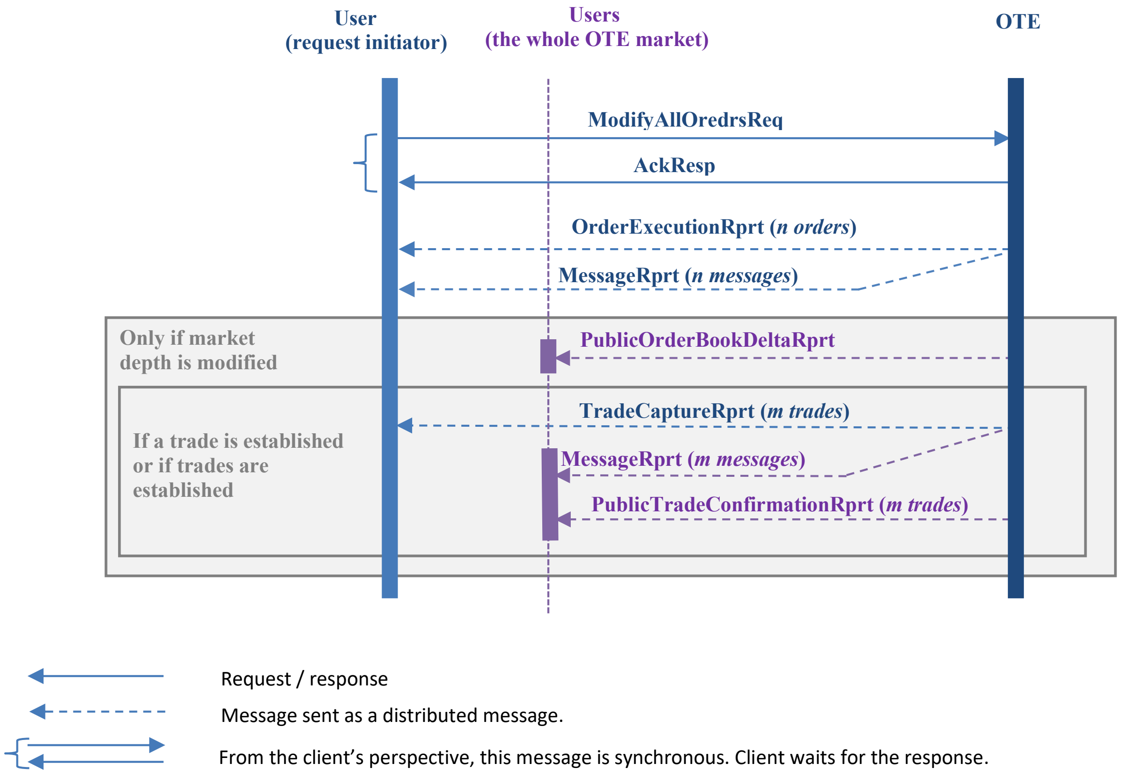

User
(request initiator)
Users
(the whole OTE market)
OTE
ModifyAllOredrsReq
AckResp
OrderExecutionRprt (n orders)
MessageRprt (n messages)
Only if market depth is modified
PublicOrderBookDeltaRprt
If a trade is established or if trades are established
TradeCaptureRprt (m trades)
MessageRprt (m messages)
PublicTradeConfirmationRprt (m trades)
Request / response
Message sent as a distributed message.
From the client's perspective, this message is synchronous. Client waits for the response.

Figure 7 - Bulk order modification (or deactivation) and the subsequent order request sequence diagram

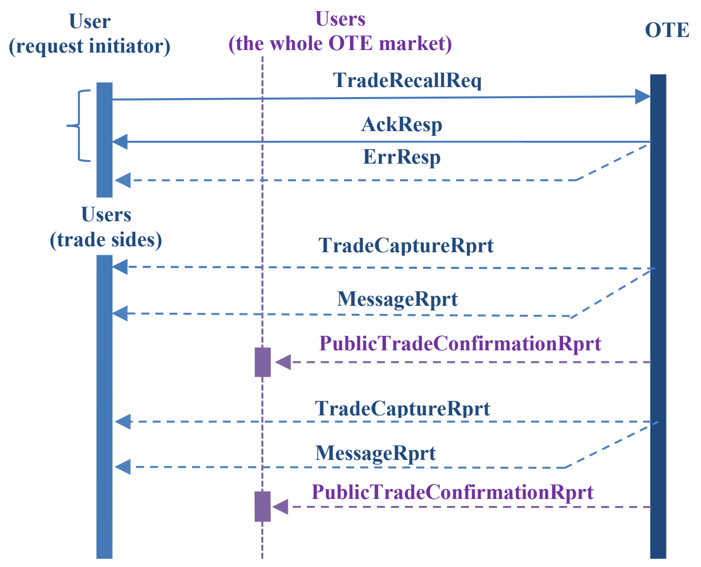
2.7.4. Trade callback

Note: The trade callback communication scenario is not available.

The user can submit a trade callback request via the *TradeCallbackReq* message. If the request is formally valid, the *AckResp* is sent to the user, otherwise the *ErrResp* containing the error specification is sent. After the internal request processing, this request is forwarded to the XBID central side.

XBID then processes the request and marks the trade as cancelled by changing its status, thereby notifying OTE, who then updates the trade status according to XBID and informs the request intiator via the *TradeCaptureReprt* message as well as the audit log *MessageRprt* message. All OTE users are also notified via *PublicTradeConfirmationRprt*.

After trade callback request process is completed on the central side, XBID informs OTE about the trade callback result, which is then reflected by OTE through a trade update change according to XBID. The request initiator is then informed again via the *TradeCaptureRprt* message as well as the audit log *MessageRprt*, which also informs about the trade callback result. All OTE users are informed about the result too via the *PublicTradeConfirmationRprt* message.



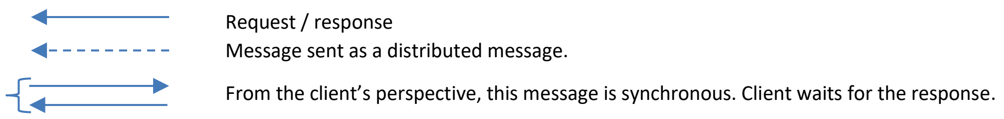

Request / response
Message sent as a distributed message.
From the client’s perspective, this message is synchronous. Client waits for the response.

Figure 8 - Trade recall sequence diagram

2.7.5. Trade cancellation

The process of initiatiating and cancelling a trade is performed within the central XBID solution, all NEMO are only informed about the trade cancellation. If at least one side of the cancelled trade is a OTE market participant, then OTE sends the *TradeCaptureRprt* and the *MessageRprt* distributed message to the cancelled trade market participant (if the cancelled trade is local, then both market participants will receive the message). All users are notified by OTE about the trade cancellation by sending the *PublicTradeConfirmationRprt* public message. This does not include cancelled trades, where both sides are not OTE users.

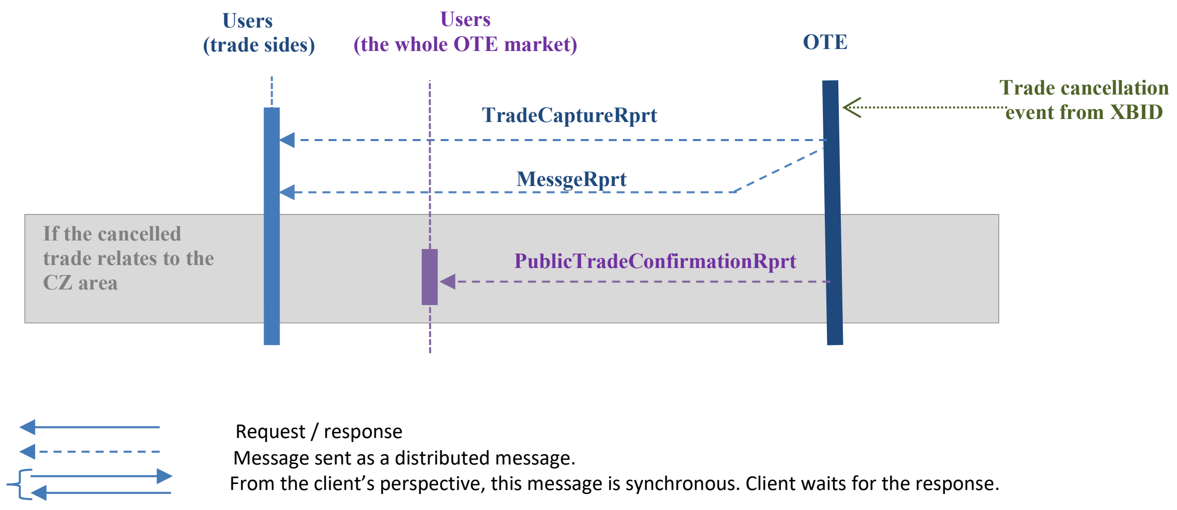

Users
(trade sides)
Users
(the whole OTE market)
OTE
Trade cancellation event from XBID
TradeCaptureRprt
MessgeRprt
If the cancelled trade relates to the CZ area
PublicTradeConfirmationRprt
Request / response
Message sent as a distributed message.
From the client's perspective, this message is synchronous. Client waits for the response.

Figure 9 - Trade cancellation sequence diagram

2.7.6. Public bid data request

The user sends a request for the list of active public market bid through *PublicOrderBooksReq* and the server responds with the *PublicOrderBooksResp* containg a copy of these bids. This is how the client receives the entire set of active bids withing the system. If a completely new bid is created or already existing bid is modified, the *PublicOrderBooksDeltaRprt* mass message will be sent.

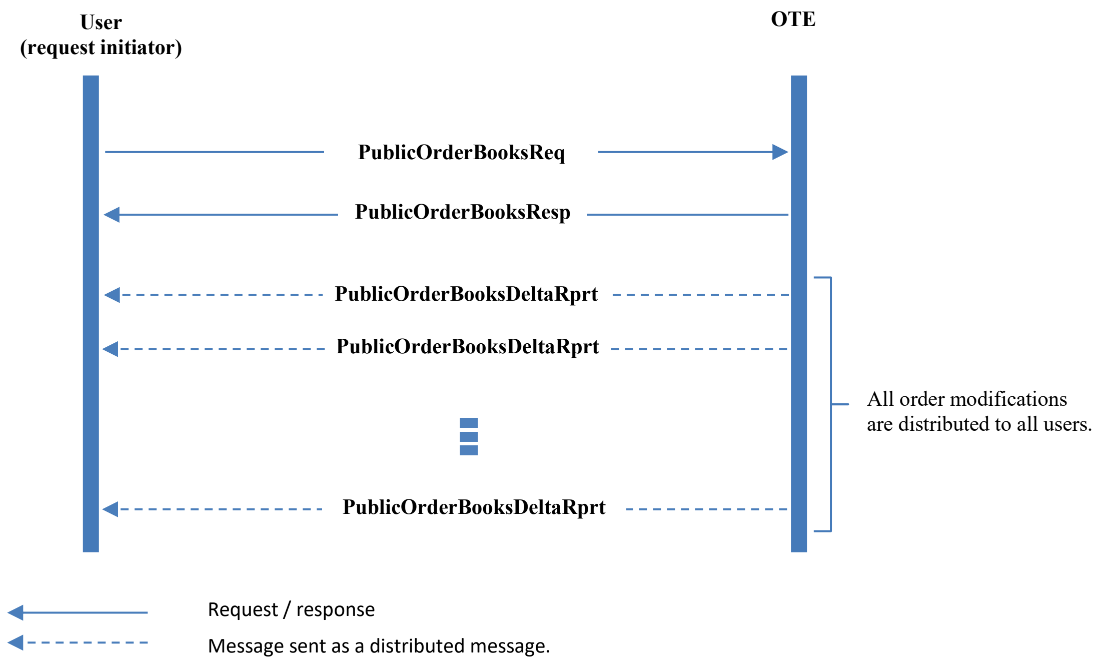

User
(request initiator)
OTE
PublicOrderBooksReq
PublicOrderBooksResp
PublicOrderBooksDeltaRprt
PublicOrderBooksDeltaRprt
PublicOrderBooksDeltaRprt
All order modifications are distributed to all users.
Request / response
Message sent as a distributed message.

