Deutsche Börse AG

Mailing Address
Mergenthalerallee 61
65760 Eschborn

Web
www.deutsche-boerse.com

DFS180 - M7 - Public Message Interface

M7 Release 6.8

Version	1.0
Status	Final
Filename	DFS180 - M7 6.8 - Public Message Interface - v 1.0 - Members
Date	04/11/2019
Author	M7 Project Team
Reviewer	M7 Project Manager

Table of Contents

Terminology

Client	Program or application acting as a client of the AMQP server.
Trader	Person that logs into the client to participate in the market.
Balancing Phase	An additional trading phase, which starts after the end of the normal trading phase, and ends in relation to the delivery start. Normal orders from market participants can be entered, but cannot trigger a trade execution. Only balance orders can trigger a trade execution (a balance order entry is usually limited to one specific member). The order book can be crossed during the balancing phase.

1 Introduction

This document describes the Public Message Interface that the backend system provides to its third party clients. In addition, the Administrative Message Interface that the backend system provides to client applications that supports users with administrative privileges such as Market Operation, are also described. The Administrative Message Interface extends the Public Message Interface to provide additional functionality to these administrative users.

The Public Message Interface allows clients to communicate with the backend system via a programmable interface.

There are 2 kinds of communication between clients and the backend:

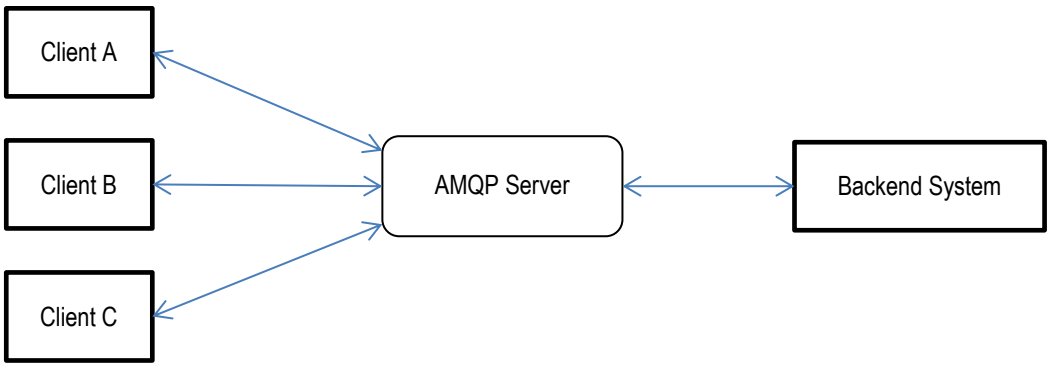
- Management and inquiry requests initiated by the client applications, and the corresponding replies from the backend
- Broadcast messages sent by the backend to all, or specific clients, triggered by market events (e.g. new orders or trades) or by changes in public or private reference data

The communication with the backend system is based on the *Advanced Message Queuing Protocol* (AMQP) as the transport layer. AMQP is a platform and language neutral open standard for the wire protocol[1] on the application layer of the OSI model.

The payload of the messages sent over AMQP are formatted in XML. See www.w3.org/XML for information about XML.

1.1 Overview

Clients that use the Message Interface communicate with an AMQP server, which in turn communicates with the backend system. The backend system itself is behind a firewall and is not directly accessible to the clients.



Overview of the communication between clients and the backend system

The AMQP server is currently using the AMQP implementation from RabbitMQ. This version supports AMQP versions 0.9.1, 0.9 and 0.8. See www.amqp.org and www.rabbitmq.com for more information.

[1] Wire protocol refers to a way of getting data from point to point when multiple applications need to interoperate.

For an optimal interoperability, clients should be implemented using the latest AMQP client library for RabbitMQ, as interoperability limitations are known in the RabbitMQ server implementation. Please refer to the following webpage: https://www.rabbitmq.com/interoperability.html.