Figure 10 - Order request sequence diagram

2.7.7. Public trade data request

The user sends a request for established market trades through the *PublicTradeConfirmationReq* and the server responds with the *PublicTradeConfirmationRprt* containing a copy of these trades. The case of a trade establishment is illustrated in the following messages.

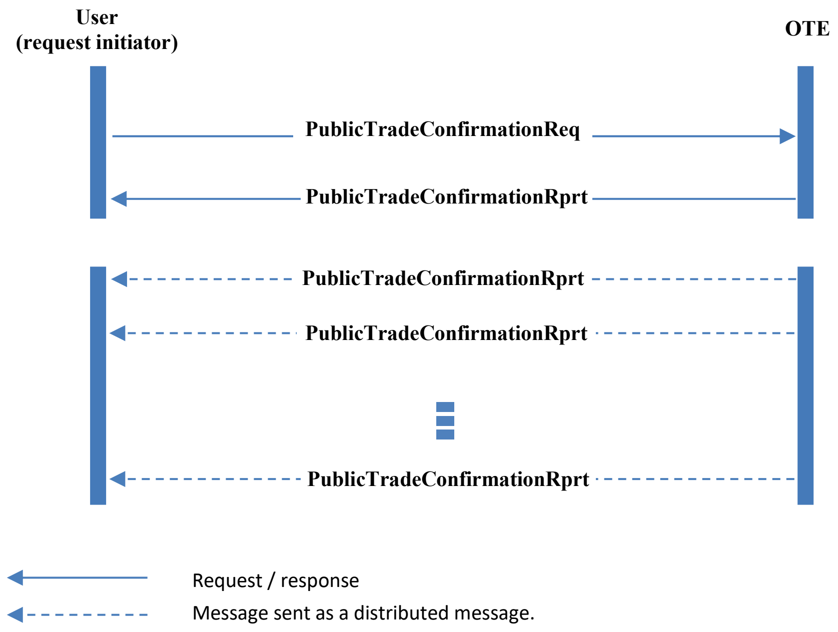

User
(request initiator)
OTE
PublicTradeConfirmationReq
PublicTradeConfirmationRprt
PublicTradeConfirmationRprt
PublicTradeConfirmationRprt
PublicTradeConfirmationRprt
Request / response
Message sent as a distributed message.

Figure 11 - Trade request sequence diagram

2.7.8. Information message request

After the successful log-in of the user, the user then sends the *MessageReq* request to the server inquiring about a message list. The user may specify within this request whether they would like to receive only private messages, only public messages or all messages. They will be delivered within the required time limit via the *MessageRprt* message and all the following messages are then distributed automatically.

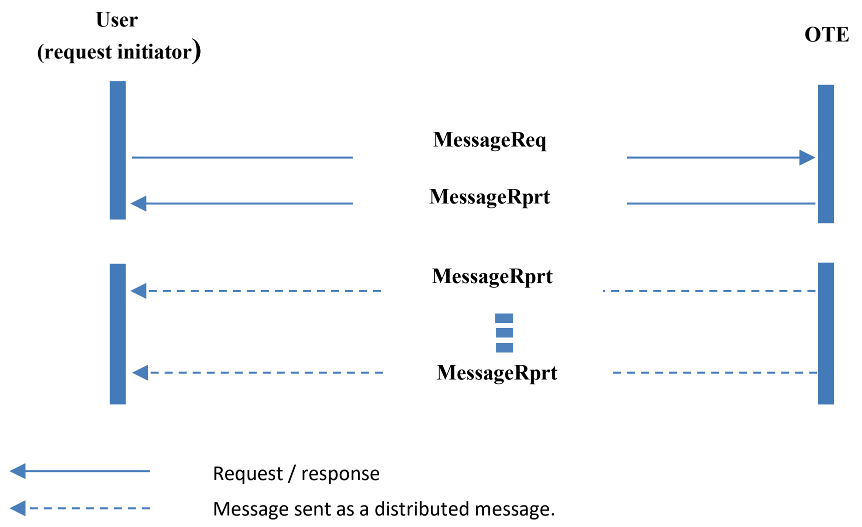


Figure 12 - Market messages request sequence diagram

2.7.9. Product and market contract request

The user may request the list of valid products via the *ProductInfoReq* request and the response is sent in the message *ProductInfoRprt*. In case of a product modification, the *ProductInfoRprt* distributed public message is sent to all OTE users.

Similar situation occurs in case of the Contract information. The user may request the list of valid contract via the *ContractInfoReq* and the response is delivered through the message *ContractInfoRprt*. In case of a contract modification, all OTE users will receive the *ContractInfoRprt* public distributed message.

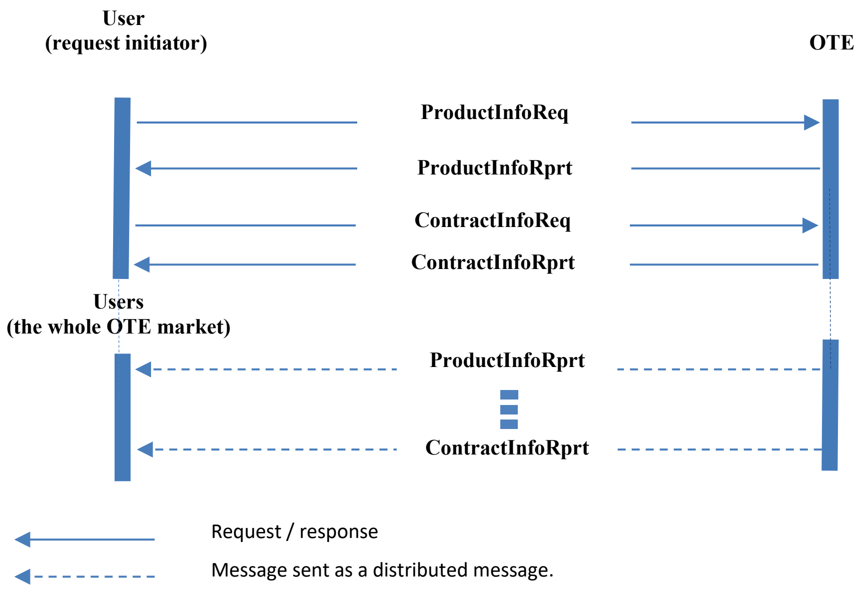


Figure 13 - Product and contract request sequence diagram

2.7.10. Market status request

The user may request the current market status information through the *MarketStateReq* request and the response will be sent via the *MarketStateRprt* message. In case of a market status modification, the *MarketStateRprt* public distributed message will be sent to all OTE users. These messages allow monitoring of the current market status, to ensure that it is not in the "Deactivated" state where trading is stopped.

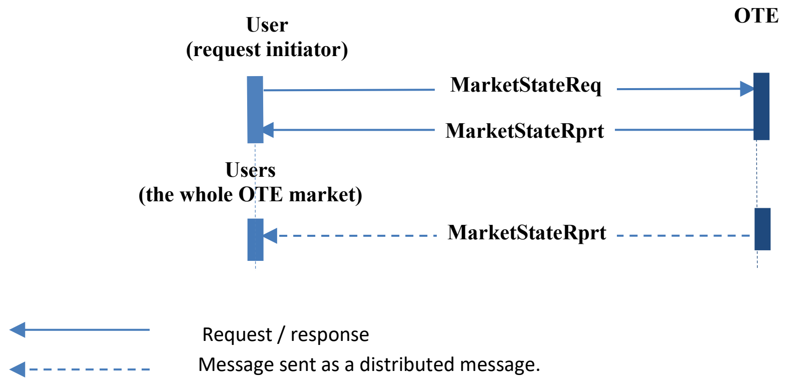


Figure 14 - Market state request sequence diagram

2.7.11. Capacity data request

The user may request the current H2H matrix status via the *HubToHubReq* request and the response is sent as the *HubToHubResp* distributed message, whose structure is similar to the *HubToHubNtfRprt* distributed message.

All the delta changes of the capacity data values represented by the H2H matrix are automatically distributed to all users via the *HubToHubNtfRprt* distributed message.

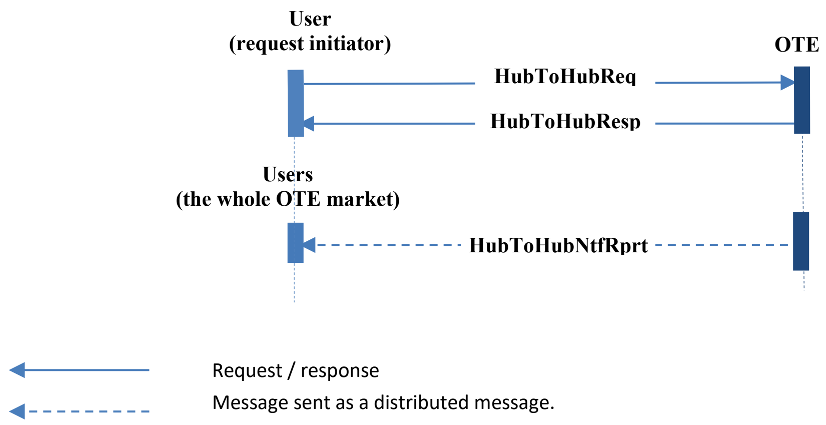


Figure 15 - H2H matrix request sequence diagram

2.7.12. Market area request

Users may request market area information via the *MarketAreaInfoReq*, the *MarketAreaInfoRprt* is sent as a response to the initiator of this request.

OTE users are informed via the *MarketAreaInfoRprt* distributed message, in case any attribute within the market area is modified.