2 Getting Started

To get started with the Public Message Interface, you need to:

- Request an account for the AMQP server at the granting authority. You will obtain the following package:
  - A TLS client certificate and password to connect to the AMQP Server (see section Security)
  - A unique application id (if one does not already exist for the used client implementation) that has to be used for further communications with the backend system
  - The source code of the reference client implementation
  - The message format of the XML Schema files (the files are part of the source code for the reference client implementation). You will need these schema files to be able to correctly format messages sent to the backend system and to read its responses. See section 5 for further details.
- Request a trader account if one does not already exist, for participating in the market.
- Download the AMQP client library for RabbitMQ from www.rabbitmq.com. Libraries from other AMQP vendors are not supported due to interoperability issues (vendor interoperability is expected to be implemented from AMQP version 1.0).
- Implement your client. The way that your program should communicate with the backend system is described in the section *Message Exchange Patterns*.

2.1 AMQP

AMQP (Advanced Message Queuing Protocol) is a wire-level protocol that enables message-based communication through the use of an AMQP compliant middleware server (the message broker). In the case of the Public Message Interface this message broker is RabbitMQ.

AMQP defines a modular set of components for the broker, as well as standard rules for connecting them. There are 3 main types of components which are connected into processing chains to create the desired functionality:

- The **"exchange"** receives messages from publisher applications and routes these to "message queues" based on arbitrary criteria, usually message properties or content
- The **"message queue"** stores messages until they can be safely processed by a "consuming" application (or multiple applications)
- The **"binding"** defines the relationship between a message queue and an exchange, and provides the message routing criteria

The exchanges, message queues and bindings that are set up by the backend system are described in more detail in the following chapter.

Please refer to the AMQP and RabbitMQ documentation for more details on the use and configuration capabilities of both AMQP and the RabbitMQ client library.

2.1.1 Connecting to the AMQP server

The first step every client program needs to take is to establish a connection to the AMQP server. This connection can then be used to create the channels that are used to communicate between the client application and the backend server. The various channels needed to communicate with the backend can all share the same connection. The same holds true for a multithreaded implementation: all threads of a client program typically share the same connection, but channels cannot be shared; each channel can only be used by a single thread.

Creating a connection requires the following information to be specified:

- Username, password, server host, port and the virtual host to connect to.
  This information will be supplied by the granting authority.
- TLS context: client certificate and client certificate key

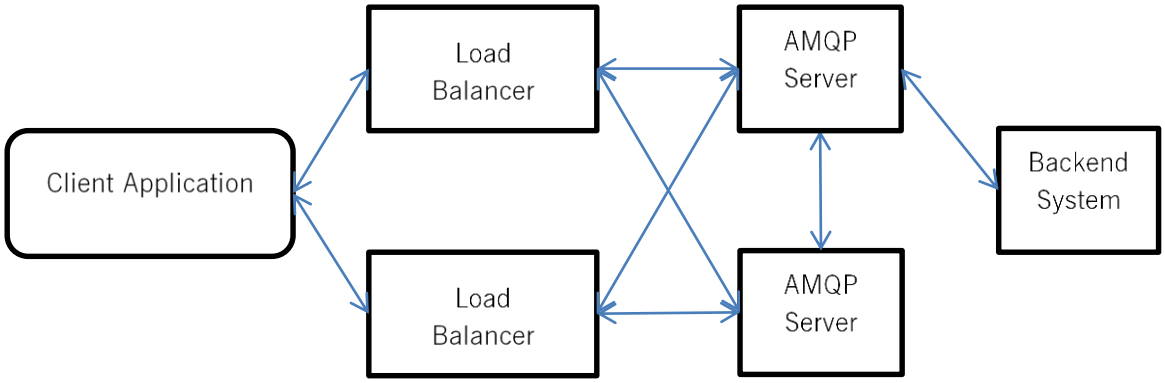
It is mandatory that all connections to the AMQP server are created with AMQP heartbeats enabled. The recommended heartbeat period is 30-60 seconds. Connections that do not have heartbeats enabled will time out on the firewall in the event that there is no traffic on the connection for a longer period of time.