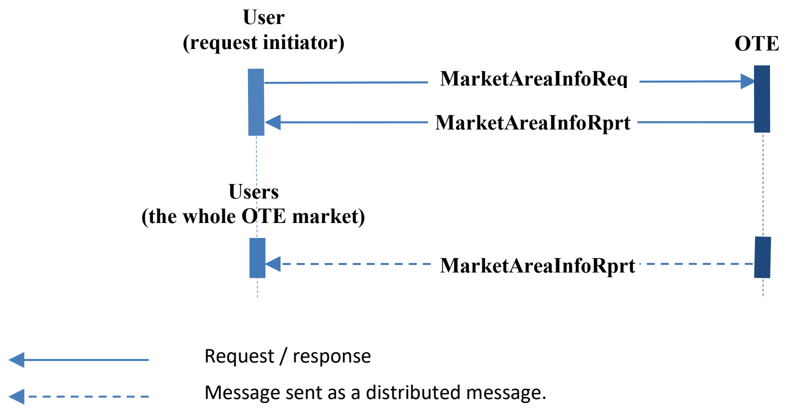


Figure 16 - Market area request sequence diagram

2.7.13. Delivery area request

Users may request delivery area data via the *DeliveryAreaInfoReq* message too, the *DeliveryAreaInfoRprt* is sent to the initiator as a response.

OTE users are infomed via the *DeliveryAreaInfoRprt* distributed message in case any attribute within the delivery area is modified.

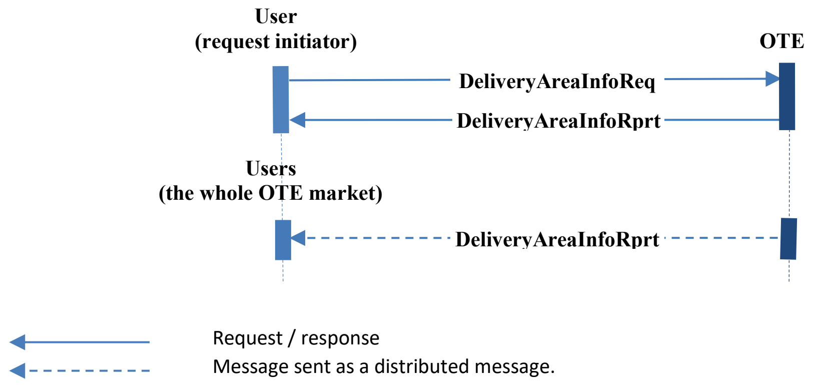


Figure 17 - Delivery area request sequence diagram

2.7.14. Market state distributed message

An informative message about the market state, distributed from XBID to OTE during market state changes (ACTI->HIBE and back) and sent from OTE to market participants.

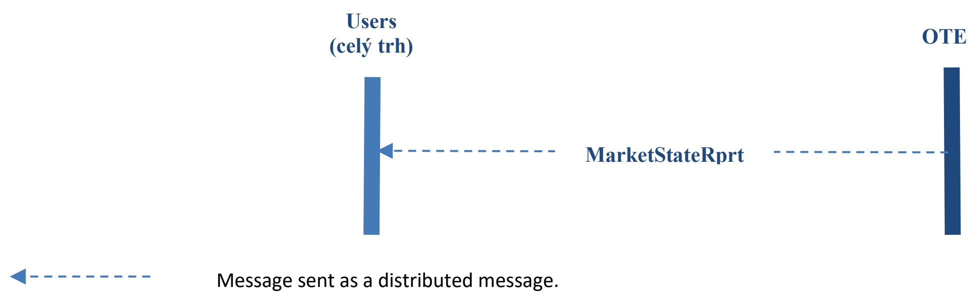


Figure 18 - Market state distributed message sequence diagram

2.8. Communication messages

The content of all messages exchanged between the user and IM application within the above mentioned communication scenarios is in binary format protobuf. A detailed description of all messages is provided in the following chapters.

Modification summary compared to the default XML interface:

- Some BINARY API specifications are identical to the verified protobuf (proto3) procedure recommendations, for example:
  - names of enum values, for example enum value of the item *validity_restriction* = "VALIDITY_RESTRICTION_TYPE_GFS", use the name of the common enum data type as a prefix – in this example it is "ValidityRestrictionType"
  - each enum type of protobuf definition includes a dedicated enum value "_UNSPECIFIED", which the protobuf framework interprets as an implicit value if the current item is empty
  - Items with timestamp and duration values use the corresponding pre-built protobuf data types.
- The naming of certain messages has been modified to increase clarity and readability (for example from the *PblcTradeConfRprt* message to the *PublicTradeConfirmationRprt)*. The following rules apply to the message name postfix:
  - Req postfix – data request message
  - Resp postfix – response message to a request
  - Rprt postfix – distributed message or request response message
- Each message item is, in some cases, renamed to improve clarity and reability, using the current protobuf (proto3) naming recommendations:
  - Items follow the *lower_snake_case* naming convention , where all letters are lowercase and words are separated by an underscore
  - The enum values of the enum-type items follow the CAPITALS_WITH_UNDERSCORES naming convention
  - In case where items contain a value field (with cardinality > 1), the name uses the English plural form with the letter "s" at the end (for example the structure *bids* within the *AddOrderReq* message)
- Message cleanup – certain obsolete items have been removed, such as:
  - Items that include field encapsulation into different structures, for example: *OrdrlList, MktAreaList, MsgList* etc.

- Removal of a *clientData* structure from the header of all *standard_header* messages, with the exception of retaining one item from this structure – *client_correlation_id*, which is now placed at the same level as the *market_id* item

The order of items in individual messages is determined by the definition protobuf(.proto) file, see [4]. The order of items specified in the AK manual is not binding.

Note: modifications compared to the original XML format in the naming of messages, items, data types and enum types are not represented in this document. Nevertheless these modifications compared to the original XML format are evident from the document [3], which maps the OTECOM message items from the original XML format to the new protobuf (proto3) format with changes to the message, item, data type and enum type names. Modifications are highlighted in red.

2.8.1. General requests and responses

2.8.1.1. Login Request (LoginReq)

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

Login request. The system responds via the *UserRprt.*

Message/Field	Type	m/o	No.	Data Type	Short description
LoginReq	MSG			Structure	
standard_header	*FIELD*	*m*		*Structure*	*Standard header of each message. Please see chapter* 2.6.7 Standard message header.
user	FIELD	m		String	Login ID of the user that want to login to the CS OTE system.
force	FIELD	m		Boolean	Flag that indicates if this user wants to force a login even if a user with the same credentials is already logged in into the CS OTE system.
disconnect_action	FIELD	m		Enum	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: "**DISCONNECT_ACTION_TYPE_NO**": No action is executed. "**DISCONNECT_ACTION_TYPE_DEACT_USER_ORDERS**": All orders of this user will be deactivated.

Table 5 - Login request message structure

2.8.1.2. 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>

This message contains the basic user attributes. The *UserRprt* message is returned as a response to the *LoginReq* and is also distributed when a configuration modification of a user's product assignment occurs.

Message/Field	Type	m/o	No.	Data Type	Short description
UserRprt	MSG			Structure	
standard_header	*FIELD*	*m*		*Structure*	*Standard header of each message. Please see chapter 2.6.7 Standard message header.*
session_id	FIELD	m		Integer(64)	The current session id of the user given after login to the system.
connection_loss_message	FIELD	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.
user	FIELD	m	1..1	Structure	
name	FIELD	m		String	Name of the user.
partic_name	FIELD	m		String	Participant name.
partic_id	FIELD	m		Integer	The participant id the user belongs to.
state	FIELD	m		Enum	Current state of the User. The following values are allowed: "**REFERENCE_DATA_STATE_TYPE_ACTI**": User is active. It is possible to trade using this User. "**REFERENCE_DATA_STATE_TYPE_DELE**": User is deleted. Trading using this User is not possible. "**REFERENCE_DATA_STATE_TYPE_SUSP**": User is suspended. Trading using this User is not possible.
user_roles	FIELD	m	1..n	String	Contains the user roles assigned to the user
user_id	FIELD	m		Integer	The unique identifier of a user.
revision_no	FIELD	m		Integer(64)	Revision number of this user. Always increasing upon a change.
assigned_markets	FIELD	m	1..n	Structure	
market_id	FIELD	m		Enum	Market Identification Code. The following values are allowed: **“MARKET_ID_TYPE_XBID”**: XBID Intraday market **"MARKET_ID_TYPE_IM"**: OTE secondary Intraday market (fallback to XBID).
default_delivery_area_id	FIELD	m		String	Delivery Area ID.

Table 6 - User report message structure

2.8.1.3. Logout Request (LogoutReq)

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

The CS OTE system user logout request.

Message/Field	Type	m/o	No.	Data Type	Short description
LogoutReq	MSG			Structure	
standard_header	*FIELD*	*m*		*Structure*	*Standard header of each message. Please see chapter 2.6.7 Standard message header.*
session_id	FIELD	m		Integer(64)	Session id of the client session passed to the client on login.

Table 7 - Logout request message structure

2.8.1.4. 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>

This message indicates a user logout from the CS OTE system. It is sent either as a response to the logout request *LogoutReq* or as a mass message triggered by a concurrent forced login of the same user (force=true).

Message/Field	Type	m/o	No.	Data Type	Short description
LogoutRprt	MSG			Structure	
standard_header	*FIELD*	*m*		*Structure*	*Standard header of each message. Please see chapter 2.6.7 Standard message header.*
session_id	FIELD	m		Integer(64)	Session id of the client session passed to the client on login.
user_id	FIELD	m		Integer	User ID identification.
text	FIELD	o		String	Text field containing information about the reason of the logout.

Table 8 - Logout report message structure

2.8.1.5. Acknowledgement Response (AckResp)

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

A confirmation message indicating that the request has been accepted for processing.

Message/Field	Type	m/o	No.	Data Type	Short description
AckResp	MSG			Structure	
standard_header	*FIELD*	*m*		*Structure*	*Standard header of each message. Please see chapter 2.6.7 Standard message header.*

Table 9 - Acknowledgement response message structure

2.8.1.6. 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>

An error message distributed in case of unsuccessful processing of a request or inquiry.

Message/Field		Type	m/o	No.	Data Type	Short description
ErrResp		MSG			Structure	
standard_header		*FIELD*	*m*		*Structure*	*Standard header of each message. Please see chapter 2.6.7 Standard message header.*
errors		FIELD	m	1..n	Structure	
	error_code	FIELD	m		Integer	Predefined error codes. Some error messages do not have a specific error code. In this case the value is 0.
	error_en	FIELD	m		String	The error message for this error – English version.
	error_cz	FIELD	m		String	The error message for this error – Czech version.
	client_order_id	FIELD	o		String	Client order ID.

Table 10 - Error response message structure

2.8.2. Bid submission and management

2.8.2.1. Add Order Request (AddOrderReq)

AddOrderReq	
Type:	Management Request
Roles:	EmtasImIns
Routing Keys:	`market.request.management`

The submission of one or more bids. The maximum number of bids within a single message is 25. The message must be encapsulated and signed using the SignedMessage message, see chapt. 3 Using the electronic signature.