In the RabbitMQ Java client, AMQP-level heartbeats are enabled using the method `ConnectionFactory.setRequestedHeartbeat(int)`*, specifying the heartbeat interval length in seconds.*

In addition, client programs may want to register a shutdown listener on the connection and/or on channels. This can help the application to detect a connection loss faster.

2.1.2 Client Failover

Access to the AMQP server is possible through two different data centres with different IP addresses and URLs. In the standard operating phase, both data centres are active and can be used to make a connection.



All connection details (the port, username, password, client certificate) are the same for each data centre apart from the URL or IP address.

Failover support needs to be implemented on the client side by using the URLs provided. In case a connection through the first URL is not possible, the client shall try to connect to the second URL. DBAG generally suggests to implement a round-robin solution within the log-in procedure to minimise the operational impact in case one data centre is temporarily unavailable.

Please notice that independent of the URL / IP address that is chosen to connect to, all commands will always be routed to the master backend instance.

2.2 Application ID

Each client application will receive its own id that has to be used to enable further communication with the backend system. If no application id is used, or the application id is not configured in the backend system, the client will not be able to participate in the market. The application id will be verified by the backend system for each request.

The application id will be given to each client by the granting authority. Each client is responsible to keep its id secret.

3 Message Exchange Patterns

Two basic patterns of message exchange are supported between the client and the backend system:

- *Request-response* communication, where a client issues a request and waits for a response from the backend system. Each response message is addressed to a specific client (the one that issued the request). This type of communication is initiated by the client.
- *Broadcast* communication, where the backend system publishes notifications that are either public - addressed to all traders - or private - only receivable by privileged traders. Clients may subscribe to receive notification broadcasts that they are interested in. This type of communication is initiated by the backend system.

Typically, all market related information is broadcast by the backend system to all of the client applications that are authorised to receive that information.

The request – response communication is mainly reserved for “actionable” requests (called “management requests” in the rest of the document) e.g. order entry, trade recall request, etc.

The “inquiry requests” serve to obtain information on the current state of the market or reference data. However, they should only be used at the start of a new session to obtain an initial view of the market, or when recovering from communication failures. Subsequent changes in the market situation will be broadcast by the backend system to all connected client applications which should handle these broadcasts, to update the market and reference data they present to their users accordingly. Failing to comply with that pattern may lead to revocation of the respective application id. Note that the backend system limits the number of inquiry requests a client application may submit in a period of time to avoid the excessive and performance degrading use of inquiry requests.

The rest of this chapter provides more detail on these two modes of communication with the backend system.

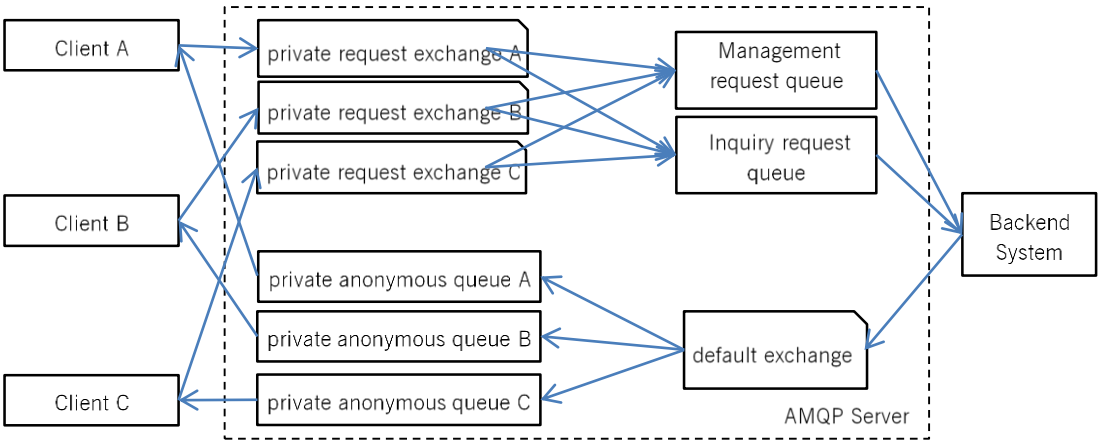
3.1 Request-Response communication

Traders that want to participate in the market send their requests to the backend system. Requests are sent to private trader-specific request exchanges. After processing a request, the backend system sends the response to a private response queue that belongs to the trader that has sent the request.

3.1.1 AMQP configuration

In case a new trader has been setup by the granting authority, the backend system will synchronise with the AMQP Server during runtime and the needed exchanges and queues will be setup within the AMQP Server.

The following diagram details the exchanges and queues that are set up in the AMQP server for request-response based communication between a typical client application and the backend system.



Request-response communication between clients and the backend system

A durable direct exchange named `m7.requestExchange.<login-id>` is set up on the AMQP server for each trader (indicated as "private request exchange X" in the figure above). These exchanges are used for trader requests of all request types.

The response queues used by the client for receiving responses upon requests won't be setup initially by the AMQP Server but from each client. Therefore, the client creates one private response queue and uses the queue name within the reply-to field of a request message.

The M7 system guarantees that it will process request messages from a client in the order in which the messages have been delivered to the queue on the AMQP server.

Each client has to first send a login request using a valid trader that is configured in the M7 system. Without sending a login request, a trader cannot participate in the market. As a result of a login request, a "User" response is sent containing all of the information required to participate in the market. The login of a client is checked by the M7 system. Although only traders with a specific user role can participate in market via the Public Message Interface, roles will also be given by the granting authority.

The M7 system supports a single login functionality so that a trader can only be logged in once.. As a result of this, the M7 system may send a "LogoutRprt" broadcast message containing the session id passed to the client within the "User" response after the login. This broadcast is sent whenever a trader is accessing the M7 system via a different interface or the Public Message Interface itself. When receiving a "LogoutRprt", the trader is already logged out from M7. The client must perform a logout of its trader from Rabbit MQ. If the client doesn't perform a logout of the current trader and allows a parallel login of the same trader, both will consume messages from the trader's response queue and therefore remove messages.

After receiving a "LogoutRprt" a trader can re-login, forcing the trader with the same login credentials to be logged out.

3.1.2 Request Types

The backend system supports two types of requests:

- *Management Requests* are used to enter, modify or delete orders or trades in the backend system.
- *Inquiry Requests* provide market or reference data which is accessible to the client.

As a response to a *management request,* the backend system will send an "AckResp" message to acknowledge the receipt of this request. After processing the request, the backend system will send one or more broadcast messages containing the requested change. All of these responses are sent to the private response queue of the requesting trader.

If the request results in a change in the market or reference data, a second set of broadcast messages will be sent to all market participants (client applications) notifying them of the change.

Example	*When a client application sends a new order to the backend, it will receive an AckResp in reply to confirm the receipt of the order.* *After processing the request, the backend system will send an Order Execution Report to the client application and a Public Order Books Delta Report to all of the client applications that have access to the market data for the product concerned.*

For details on the content of the requests, see section 5 "Message Format".

3.1.2.1 Management-related requests

For management requests, it is possible to send multiple requests of the same type in a single message. The backend system confirms the receipt of the request with one AckResp message. Each individual request will be responded by an individual response message.

When sending an order management request, the request may specify a client order id. The response sent by the backend system contains this client order id, to enable the client to identify the response to the order in its own system.

The maximum number of management-related requests that can be bundled in one single request is determined by a limit which is defined in the XML Schema files for each request. Should this limit be exceeded, the whole request message will be rejected and the client will get an error response containing information about the current limit setting. The limit may be subject to change in future schema versions.