Message/Field		Type	m/o	No.	Data Type	Short description
AddOrderReq		MSG	m	1	Structure	
standard_header		*FIELD*	m		*Structure*	*Standard header of each message. Please see chapter 2.6.7 Standard message header.*
list_execution_instruction		FIELD	o		Enum	Defines the execution instruction for the whole list of orders: **"LIST_EXECUTION_INSTRUCTION_TYPE_LNKD"**: Linked orders - the provided orders are linked together and should be executed all at once. This option can only be used, if all orders have execution restriction ORDER_EXECUTION_RESTRICTION_TYPE_FOK (Fill or Kill). Orders can be submitted for Contracts of different Products. In case one of the orders cannot be executed, the whole list is not executed. The Linked Orders feature is configurable and might be turned off. **"LIST_EXECUTION_INSTRUCTION_TYPE_NONE"**: All orders are treated independently. This is the Default Value. **"LIST_EXECUTION_INSTRUCTION_TYPE_VALID"**: All orders must be valid, meaning they must past the order validation of the XBID system (e.g. the price of the order must be in the price range of the product). If one order does not pass the validation, the full list of submitted orders is rejected.
orders		FIELD	m	1..25	Structure	
	state	FIELD	o		Enum	**"ORDER_ENTRY_STATE_TYPE_ACTI"**: The order is entered and immediately exposed to the market for execution. This is the default value. **"ORDER_ ENTRY_STATE_TYPE_HIBE"**: The order is entered into the CS OTE system but not exposed to the market.
	validity_restriction	FIELD	o		Enum	Validity restriction of the order. If this field is omitted, the order will be treated as a "Good for Session" order. Valid values: **"VALIDITY_RESTRICTION_TYPE_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. **"VALIDITY_RESTRICTION_TYPE_GTD"**. The order rests in the order book until the date specified in the validity_date field. **"VALIDITY_RESTRICTION_TYPE_NON"** (No validity restriction): Mandatory for orders with the execution restriction "ORDER_EXECUTION_RESTRICTION_TYPE_NON FOK" or "ORDER_EXECUTION_RESTRICTION_TYPE_NON IOC".
	validity_date	FIELD	o		Timestamp	This field is mandatory in case of validityRes equals "VALIDITY_RESTRICTION_TYPE_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.
	text	FIELD	o		String	Comment entered by the user. Maximum possible length is 250 characters.
	type	FIELD	m		Enum	Order type. Valid values: **"ORDER_TYPE_O":** Regular limit order (for all predefined contracts). **"ORDER_TYPE_I":** Iceberg order. **"ORDER_TYPE_B":** User defined block order.
	client_order_id	FIELD	o		String	Client Order Id with a maximum length of 40 characters.
	delivery_area_id	FIELD	m		String	Defines the delivery area of the order.The delivery_area_id is respecting codes provided by DeliveryAreaInfoRprt.

Message/Field		Type	m/o	No.	Data Type	Short description
	order_execution_restriction	FIELD	o		Enum	Execution restriction of the order. Valid values: **"ORDER_EXECUTION_RESTRICTION_TYPE_NON":** No restriction. This is the default. **"ORDER_EXECUTION_RESTRICTION_TYPE_FOK"** (Fill or Kill): The order is immediately fully executed or deleted. **"ORDER_EXECUTION_RESTRICTION_TYPE_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. **"ORDER_EXECUTION_RESTRICTION_TYPE_AON"** (All or None): The order must be filled completely or not at all. The order stays in the order book until it is executed or removed by the system or user. This execution restriction can be used only in combination with User Defined Block Orders (for which only ORDER_EXECUTION_RESTRICTION_TYPE_AON execution restriction is allowed).
	quantity	FIELD	m		Integer	Contains the total quantity of the order. In case of an Iceberg order this field corresponds to the hidden quantity + display quantity.
	display_quantity	FIELD	o		Integer	Used to define display quantity of an Iceberg Order. This field is required only in the case of type='ORDER_TYPE_I'.
	price	FIELD	o		Integer(64)	Limit price of the order in currency defined by contracts. Value is multiplied by 100, e.g. 1 Euro = 100.
	side	FIELD	m		Enum	Defines on which side of the market the order is entered (**"DIRECTION_TYPE_BUY"**, **"DIRECTION_TYPE_SELL"**).
	product_name	FIELD	o		String	Product identifier. Required in case of the (long name) is omitted.
	contract	FIELD	o		String	Contract code identifier (long name). Applicable for orders for pre-defined contracts only.
	delivery_start	FIELD	o		Timestamp	Start of delivery of the underlying contract. Applicable for User Defined Block Orders only. The field is ignored if *contract* is filled in.
	delivery_end	FIELD	o		Timestamp	End of delivery of the underlying contract. Applicable for User Defined Block Orders only. The field is ignored if *contract* is filled in.
	peak_price_delta	FIELD	o		Integer(64)	Peak price delta for Iceberg orders. • The peak_price_deltaof buy orders must be smaller or equal than zero. • The peak_price_deltaof sell orders must be greater or equal than zero. If it is omitted the system will assume a value of "0,00".

Table 11 - Add order entry request message structure

2.8.2.2. Order Modify Request (ModifyOrderReq)

ModifyOrderReq	
Type:	Management Request
Roles:	EmtasImIns
Routing Keys:	`market.request.management`

The modification message for one or more bids. The maximum number of bids within a single message is 25. In the case of an bid activation or deactivation request in the XBID market, modification of other attributes is disabled. The message must be encapsulated and signed using the SignedMessage message, see chapt. 3 Using the electronic signature.

Message/Field		Type	m/o	No.	Data Type	Short description
ModifyOrderReq		MSG	m	1	Structure	
standard_header		*FIELD*	m		*Structure*	*Standard header of each message. Please see chapter 2.6.7 Standard message header.*
modify_order_type		FIELD	m		Enum	Offers the possibility to activate, deactivate, modify or delete all orders contained in the basket. **“MODIFY_ORDER_TYPE_ACTI”**: Activate all orders contained in this basket. Already active orders are ignored. **“MODIFY_ORDER_TYPE_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. **“MODIFY_ORDER_TYPE_MODI”**: Modifies all orders in the basket. **“MODIFY_ORDER_TYPE_DELE”**: Deletes all orders in the basket.
orders		FIELD	m	1..25	Structure	List of single order definitions.
	revision_no	FIELD	m		Integer(64)	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.
	validity_restriction	FIELD	o		Enum	Validity restriction of the order. If this field is omitted, the order will be treated as a “Good for Session” order. Valid values: **“VALIDITY_RESTRICTION_TYPE_ 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. **“VALIDITY_RESTRICTION_TYPE_ GTD”** (Good till date): The order rests in the order book until the date specified in the validityDate field. **“VALIDITY_RESTRICTION_TYPE_ NON”** (No validity restriction): Mandatory for orders with the execution restriction “ORDER_EXECUTION_RESTRICTION_TYPE_FOK” or “ORDER_EXECUTION_RESTRICTION_TYPE_IOC”.
	validity_date	FIELD	o		Timestamp	This field is mandatory in case of validity_restriction equals “VALIDITY_RESTRICTION_TYPE_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	FIELD	m		Enum	Order type. Valid values: **“ORDER_TYPE_O”:** Regular limit order (for all predefined contracts). **“ORDER_TYPE_I”:** Iceberg order. **“ORDER_TYPE_B”:** User defined block order. Order type cannot be changed by modification to or from order type “ORDER_TYPE_B”.
	text	FIELD	o		String	Comment entered by the user. Maximum possible length is 250 characters.

Message/Field		Type	m/o	No.	Data Type	Short description
	order_execution_restriction	FIELD	o		Enum	Execution restriction of the order. Valid values: **“ORDER_EXECUTION_RESTRICTION_TYPE_NON”:** No restriction. This is the default. **“ORDER_EXECUTION_RESTRICTION_TYPE_FOK”** (Fill or Kill): The order is immediately fully executed or deleted. **“ORDER_EXECUTION_RESTRICTION_TYPE_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. **“ORDER_EXECUTION_RESTRICTION_TYPE_AON”** (All or None): The order must be filled completely or not at all. The order stays in the order book until it is executed or removed by the system or user. This execution restriction can be used only in combination with User Defined Block Orders (for which only ORDER_EXECUTION_RESTRICTION_TYPE_AON execution restriction is allowed).
	quantity	FIELD	m		Integer	Contains the total quantity of the order. In case of an Iceberg order this field corresponds to the hidden quantity + display quantity.
	display_quantity	FIELD	o		Integer	Used to define display quantity of an Iceberg Order.
	price	FIELD	o		Integer(64)	Limit price of the order in currency defined by contract. Value is multiplied by 100, e.g. 1 Euro = 100.
	client_order_id	FIELD	o		String	client_order_id with a maximum length of 40 characters.
	order_id	FIELD	m		Integer(64)	Order Id as returned by the CS OTE system. This value is used to identify the order to be modified.
	peak_price_delta	FIELD	o		Integer(64)	Peak price delta for Iceberg orders. • The peak_price_delta of buy orders must be smaller or equal than zero. • The p peak_price_delta pd of sell orders must be greater or equal than zero. If it is omitted the system will assume a value of “0,00”.

Table 12 - Order modify message structure

2.8.2.3. Order Request (OrderReq)

OrderReq	
Type:	Inquiry Request
Roles:	EmtasImTsAcc
Routing Keys:	`market.request.inquiry`
Request Limits:	10/30

A custom bid status request.

Message/Field	Type	m/o	No.	Data Type	Short description
OrderReq	MSG	m	1	Structure	
standard_header	*FIELD*	m		Structure	*Standard header of each message. Please see chapter 2.6.7 Standard message header.*
contracts	FIELD	o	0..1000	String	List of contract codes (long name). If no contract code is given, the own orders for all contracts assigned to the requesting user are returned.

Table 13 - Order request message structure

2.8.2.4. Order Execution Report (OrderExecutionRprt)

OrderExecutionRprt	
Type:	Management Response; Broadcast
Response to:	AddOrderReq; ModifyOrderReq; OrderReq; ModifyAllOrdersReq; (sent to the user-generated private response queue or a broadcast to `market.broadcastQueue.<login-id>`)
Broadcast:	Yes
Broadcast Routing Keys:	`<product_name>.PRTC_<partic_id>`
Roles:	EmtasImTsAcc

A message indicating a successful bid modification. This message is sent to market participants in the following cases:

- A successful bid submission,
- A successful bid modification,
- A partially or fully traded bid,
  As a response to an bid request (in this case, it is sent to the private response queue, in all other cases it is sent to the mass message queue).

Message/Field		**Type**	**m/o**	**No.**	**Data Type**	**Short description**
OrderExecutionRprt		MSG	m	1	Structure	
standard_header		*FIELD*	*m*		*Structure*	*Standard header of each message. Please see chapter 2.6.7 Standard message header.*
orders		FIELD	o	0..n	Structure	
	action	FIELD	m		Enum	Code of the last action provided on the order. Valid values are: **“ORDER_ACTION_TYPE_UADD”**: Order added by user. **“ORDER_ACTION_TYPE_UHIB”**: Order hibernated by user. **“ORDER_ACTION_TYPE_UMOD”**: Order modified by user. **“ORDER_ACTION_TYPE_UDEL”**: Order deleted by user. **“ORDER_ACTION_TYPE_SHIB”**: Order hibernated by the system. **“ORDER_ACTION_TYPE_SMOD”**: Order modified by the system. **“ORDER_ACTION_TYPE_SDEL”**: Order deleted by the system. **“ORDER_ACTION_TYPE_FEXE”**: Order is fully executed. If an order comes into the system and gets executed immediately by matching an already existing order only one OrderExecutionRprt for this order is sent with action ORDER_ACTION_TYPE_FEXE or ORDER_ACTION_TYPE_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 ORDER_ACTION_TYPE_UADD and later one for the execution with either ORDER_ACTION_TYPE_FEXE or ORDER_ACTION_TYPE_PEXE. **“ORDER_ACTION_TYPE_PEXE”**: Partial execution of order. **“ ORDER_ACTION_TYPE_IADD”**: A new slice of an Iceberg order was added to the service.
	validity_restriction	FIELD	o		Enum	Validity restriction of the order. If this field is omitted, the order will be treated as a “Good for Session” order. Valid values: **“VALIDITY_RESTRICTION_TYPE_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. **“VALIDITY_RESTRICTION_TYPE_GTD”** (Good till date): The order rests in the order book until the date specified in the vldtyDate field. **“VALIDITY_RESTRICTION_TYPE_NON”** (No validity restriction): Mandatory for orders with the execution restriction “ORDER_EXECUTION_RESTRICTION_TYPE_FOK” or “ORDER_EXECUTION_RESTRICTION_TYPE_IOC”.

Message/Field		Type	m/o	No.	Data Type	Short description
	validity_date	FIELD	o		Timestamp	This field is mandatory in case of validity_restriction equals “VALIDITY_RESTRICTION_TYPE_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.
	timestamp	FIELD	m		Timestamp	Timestamp of the order entry as determined by the CS OTE system. This timestamp determines the execution priority in case of identical limit prices.
	revision_no	FIELD	m		Integer(64)	This value is increased in case of a partial execution, hibernation, modification without execution priority change.
	user_code	FIELD	m		String	User code of the user who entered the order.
	state	FIELD	m		Enum	The current state of the order in the system. Valid values: **“ORDER_STATE_TYPE_HIBE”:** The order is entered into the XBID SOB system but not exposed to the market. **“ORDER_STATE_TYPE_ACTI”:** The order is entered and immediately exposed to the market for execution **“ORDER_STATE_TYPE_IACT”:** The order is inactive due time validity or fully executed. “**ORDER_STATE_TYPE_DELE**”: The order is deleted
	type	FIELD	m		Enum	Order type. Valid values: **“ORDER_TYPE_O”:** Regular limit order (for all predefined contracts). **“ORDER_TYPE_I”:** Iceberg order. **“ORDER_TYPE_B”:** User defined block order.
	client_order_id	FIELD	o		String	LTS Order Id with a maximum length of 40 characters. This value is not modified by the CS OTE system and may be used by LTS to identify orders.
	delivery_area_id	FIELD	m		String	Defines the delivery area of the order.The delivery_area_id is respecting codes provided by DeliveryAreaInfoRprt.
	text	FIELD	o		String	Comment entered by the user. Maximum possible length is 250 characters.
	order_execution_restriction	FIELD	o		Enum	Execution restriction of the order. Valid values: **“ORDER_EXECUTION_RESTRICTION_TYPE_FOK”** (Fill or Kill): The order is immediately fully executed or deleted. **“ORDER_EXECUTION_RESTRICTION_TYPE_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. **“ORDER_EXECUTION_RESTRICTION_TYPE_NON”:** No restriction. **“ORDER_EXECUTION_RESTRICTION_TYPE_AON”** (All or None): The order must be filled completely or not at all. The order stays in the order book until it is executed or removed by the system or user. ORDER_EXECUTION_RESTRICTION_TYPE_AON execution restriction can be used only in combination with User Defined Block Orders (for which only ORDER_EXECUTION_RESTRICTION_TYPE_AON execution restriction is allowed) and hence can’t be changed by modification.
	initial_quantity	FIELD	m		Integer	The total quantity entered with this order. If the order is partially matched, the initial_quantity still contains the original quantity value.
	quantity	FIELD	m		Integer	Contains the quantity exposed to the market. In case of an Iceberg Order this is the rest of the display quantity.
	hidden_quantity	FIELD	o		Integer	Contains the hidden quantity of the Iceberg order. The total executable quantity may be calculated by adding the hidden_quantity to the quantity.

Message/Field	Type	m/o	No.	Data Type	Short description
display_quantity	FIELD	o		Integer	Used to define display quantity of an Iceberg Order.
price	FIELD	o		Integer(64)	Limit price of the order in currency defined by contract. Value is multiplied by 100, e.g. 1 Euro = 100.
side	FIELD	m		Enum	Defines on which side of the market the order is entered. Valid values: **“DIRECTION_TYPE_BUY”:** Buy order. **“DIRECTION_TYPE_SELL”**: Sell order.
contract	FIELD	m		String	Contract code identifier (long name).
initial_order_id	FIELD	m		Integer(64)	In case of an order modification, this value contains the Id of the first order in the modification chain.
parent_order_id	FIELD	o		Integer(64)	In case of an order modification this field contains the Id of the modified order.
order_id	FIELD	m		Integer(64)	Order Id as returned by the CS OTE system.
last_update_user_info	FIELD	m		String	Information about the user who last updated the order
peak_price_delta	FIELD	o		Integer(64)	Peak price delta for Iceberg orders.

Table 14 - Order execution report message structure

2.8.2.5. Modify All Orders Request (ModifyAllOrdersReq)

ModifyAllOrdersReq	
Type:	Management Request
Roles:	EmtasImIns
Routing Keys:	`market.request.management`

A message for a mass activation, deactivation and bid cancellation. The message must be encapsulated and signed using the SignedMessage, see chapt. 3 Using the electronic signature.

Message/Field	Type	m/o	No.	Data Type	Short description	
ModifyAllOrdersReq	MSG			Structure		
standard_header	*FIELD*	*m*		*Structure*	*Standard header of each message. Please see chapter 2.6.7 Standard message header.*	
partic_id	FIELD	o		String	Unique identifier of a partic.	One and only one of these fields must be supplied.
user_id	FIELD	o		Integer	Unique identifier of a user.	
modify_order_type	FIELD	m		Enum	Modification type for the orders: **“MODIFY_ORDER_ALL_TYPE_ACTI”**: Activate all orders. Already active orders are ignored. **“MODIFY_ORDER_ALL_TYPE__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. **“MODIFY_ORDER_ALL_TYPE_DELE”**: Deletes all orders.	
product_names	FIELD	o	0.100	String	Only orders for products with the given product names will be modified.	
delivery_area_ids	FIELD	o	0..100	String	Orders for the given user_id and list of DA’s in delivery_area_id will be Deactivated or Deleted. This field can only be supplied when user_id is provided in the message. The delivery_area_id is respecting codes provided by DeliveryAreaInfoRprt.	
contracts	FIELD	o	0.. 1000	String	List of contract codes (long name). If no contract code is given, the own orders for all contracts assigned to the specified participant or user are changed.	

Table 15 - Modify all orders request message structure

2.8.3. IM Trade management

2.8.3.1. Trade Recall Request (TradeRecallReq)

TradeRecallReq	
Type:	Management Request
Roles:	EmtasImIns
Routing Keys:	`market.request.management`

This message is used to submit a trade recall request. The request may only be submitted by the participant who owns an bid on at least one side of the trade, for both national and international (cross-border) trades. It contains only an identifier and a trade version.

Message/Field	Type	m/o	No.	Data Type	Short description
TradeRecallReq	MSG	m		Structure	
standard_header	*FIELD*	*m*		*Structure*	*Standard header of each message. Please see chapter 2.6.7 Standard message header.*
trade_id	FIELD	m		Integer(64)	Trade Id of the trade to be recalled.
revision_no	FIELD	m		Integer(64)	The latest revision number of the trade must be provided by the MP. In case the OTE-COM system has another revision number, it will reject the request with an ErrResp.

Table 16 - Trade recall request message structure

Note: The trade recall communication scenario is not available.

2.8.4. Market information

2.8.4.1. Public Order Books Request (PublicOredrBooksReq)

PublicOrderBooksReq	
Type:	Inquiry Request
Roles:	<All>
Routing Keys:	`market.request.inquiry`
Request Limits:	10/40

A public order book request for the specified contract.

Message/Field	Type	m/o	No.	Data Type	Short description
PublicOrderBooksReq	MSG		1	Structure	
standard_header	*FIELD*	*m*		*Structure*	*Standard header of each message. Please see chapter 2.6.7 Standard message header.*
contract_type	FIELD	(m)		Enum	Defines which kind of contracts should be retrieved: Possible values are: **“CONTRACT_TYPE_ ALL”** – All kind of contracts (pre-defined and user-defined) **“CONTRACT_TYPE_ PDC”** – Only pre-defined contracts **“CONTRACT_TYPE_ UDC”** – Only user-defined contracts This field is ignored when contracts field is specified.
product_names	FIELD	(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.
contracts	FIELD	(m)	0.. 1000	String	List of contract codes (long name). **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.
delivery_area_ids	FIELD	O	0.. 1000	String	List of delivery areas for which the order book(s) should be retrieved.

Table 17 - Public order books request message structure

2.8.4.2. Public Order Books Response (PublicOrderBooksResp)

PublicOrderBooksResp	
Type:	Inquiry Response
Response to:	PublicOrderBooksReq (sent to the user-generated private response queue)
Broadcast:	No
Broadcast Routing Keys:	---
Roles:	`<All>`

Public information about the current bids for the specified contract. The message is distributed as a response to the *PublicOrderBooksReq* request.

Message/Field			Type	m/o	No.	Data Type	Short description
PublicOrderBooksResp			MSG	m	1	Structure	
standard_header			*FIELD*	*m*		*Structure*	*Standard header of each message. Please see chapter 2.6.7 Standard message header.*
order_books			FIELD	o	0..n	Structure	
	revision_no		FIELD	m		Integer(64)	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. OTE system checks gap detection (missing number in the XBID revision sequence) for every order book of contract separately. If a gap is detected (e.g. due to network interruption), the OTE system will perform an initialization process for the order book and the revision number is reset to 0. The initialization process is performed only for the order book with the detected gap.
	contract		FIELD	m		String	Contract code identifier (long name).
	delivery_area_id		FIELD	m		String	Delivery Area to which the attached order books refer to.
	last_price		FIELD	o		Integer(64)	Last traded price.
	price_direction		FIELD	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
	last_quantity		FIELD	o		Integer	Last traded quantity.
	total_quantity		FIELD	o		Integer(64)	The total quantity traded during this trading session.
	last_trade_time		FIELD	o		Timestamp	Timestamp of the last execution.
	high_price		FIELD	o		Integer(64)	Highest traded price since the start of the trading period.
	low_price		FIELD	o		Integer(64)	Lowest traded price since the start of the trading period.
	sell_orders		FIELD	o	0..n	Structure	
		order_id	FIELD	m		Integer(64)	Order Id as determined by the CS OTE system.
		quantity	FIELD	m		Integer(64)	The quantity of the order which is exposed in that delivery area.
		price	FIELD	m		Integer(64)	Limit price of the order in currency defined by contract. Value is multiplied by 100, e.g. 1 Euro = 100.
		order_entry_time	FIELD	m		Timestamp	Timestamp of the order.
		order_execution_restriction	FIELD	o		Enum	Execution restriction of the order. This field is set only in case of AON orders. **"ORDER_EXECUTION_RESTRICTION_TYPE_AON"**: AON Order.
		order_type	FIELD	o		Enum	**"ORDER_TYPE_ O":** Regular limit order. **"ORDER_TYPE_ I":** Iceberg order.