3.1.2.2 Inquiry requests

The following inquiry request types are supported:

- *Private Message Requests are* used to retrieve the client specific private messages held by the backend system.
- *Public Message Requests are* used to retrieve public messages held by the backend system.
- *Trade Requests* are used to retrieve all trades for products that a client is assigned to.
- *Order Book Requests* are used to retrieve the current order book situation managed by the backend system.
- *Order Requests* are used to retrieve orders based on the client assignments.
- *Reference Data Requests* are used to retrieve reference data needed by the client for initialisation.

For inquiry requests, it is only possible to send a single request in each message. The backend system returns a single response for each inquiry request.

3.1.3 How to Send Requests

A client that wants to send a request to the backend system shall do the following:

- Declare one private (exclusive) response queue with the following name on the AMQP server:

```
m7.private.responseQueue.<login-id>.<unique-id>
```

This queue will be used as long as the client is connected to the broker. A *return listener* for the AMQP channel can be used to detect unrouteable requests. The unique ID needs to be generated by the client (e.g. UUID, GUID, etc.). The format of the unique ID can be defined by the client, but it's the responsibility of the client to make sure that a second login with the same login id generates a different unique id. Otherwise the AMQP server will refuse to create a queue with the same queue name.

The maximuim length of a queue name is 127 characters and it is validated[2] by a regexp ^[a-zA-Z0-9-_.:]$

- Create an AMQP message and put the XML-encoded request into the message body.
- Set the `content-type` message attribute to the version of the XML schema used to encode the message body. See "AMQP Message Properties" for the formatting of the content-type attribute.
- Set the `reply-to` message attribute to the name of the private response queue (`m7.private.responseQueue.<login-id>.<unique-id>`). See "AMQP Message Properties" for the formatting of the reply-to attribute.
- Set the `user-id` message attribute to contain the user's login-id. See "AMQP Message Properties "for the formatting of the user-id attribute
- Set the `app-id` message attribute to contain the application id given by the backend system Granting Authority. See "AMQP Message Properties" for the formatting of the app-id attribute
- Set the `correlation-id` message attribute to a unique request ID that can be used to match the responses with the requests. The uniqueness of a correlation-id is the responsibility of each client. The backend system won't check the uniqueness of the correlation-id sent by the client. See "AMQP Message Properties" for the formatting of the correlation-id attribute. The correlation-id may appear in public broadcasts (e.g. in PblcTradeConfRprt after sending an OrdrEntry management request) and may therefore be seen by other clients. It is recommendable to keep the correlation-id unrecognisable.
- Optionally, set the `expiration` message attribute to a time stamp when the message should expire (see details below). See "AMQP Message Properties" for the formatting of the expiration attribute
- Send the message to the `m7.requestExchange.<login-id>` exchange in "*Mandatory"* mode with either the `m7.request.management` or `m7.request.inquiry` routing key, depending on the request type. When sending requests in mandatory mode, the sender will be informed immediately if there is no consumer registered for his request queue. See www.rabbitmq.com for more information.

Once the backend system has processed the request, it will send a response to the client's response queue specified in the request's `reply-to` attribute. The response message is sent with the `correlation-id` attribute set to the value contained in the request. This allows the client to match responses with requests.

The backend system will send a response to each request. As long as the XML schema version used by the client is supported by the backend system, the response will be encoded using the same schema version that was used for the request.

For details about the request and response formats, see section 5 "Message Format".

3.1.3.1 Invalid and Unrouteable Requests

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

[2] Full reference can be found on https://www.rabbitmq.com/protocol.html

If the backend system cannot process the request because the XML schema version in the request message header is missing or invalid, the backend system will send a native error response. This response has the content-type attribute set with a value of `x-m7/error`. The body contains an error message encoded in UTF-8. The reasons for sending a native error message may be caused by validation errors detected by the backend system. Validation errors may occur because of one of the following:-

- The Application Id is not set
- The User Id is not set
- The ContentType is not set
- The ReplyTo is not set
- The CorrelationId is not set

If the backend system cannot process the request because its XML code is invalid, it will send a special xml error response. See the "Message Format" section for information about the "ErrResp" format.