Message/Field			Type	m/o	No.	Data Type	Short description
							"ORDER_TYPE_ B": User defined block order.
	buy_orders		FIELD	0	0..n	Structure	
		order_id	FIELD	m		Integer(64)	Order Id as determined by the CS OTE system.
		quantity	FIELD	m		Integer(64)	The quantity of the order which is exposed in that delivery area.
		price	FIELD	m		Integer(64)	Limit price of the order in currency defined by contract. Value is multiplied by 100, e.g. 1 Euro = 100.
		order_entry_tim e	FIELD	m		Timestamp	Timestamp of the order.
		order_execution _restriction	FIELD	o		Enum	Execution restriction of the order. This field is set only in case of AON orders. "**ORDER_EXECUTION_RESTRICTION_TYPE_AON**": AON Order.
		order_type	FIELD	o		Enum	**"ORDER_TYPE_ O"**: Regular limit order. **"ORDER_TYPE_ I"**: Iceberg order. **"ORDER_TYPE_ B"**: User defined block order.

Table 18 - Public order books report message structure

2.8.4.3. Public Order Books Delta Report (PublicOrderBooksDeltaRprt)

PublicOrderBooksDeltaRprt	
Type:	Broadcast
Response to:	n/a
Broadcast:	Yes
Broadcast Routing Keys:	`<product_name>.<delivery_area>`
Roles:	`<All>`

The *PublicOrderBooksDeltaRprt* message is sent when an active bid is implemented or modified. This message includes all bids that have been modified since the previous distribution of the *PublicOrderBooksDeltaRprt* for the specified contract.

The message format is the identical to that of the *PublicOrderBooksResp* message.

2.8.4.4. Message Request (MessageReq)

MessageReq	
Type:	Inquiry Request
Roles:	<ALL>
Routing Keys:	`market.request.inquiry`
Request Limits:	2/10

A request for trading system messages, that were created in the trading system in the past. It is possible to request up to one day prior.

Message/Field	Type	m/o	No.	Data Type	Short description
MessageReq	MSG		1	Structure	
standard_header	*FIELD*	*m*		*Structure*	*Standard header of each message. Please see chapter 2.6.7 Standard message header.*
type	FIELD	m		Enum	Defines what kinds of messages are returned, allowing filtering the messages on a request level. Valid Values: **"MESSAGE_TYPE_ALL":** Return all messages. **"MESSAGE_TYPE_PUBLIC":** Return only public messages. **"MESSAGE_TYPE_PRIVATE":** Return only private messages.
end_date	FIELD	m		Timestamp	Timestamp defining to which point in time the messages should be retrieved.

Message/Field	Type	m/o	No.	Data Type	Short description
start_date	FIELD	m		Timestamp	Timestamp defining from which point in time the messages should be retrieved. It is possible only to retrieve messages from the last 1 day.

Table 19 - Message request message structure

2.8.4.5. Message Report (MessageRprt)

MessageRprt	
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_<partic_id>`
	`<product_name>`
	`<product_name>.PRTC_<partic_id>`
	`public`

Roles:	`<All>`

Messages from the trading system are sent in response to the *MessageReq* and subsequently distributed whenever a new message is created in the trading system.

Message/Field		Type	m/o	No.	Data Type	Short description
MessageRprt		MSG	m	1	Structure	
standard_header		*FIELD*	*m*		*Structure*	*Standard header of each message. Please see chapter 2.6.7 Standard message header.*
messages		FIELD	o	0..n	Structure	
	message_id	FIELD	m		Integer(64)	The message Id as assigned by the CS OTE system.
	type	FIELD	m		Enum	Defines the message type. Valid Values: **"MESSAGE_TYPE_PUBLIC":** The message is a public message. **"MESSAGE_TYPE_PRIVATE":** The message is a private message.
	contract	FIELD	o		String	Related underlying contract (if any).
	message_code	FIELD	m		Integer	Message code of the message
	timestamp	FIELD	m		Timestamp	Timestamp of the message as assigned by the CS OTE system.
	severity	FIELD	m		Enum	Severity of the message: **"MESSAGE_SEVERITY_TYPE_URG":** Urgent message. **"MESSAGE_SEVERITY_TYPE_ERR":** Error. **"MESSAGE_SEVERITY_TYPE_HIG":** High prioritized message. **"MESSAGE_SEVERITY_TYPE_MED":** Medium prioritized message. **"MESSAGE_SEVERITY_TYPE_LOW":** Low priority message.
	market_supervision_message	FIELD	m		Boolean	Determines if the message has been send by market supervision
	text_en	FIELD	m		String	Message text. – English version.
	text_cz	FIELD	m		String	Message text – Czech version.
	sell_delivery_area_id	FIELD	o		String	In case of an order execution, this field contains the delivery area of the sell side.
	buy_delivery_area_id	FIELD	o		String	In case of an order execution, this field contains the delivery area of the buy side.

Table 20 - Message report message structure

2.8.4.6. Trade Capture Request (TradeCaptureReq)

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

A custom trades request. It is possible to request data up to seven days prior with a maximum date range of 24 hours. If the input parameters are invalid, the *ErrResp* is returned.

Message/Field	Type	m/o	No.	Data Type	Short description
TradeCaptureReq	MSG				
standard_header	*FIELD*	*m*		*Structure*	*Standard header of each message. Please see chapter 2.6.7 Standard message header.*
start_date	FIELD	m		Timestamp	Start of the period for which the trades are retrieved. This value must fulfil the following conditions: • end_date – start_date <= 24 hours
end_date	FIELD	o		Timestamp	End of the period for which the trades are retrieved. The following condition must be fulfilled: • end_date – start_date <= 24 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 difference is bigger than max value.

Table 21 - Trade capture request message structure

2.8.4.7. 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.<product_name>.PRTC_<partic_id>`
Roles:	EmtasImTsAcc

A message that contains information about a trade creation or modification. It is sent to both participants of the specified trade (or only to one in the case of an international trade). For each receipient only the relevant part of the trade is included. This message is also sent in response to *TradeCaptureReq* and *TradeRecallReq*.

This message is also distributed in the following cases:

- A request for a cross-border CZ trade callback initiated from a different country, which modifies the trade status,
- A trade cancellation initiated by the central XBID side.

Message/Field		Type	m/o	No.	Data Type	Short description
TradeCaptureRprt		MSG	m	1	Structure	
standard_header		*FIELD*	*m*		*Structure*	*Standard header of each message. Please see chapter 2.6.7 Standard message header.*
trades		FIELD	o	0..n	Structure	
	trade_id	FIELD	m		Integer(64)	Trade ID of the trade.
	revision_no	FIELD	m		Integer(64)	Revision number of this trade. With every change of the trade the revision number is increased by one.

Message/Field	Type	m/o	No.	Data Type	Short description
state	FIELD	m		Enum	Current state of the trade. Valid values are: **"TRADE_STATE_TYPE_ACTI":** Trade is active (this is the default value). **"TRADE_STATE_TYPE_CNCL":** Trade was cancelled. **"TRADE_STATE_TYPE_RREQ":** Recall of this trade was requested. **"TRADE_STATE_TYPE_RREJ":** Recall request was rejected - trade is still valid then. **"TRADE_STATE_TYPE_RGRA":** Recall request was granted – trade has been recalled.
contract	FIELD	m		String	Contract code (long name).
quantity	FIELD	m		Integer	Executed quantity.
price	FIELD	m		Integer(64)	Execution price in currency defined by contract. Value is multiplied by 100, e.g. 1 Euro = 100.
execution_time	FIELD	m		Timestamp	Execution date as assigned by system.
latest_recall_process_time	FIELD	o		Timestamp	Informs until when a recall request can be processed.
recall_req_time	FIELD	o		Timestamp	Date and time of a recall request.
recall_granted_time	FIELD	o		Timestamp	Date and time when the recall was granted or when the trade was cancelled.
recall_rejected_time	FIELD	o		Timestamp	Date and time when the recall was rejected.
contract_phase	FIELD	m		Enum	**“CONTRACT_PHASE_TYPE_CONT”**: The trading in the contract is in continuous mode. **"CONTRACT_PHASE_TYPE_AUCT"**: The trade results from an Intraday Micro Auction.
buy	FIELD	o	0..1	Structure	
order_id	FIELD	m		Integer(64)	Order Id of the buy side order.
delivery_area_id	FIELD	m		String	Delivery Area to which the attached order books refer to.
partic_id	FIELD	m		String	Participant who entered the buy side order.
user_code	FIELD	m		String	User code of the user who entered the buy side order.
client_order_id	FIELD	o		String	Client’s identification of order.
text	FIELD	o		String	Text of the buy side order.
sell	FIELD	o	0..1	Structure	
order_id	FIELD	m		Integer(64)	Order Id of the sell side order.
delivery_area_id	FIELD	m		String	Delivery Area to which the attached order books refer to.
partic_id	FIELD	m		String	Participant who entered the sell side order.
user_code	FIELD	m		String	User code of the user who entered the sell side order.
client_order_id	FIELD	o		String	Client’s identification of order.
text	FIELD	o		String	Text of the sell side order.

Table 22 - Trade capture report message structure

2.8.4.8. Public Trade Confirmation Request (PublicTradeConfirmationReq)

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

A request for public information about established trades. It is possible to request data up to 7 days prior with a maximum date range of 24 hoursIf the input parameters are invalid, the *ErrResp* is returned.

Message/Field	Type	m/o	No.	Data Type	Short description
PublicTradeConfirmationReq	MSG	m		Structure	
standard_header	*FIELD*	*m*		*Structure*	*Standard header of each message. Please see chapter 2.6.7 Standard message header.*
start_date	FIELD	m		Timestamp	Start of the period for which the trades are retrieved. This value must fulfil the following conditions: • end_date – start_date <= 24 hours
end_date	FIELD	o		Timestamp	End of the period for which the trades are retrieved. The following condition must be fulfilled: • end_date – start_date <= 24 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 difference is bigger than max value.
product_names	FIELD	o	0..1000	String	List of product names for which the public trade confirmations are requested. If not supplied all products for which the user has access rights are returned

Table 23 - Public trade confirmation request message structure

2.8.4.9. Public Trade Confirmation Report (PublicTradeConfirmationRprt)

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

A message containing a trade establishment. This message is distributed to all users assigned to the contract relevant to the established trade. It is also sent in response to *PublicTradeConfirmationReq*.