If the backend system cannot process the request because it is down, the request message will be discarded by the AMQP server and the client will be notified about it via its return listener.

3.1.4 How to Receive Responses

The client must receive the response asynchronously using a *consumer*. The client registers a consumer for the response queue, and AMQP invokes the consumer when a message arrives. This method allows the client to wait passively without consuming any CPU. The maximum wait time must be limited, because it is possible that the response will never arrive (see below).

Once a response is received, the client should extract the following message attributes:

- Attribute `content-type`, which contains the XML schema version used for the encoding of the response. This allows the client to choose the correct XML schema for XML unmarshalling. See "AMQP Message Properties" for information about the formatting of the content-type attribute.
- Attribute `correlation-id`, which contains the correlation ID from the request. This allows the client to match responses with requests. See "AMQP Message Properties" for information about the formatting of the correlation-id attribute.
- Attribute `type`, which contains the delivered Message classname (e.g. `ContractInfoRprt`) The Classname for each message is shown in the "Message properties" summary table header.

For an example of some sample code that receives responses using a consumer, please refer to the reference client implementation.

3.1.5 Acknowledgement of Responses

The client must not leave any unacknowledged messages on the AMQP server. Client applications are requested to use the auto acknowledgement functionality on RabbitMQ channels to acknowledge the receipt of all response messages as soon as the message has been received (before processing the message).

If the client does not receive a response in a timely manner, it is encouraged to re-inquire for such message as described in 3.1.7 "Unacknowledged Requests".

If the server side queue gets filled with unacknowledged messages, it might lead to high memory consumption. The private response queues on the server side are constantly monitored and in case one of the queues hit a certain threshold, the connection might be actively disconnected by the server and the Application ID gets deactivated.

Please refer to the Rabbit MQ documentation for more details on the acknowledgement of messages at the AMQP level.

3.1.6 Connection Failure

Clients need to be designed to handle unexpected connection closures. When a connection closes unexpectedly, some requests will not be acknowledged and the client must re-connect and then proceed as described in the "Unacknowledged Requests" section below. It is otherwise not possible to know which requests were processed by the backend, and which were discarded when the connection closed.

It is recommended that clients register a shutdown listener for the AMQP connection to implement this behaviour. The shutdown listener will be called whenever the client detects the connection has closed, regardless of the cause.

Clients must use an exponential back-off when the re-connection repeatedly fails.

3.1.7 Unacknowledged Requests

A request may not be acknowledged because the connection used to send it is closed (see 3.1.6 Connection Failure), the request was discarded without being processed, or because the response was discarded. To detect that a message has been discarded, the client must time-out when waiting for responses.

The timeout setting of the client has to take network latency into account. The excessive resending of requests will impact performance.

When a request is unacknowledged, the client must determine if the request was received by the backend.

For non-management requests, it is possible to resend the request to receive the response. You may receive the response more than once. For Management requests, you must send the required inquiry requests to determine if the management request has been executed. If the request has not been executed, you can resend the request.

3.1.8 Message Overflow Handling

If the backend system is not able to process requests (it is not listening on the request queue or it is too busy), the client's attempts to send request messages will be rejected by the AMQP server (because the messages are sent in mandatory mode).

To detect this, the client must register a return listener for the channel being used to send requests. This is the only circumstance under which the client may assume a request was not processed by the backend. See www.rabbitmq.com for more information about how to register return listeners.

This ensures that most requests are either processed immediately or rejected. Otherwise, client requests could get stalled in the request queue without the possibility to cancel them.

3.1.9 Flow Control

If clients send requests too often, a backlog of unprocessed requests could build up on the server side, resulting in delays in request processing, negatively affecting all of the clients.

Several measures are taken to prevent this scenario.

3.1.9.1 Request Rate Limit

The backend system constantly monitors the request rate of each user and each inquiry request type. For each inquiry message, the backend system defines a short term and long term request limit per user. Currently the short term and long term time periods are set to 1 minute and 60 minutes respectively.

If the number of a specific inquiry request sent by a user within a time period exceeds the configured limit, the backend system will start rejecting this request from that user until the request rate goes back below the limit. The backend system will send the following error response to the user:

Limit is " + <requestLimit> + " per " + <requestPeriod> + " ms.

When the request rate is exceeded, instead of processing an incoming request, the backend system sends a special *back off error response* containing details about the length of the measurement period and the request rate limit.

Note that the limits are specified in such a way as to allow a full initial market state inquiry, as well as a 'normal' amount of failure recovery. If no failures occur, client applications can remain up to date in respect of the market and reference data by correctly processing the broadcast messages sent by the backend system.

The limits are counted starting with the first request of each particular type.

Example	*If the short and long term limits for a message are set to 1 and 10 respectively, the backend system will only allow one request per minute (application of the short term limit). A subsequent request from the same user should thus be sent at least 1 minute after the previous request of the same message type has been sent.* *In addition, the same request cannot be sent more than 10 times in a period of 1 hour (60 minutes), even if the requests are spaced more than 1 minute apart (application of the long term limit).*

3.1.9.2 Message Expiration

Clients may specify an expiration time for each request message. This allows the clients to protect themselves against the situation when a request waiting in the request queue for a long time becomes obsolete, but eventually gets processed even if the client does not wish to execute the request anymore.

3.1.9.3 AMQP Server Flow Control

In the AMQP protocol, *flow control* is an emergency procedure used to halt the flow of messages from a peer. It works in the same way between a client and a server, and is implemented by the AMQP `Channel.Flow` command. Flow control is the only mechanism that can stop an over-producing publisher.

This feature however, is not used by the RabbitMQ server. Instead, in the event that the server hits a preconfigured memory watermark, it uses TCP backpressure to temporarily block all connections that publish messages.

The RabbitMQ server persists the queue contents to disk if required. This allows it to keep more messages than will fit into the machine memory; the queue size is theoretically only limited by the disk space.

In case of very high traffic, the AMQP server can be forced to temporarily throttle publishers to gain time to move some queue contents to disk, releasing memory for new messages. Once this is done, the server will start receiving messages again. (Under normal circumstances, this should not happen because the measures described above should always keep the queues relatively small.)

3.1.10 Message type

The M7 system uses an AMQP message attribute named `type` to provide information about the content of each message. Example: `ContractInfoRprt`

This attribute can be used by a client application to determine the content of each message even before unpacking/processing.

Note: The message type of heartbeat messages is “NULL”.