Message/Field		Type	m/o	No.	Data Type	Short description
PublicTradeConfirmationRprt		MSG	m	1	Structure	
standard_header		*FIELD*	*m*		*Structure*	*Standard header of each message. Please see chapter 2.6.7 Standard message header.*
trades		FIELD	o	0..n	Structure	
	trade_id	FIELD	m		Integer(64)	Trade Id of the underlying trade.
	revision_no	FIELD	m		Integer(64)	Revision number of the trade. This is increased by one every time the trade is changed.
	state	FIELD	m		Enum	Current state of the trade. Valid values are: **"TRADE_STATE_TYPE_ACTI":** Trade is active (this is the default value). **"TRADE_STATE_TYPE_CNCL":** Trade was cancelled. **"TRADE_STATE_TYPE_RREJ":** Recall request was rejected - trade is tsill valid. **"TRADE_STATE_TYPE_RGRA":** Recall request was granted – trade has been recalled. **"TRADE_STATE_TYPE_RREQ":** Recall of this trade was requested.
	contract	FIELD	m		String	Contract code (long name) of the trade.

	Message/Field	Type	m/o	No.	Data Type	Short description
	price	FIELD	m		Integer(64)	Execution price in currency defined by contract. Value is multiplied by 100, e.g. 1 Euro = 100.
	quantity	FIELD	m		Integer	Traded quantity.
	trade_execution_time	FIELD	m		Timestamp	Trade execution time.

Table 24 - Public trade confirmation report message structure

2.8.4.10. Contract Information Request (ContractInfoReq)

ContractInfoReq	
Type:	Inquiry Request
Roles:	<ALL>
Routing Keys:	`market.request.inquiry`
Request Limits:	10/40

A contract request. It is possible to request data up to 7 days prior. If the input parameters are invalid, *ErrResp* is returned in the response.

Message/Field	Type	m/o	No.	Data Type	Short description
ContractInfoReq	MSG			Structure	
standard_header	*FIELD*	*m*		*Structure*	*Standard header of each message. Please see chapter 2.6.7 Standard message header.*
start_date	FIELD	o		Timestamp	Start date for which the contract information is requested. Notes: • if contract field is specified this field is ignored • if product_names field is specified or neither contract nor product_names fields are specified, this field becomes mandatory.
end_date	FIELD	o		Timestamp	End date for which the contract information is requested. Notes: • if contract field is specified this field is ignored • if product_names field is specified or neither contract nor product_names fields are specified, this field becomes mandatory.
product_names	FIELD	o	0..1000	String	The contract information for all contratcs belonging to products with given product names is requested. If product_names field is specified, the contract field cannot be specified and the start_date and end_date fields are mandatory.
contract	FIELD	o	0..1	String	Contract code (long name). If contract is specified, the products field cannot be specified and the start_date and end_date fields are ignored.

Table 25 - Contract information request message structure

2.8.4.11. 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:	`<product_name>`
Roles:	EmtasImTsAcc

A contract information. This message is distributed whenever any contract attribute is modified or in response to the *ContractInfoReq* request.

Message/Field		Type	m/o	No.	Data Type	Short description
ContractInfoRprt		MSG	m	1	Structure	
standard_header		*FIELD*	*m*		*Structure*	*Standard header of each message. Please see chapter 2.6.7 Standard message header.*
	contracts	FIELD	o	0..n		
	contract_id	FIELD	m		Integer	Contract ID.
	revision_no	FIELD	m		Integer(64)	Revision number of the contract.
	product_name	FIELD	m		String	Underlying product.
	product_revision_no	FIELD	m		Integer(64)	Revision number of the underlying product.
	name	FIELD	m		String	Contract name. This is used for display purposes.
	long_name	FIELD	m		String	Contract long name, containing additional information.
	delivery_start	FIELD	m		Timestamp	Start of delivery.
	delivery_end	FIELD	m		Timestamp	End of delivery.
	duration	FIELD	o		Double	The duration of the contract in full hours. For quarterly contracts the value would be 0.25. An hourly contract would have 1.0. An block contract would have value in interval from 2 to 24 (or 23/25 in case of short/long clock change).
	predefined	FIELD	m		Boolean	Flag that indicates, if a contract has been automatically created by the system or if the contract was generated with an entry of a user-defined block order. 1 = automatically generated, 0= user defined
	state	FIELD	m		Enum	Current state of the contract. The following values are allowed: **"CONTRACT_STATE_TYPE_HIBE":** Hibernated, the contract was manually deactivated by Central Admin. **"CONTRACT_STATE_TYPE_ISSUED":** The contract is issued, but not available for trading. **"CONTRACT_STATE_TYPE_OPEN":** Contract is active and available for trading. **"CONTRACT_STATE_TYPE_CLOSE"**: Contract is closed and not available for trading. **"CONTRACT_STATE_TYPE_TERM"**: Contract is terminated and not available for trading.
	trading_phase_start	FIELD	m		Timestamp	Start date and time of the current/next trading phase.
	trading_phase_end	FIELD	o		Timestamp	End date and time of the current/next trading phase.

Table 26 - Contract information report message structure

2.8.4.12. Product Information Request (ProductInfoReq)

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

A detailed product information request.

Message/Field	Type	m/o	No.	Data Type	Short description
ProductInfoReq	MSG	m	1	Structure	
standard_header	*FIELD*	*m*		*Structure*	*Standard header of each message. Please see chapter 2.6.7 Standard message header.*
product_names	FIELD	o	0.. 1000	String	List of product names.

Table 27 - Product information request message structure

2.8.4.13. Product Information Report (ProductInfoRprt)

ProductInfoRprt	
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:	`<product_name>`
Roles:	EmtasImTsAcc

A detailed product information as a response to the *ProductInfoReq*.

Since *ContractInfoReq* allows requests up to 7 days prior (see also 2.8.4.10 Contract Information Request (ContractInfoReq)), this response returns only product revisions that even the latest contract can reference.

The response may also include multiple product versions with the same name, where the unique identifier is the product name combined with the revision number.

Message/Field			Type	m/o	No.	Data Type	Short description
ProductInfoRprt			MSG	m	1	Structure	
standard_header			*FIELD*	*m*		*Structure*	*Standard header of each message. Please see chapter 2.6.7 Standard message header.*
products			FIELD	o	0..n	Structure	
	product_name		FIELD	m		String	Unique identifier name of the product.
	display_name		FIELD	m		String	String used to display the product.
	currency		FIELD	m		String	The currency of the product. The value is always “EUR”.
	revision_no		FIELD	m		Integer(64)	Revision number of the product. This value is increased by one every time the product is modified by the system.
	quantity_unit		FIELD	m		String	Defines the quantity unit.
	min_quantity		FIELD	o		Integer	Minimal display quantity.
	decimal_shift_quantity		FIELD	m		Integer	Decimal shift of the quantity information. A value of 2 results in a display of 100 kW.
	max_quantity		FIELD	m		Integer	Maximal allowed quantity for orders entered in contracts belonging to this product.
	min_price		FIELD	m		Integer(64)	Minimal price allowed for orders entered in contracts belonging to this product.
	max_price		FIELD	m		Integer(64)	Maximal price allowed for orders entered in contracts belonging to this product.
	decimal_shift_price		FIELD	m		Integer	Decimal shift of the price information. A value of 2 results in a display in Eurocents.
	contract_name_pattern		FIELD	o		String	Format string for the contract name.
	tick_size		FIELD	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.
	lot_size		FIELD	m		Integer	Defines the smallest tradable unit of the product.
	product_configurations		FIELD	o	0..n	Structure	
		key	FIELD	m		String	Exchange specific product attribute names (e.g.: blockOrderProduct, icebergMinPeakSize, icebergPriceDeltaRange)
		value	FIELD	m		String	Exchange specific product attribute values

Table 28 - Product information report message structure

2.8.4.14. Market State Request (MarketStateReq)

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

A current market status request. The required market is specified in the message header *standard_header*.

Message/Field	Type	m/o	No.	Data Type	Short description
MarketStateReq	MSG	m	1	Structure	
standard_header	*FIELD*			*Structure*	*Standard header of each message. Please see chapter 2.6.7 Standard message header.*

Table 29 - Market state request message structure

2.8.4.15. Market State Report (MarketStateRprt)

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

Current information about the market trading status. This message is distributed whenever the market status is modified and in response to the *MarketStateReq* request.

Message/Field	Type	m/o	No.	Data Type	Short description
MarketStateRprt	MSG	m	1	Structure	
standard_header	*FIELD*	*m*		*Structure*	*Standard header of each message. Please see chapter **Error! Reference source not found. Error! Reference source not found.**.*
state	FIELD	m		Enum	Contains the current market state. The following values are allowed: **"MARKET_STATE_TYPE_HIBE":** Hibernated; no trading is possible and order books are empty. Done on WebGui by Admin. **"MARKET_STATE_TYPE_ACTI":** Market is active and trading is possible.
connected_xbid	FIELD	o		Enum	State identification of physical connection to XBID solution. **"CONNECTED_XBID_TYPE_ACTI"** – Connection to XBID solution is valid. **"CONNECTED_XBID_TYPE_DISC"** – Disconnected from XBID solution. Used only for market_id "MARKET_ID_TYPE_XBID".
trading_xbid	FIELD	o		Enum	**"TRADING_XBID_TYPE_OPER"** – Trading on XBID is allowed by OTE at OTE-COM (in operation). **"TRADING_XBID_TYPE_SUSP"** – Trading on XBID is suspended by OTE at OTE-COM. Used only for market_id "MARKET_ID_TYPE_XBID".
revision_no	FIELD	m		Integer(64)	Revision number of the market. With every change of the market state this value is increased by one.

Table 30 - Market state report message structure

2.8.4.16. Hub-to-Hub ATC Matrix Request (HubToHubReq)

HubToHubReq	
Type:	Inquiry Request
Roles:	EmtasImTsAcc
Routing Keys:	`market.request.inquiry`
Request Limits:	2/10

This request provides Hub-to-Hub data capacity matrix acquisition.

Message/Field	Type	m/o	No.	Data Type	Short description
HubToHubReq	MSG	m		Structure	
standard_header	*FIELD*			*Structure*	*Standard header of each message. Please see chapter 2.6.7 Standard message header.*
delivery_area	FIELD	m		String	Delivery Area
delivery_day	FIELD	m		Timestamp	Date

Table 31 - Hub-to-Hub matrix request message structure

2.8.4.17. Hub-to-Hub Matrix Report (HubToHubResp)

HubToHubResp	
Type:	Inquiry Response
Response to:	HubToHubReq (sent to private autogenerated response queue)
Broadcast:	No
Roles:	EmtasImTsAcc

This message is sent in response to *HubToHubReq*. It is sent to the private message queue of the user, who submitted the corresponding *HubToHubReq* request.

Message/Field				Type	m/o	No.	Data Type	Short description
HubToHubResp				MSG	m		Structure	
standard_header				*FIELD*	*m*		*Structure*	*Standard header of each message. Please see chapter 2.6.7 Standard message header.*
hub_to_hub_atcs				FIELD	m	1..n	Structure	
	delivery_start			FIELD	m		Timestamp	Delivery start date
	delivery_end			FIELD	m		Timestamp	Delivery end date
	timestamp			FIELD	m		Timestamp	Timestamp when the ATC data was received from the Capacity system
		hub_froms		FIELD	o	0..n	Structure	
		from		FIELD	m		String	The outgoing Delivery area code
		atcs		FIELD	o	0..n	Structure	
			to	FIELD	m		String	Inbound Delivery area code
			in	FIELD	m		Integer	Available capacity DA(to)-DA(from)
			out	FIELD	m		Integer	Available capacity DA(from)-DA(to)

Table 32 - Hub-to-Hub matrix report message structure

2.8.4.18. Hub-to-Hub Notification (HubToHubNtfRprt)

HubToHubResp	
Type:	Broadcast
Broadcast:	Yes
Broadcast Routing Keys:	public.<market_id>
Roles:	EmtasImTsAcc

This message is distributed automatically whenever the *Hub-to_Hub* data matrix is modified (for example when a cross-border trade is established or in the case of an explicit allocation).

Message/Field				Type	m/o	No.	Data Type	Short description
HubToHubNtfRprt				MSG	m		Structure	
standard_header				*FIELD*			*Structure*	*Standard header of each message. Please see chapter 2.6.7 Standard message header.*
hub_to_hub_atcs				FIELD	m	1..n	Structure	
	delivery_start			FIELD	m		DateTime	Delivery start date
	delivery_end			FIELD	m		DateTime	Delivery end date
	timestamp			FIELD	m		DateTime	Timestamp when the ATC data was received from the Capacity system
	hub_froms			FIELD	o	0..n	Structure	
		from		FIELD	m		String	The outgoing Delivery area code
		atcs		FIELD	o	0..n	Structure	
			to	FIELD	m		String	Inbound Delivery area code
			in	FIELD	m		Integer	Available capacity DA(to)-DA(from)
			out	FIELD	m		Integer	Available capacity DA(from)-DA(to)

Table 33 - Hub-to-Hub matrix report message structure

2.8.5. IM reference data

This chapter describes the message structure including information about the delivery and market area. Since the central XBID side distributes data only for the Czech market and delivery area to OTE, MP OTE also receive information only for the Czech area.

2.8.5.1. Delivery Area Information Request (DeliveryAreaInfoReq)

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

Message/Field	Type	m/o	No.	Data Type	Short description
DeliveryAreaInfoReq	MSG	m		Structure	
standard_header	*FIELD*			*Structure*	*Standard header of each message. Please see chapter 2.6.7 Standard message header.*
product_names	FIELD	o	0.. 1000	String	List of products.

Table 34 - Delivery are information request message structure

2.8.5.2. Delivery Area Information Report (DeliveryAreaInfoRprt)

DeliveryAreaInfoRprt	
Type:	Inquiry Response, Broadcast
Response to:	DeliveryAreaInfoReq (sent to private autogenerated response queue)
Broadcast:	Yes
Broadcast Routing Keys:	public.<market_id>
Roles:	EmtasImTsAcc

This message is distributed whenever a delivery area attribute is modified. It is also sent in response to the *DeliveryAreaInfoReq* request.

Message/Field		Type	m/o	No.	Data Type	Short description
DeliveryAreaInfoRprt		MSG	m	1	Structure	
standard_header		*FIELD*	*m*		*Structure*	*Standard header of each message. Please see chapter 2.6.7 Standard message header.*
delivery_areas		FIELD	o	0..n	Structure	
	delivery_area_id	FIELD	m		String	Delivery Area ID.
	revision_no	FIELD	m		Integer(64)	Revision number. With every change of the delivery area this value is increased by one.
	name	FIELD	m		String	Name of the delivery area usually used for display purposes.
	long_name	FIELD	m		String	Long name of the delivery area.
	state	FIELD	m		Enum	Current state of the delivery area. The following values are allowed: **"REFERENCE_DATA_STATE_TYPE_IACT"**: Delivery area is inactive and thus not tradable. **"REFERENCE_DATA_STATE_TYPE_ACTI"**: Delivery area is active. It is possible to trade in that area. **"REFERENCE_DATA_STATE_TYPE_HIBE"**: Delivery area is deactivated (hibernated). Trading in that delivery area is not possible.
	market_area_id	FIELD	m		String	ID of the Market Area this delivery area belongs to.
	product_names	FIELD	o	0..n	String	List of assigned products. In case of a state change for a delivery area, this list is not provided.

Table 3536 - Delivery area information report message structure

2.8.5.3. Market Area Information Request (MarketAreaInfoReq)

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

This message provides acquisition of market area information.

Message/Field	Type	m/o	No.	Data Type	Short description
MarketAreaInfoReq	MSG	m		Structure	
standard_header	*FIELD*			*Structure*	*Standard header of each message. Please see chapter 2.6.7 Standard message header.*
product_names	FIELD	o	0.. 1000	String	List of products.

Table 37 - Market area information request message structure

2.8.5.4. Market Area Information Report (MarketAreaInfoRprt)

MarketAreaInfoRprt	
Type:	Inquiry Response, Broadcast
Response to:	MarketAreaInfoReq (sent to private autogenerated response queue)
Broadcast:	Yes
Broadcast Routing Keys:	public.<market_id>
Roles:	`EmtasImTsAcc`

This message is distributed whenever a market area attribute is modified. It is sent in response to the *MarketAreaInfoReq* request.

Message/Field		Type	m/o	No.	Data Type	Short description
MarketAreaInfoRprt		MSG	m	1	Structure	
standard_header		*FIELD*	*m*		*Structure*	*Standard header of each message. Please see chapter 2.6.7 Standard message header.*
market_areas		FIELD	o	0..n	Structure	
	market_area_id	FIELD	m		String	Market Area ID.
	name	FIELD	m		String	Name of the market area usually used for display purposes.
	long_name	FIELD	m		String	Long name of the market area usually.
	state	FIELD	m		Enum	Current state of the market area. The following values are allowed: **"REFERENCE_DATA_STATE_TYPE_IACT**": Market area is inactive and thus not tradable. **"REFERENCE_DATA_STATE_TYPE_ACTI**": Market area is active. It is possible to trade in that area. **"REFERENCE_DATA_STATE_TYPE_HIBE**": Market area is deactivated (hibernated). Trading in that market area is not possible.
	revision_no	FIELD	m		Integer	Revision number. With every change of the market area this value is increased by one.

Table 38 - Market area information report message structure

2.9. Scenarios for the current automatic communication through the KSP/KSM communication server

2.9.1. Configuration/modification/response to the new IM limit

The current IM limit state including additional values, which returns a modified current limit state in the form of an established SFVOTLIMITS structure.

The structure SFVOTSETTINGS provides an IM limit modification through AC(KSP). Apart from the standard header and the receipient identification it also contains:

```
SFVOTSETTINGS/Setting -main encapsulating data element
SFVOTSETTINGS/Limit -main element used for limit configuration
SFVOTSETTINGS/Limit@type – limit, enum type, currently in IM only
SFVOTSETTINGS/Limit@value – new value for the specified limit in CZK
```

An example where 20 000 CZK is set up as limit:

```xml
<?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="481" 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" value="20000"/>
    </Setting>
</SFVOTSETTINGS>
```

The response contains the RESPONSE structure with msg code 483 and in case of a successful run also a data copy (where SFVOTLIMITS contains msg code 482). The current return codes from financial reporting area are used.

RESPONSE/Reason@code	Popis
S09000	The request has been processed successfully, the configuration has been changed.
S09008	The participant does not have the required configuration (the limits are undefined).
S09009	The participant is not authorized to make the modification.
S09010	Insufficient funds.
S09011	Invalid value.
S09012	Unexpected error.

Table 39 - Response reason codes with msg code 483

2.9.2. Message indicating the transfer of a portion of the IM limit into the main trading limit

During the partial transfer of the IM limit into the main trading limit, which occurs automatically when establishing a trade that depletes the funds of the main trading limit, it is also necessary to notify the participant through the AC. Information sent to the participant is the following:

- Transferred financial value from the IM limit to the main trading limit (CZK)
- Remaining value of the IM limit (CZK)
- Remaining available financial funds in the IM settlement (CZK)
- ID of the trade that initiated the transfer
- Trade delivery date

The SFVOTLIMITCHANGE structure is used for this purpose. It is sent unsolicited via the KSP. Apart from the standard header and the receipient and sender identification, it also contains:

```
SFVOTLIMITCHANGE/Limits – main encapsulating 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, enum type, currently only
IM

SFVOTLIMITCHANGE/Limit@value – new value of the specific limit in CZK

SFVOTLIMITCHANGE/Limit@moved – funds transferred into a different type
in CZK (for IM to utilization of short-term trades in the main trading
limit)

SFVOTLIMITCHANGE/Limit@free – funds available for the specific 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="484 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" value="15000" moved="5000" free="1280"/>
    </Limits>
</SFVOTLIMITCHANGE>
```

3. Using the electronic signature

Messages are transferred between the client application and the backend system in a binary protobuf format. To ensure integrity and indisputability, specific messages are secured via the electronic signature.

Electronic signature security includes the following messages:

- ModifyOrderReq
- AddOrderReq
- ModifyAllOrdersReq

The structures of the above mentioned messages will, after the introduction of the electronic signature, become part of the SignedMessage structure, which will contain an item:

- Content type bytes, which is the original message and the digital signature in a binary CMS format, serialized into a byte field before the signature is created

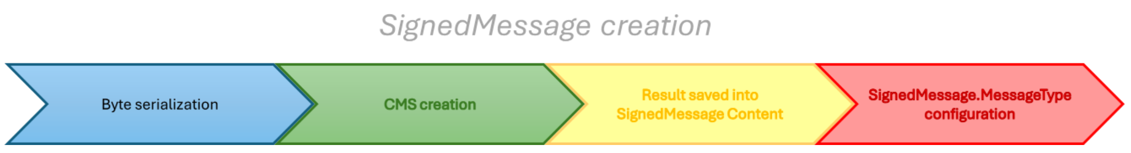

SignedMessage creation
Byte serialization
CMS creation
Result saved into SignedMessage Content
SignedMessage.MessageType configuration

Figure 19 - SignedMessage creation

After the signature and certificate validation on the receipient's side, the original message must be extracted from the CMS format and based on the message type, deserialized into the relevant objects for further processing.

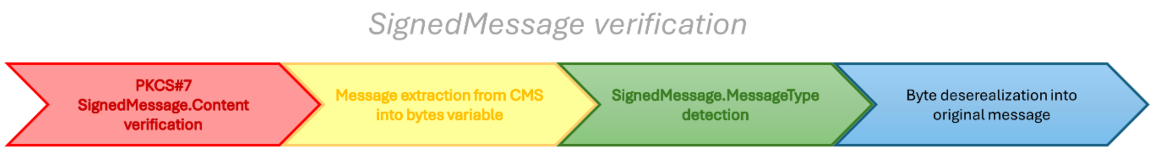

SignedMessage verification
PKCS#7 SignedMessage.Content verification
Message extraction from CMS into bytes variable
SignedMessage.MessageType detection
Byte deserealization into original message

Figure 20 - Digital signature message verification with original message extraction

For a digital signature, it is necessary to use the standard CMS defined in RFC 5625. It is a message of signed-data type, which contains the SignedData ASN.1 structure. It contains the original message, the signature and the certificate corresponding to the private key used for signing. For the hash function, at least the SHA256 algorithm or a stronger one must be used.

SignedMessage structure definition:

Message/Field	Type	m/o	No.	Data Type	Short description
SignedMessage	MSG			Structure	
content	FIELD	m		Bytes	Original binary message together with digital signature in binary format.

Table 40 - SignedMessage message structure