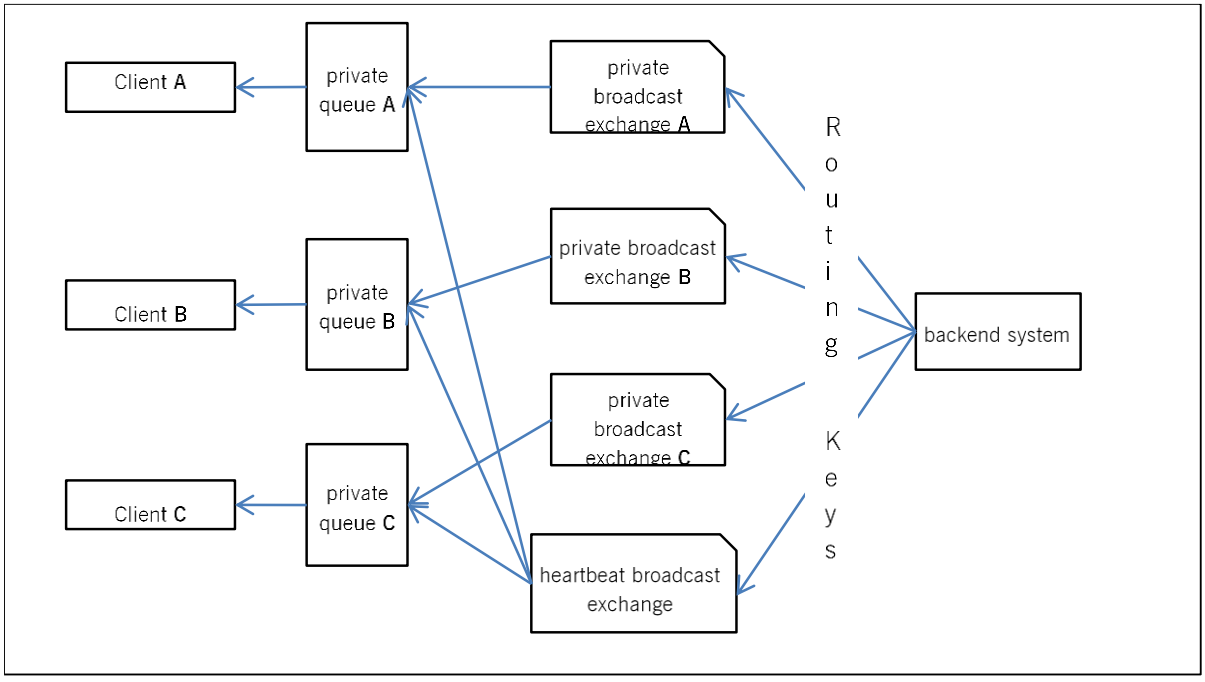
3.2 Broadcast

The backend system broadcasts three types of information:

- *Heartbeat* broadcasts sent in regular intervals allow the clients to monitor the availability of the backend system. This is also used to check if clients are still connected to the backend system.
- *Market Data* broadcasts inform clients about changes within the current market. Traders will only receive information if its assignments are matched by the market data change.
- *Reference Data* broadcasts inform clients about any changes to the reference data.

Broadcast messages sent by the backend system are distributed (pushed) via the AMQP server to all clients that have subscribed to receive the information.

Several system operations (ex: contract closure, service or delivery areas halt/trading) may generate a large number of broadcasts since they have an impact on reference/market data (ex: contracts or orders state).



The broadcast routing architecture of the backend system

Broadcast messages are encoded and sent using all supported XML schema versions. To avoid supporting too many schema versions, and therefore a payload overhead the backend system only supports the latest 2 XML schema versions.

3.2.1 M7 Heartbeat Broadcasts

A durable topic exchange named

- `m7.heartbeatExchange`

is used for broadcasting heartbeat messages. This exchange is bound to the traders broadcast queue.

The heartbeat messages are sent with a routing key

```
<schema-version>.m7.heartbeat
```

Each heartbeat message contains information about the heartbeat interval length, as well as a message attribute within the header property of each message containing the send timestamp called “server-timestamp”. This allows the clients to monitor the availability of the back-end system. An increased message latency indicates a higher system load and/or network delays. See the “Heartbeat” section for information about the heartbeat message format.

The backend system has a connection loss functionality to enable a user to specify actions that will be executed in a failure scenario. If a client has not been connected to the message broker for a specified amount of time, the prior defined actions (see Message Format) will be executed and the client will be logged out by the backend system.

Please note the difference between application-level heartbeat broadcasts sent by the backend system and AMQP-level heartbeats:

- AMQP heartbeats are defined by the AMQP protocol specification and allow the client to monitor the existence of a connection between the client and the AMQP server.
- Heartbeat broadcasts published by the backend system are proprietary to the Public Message Interface and allow the client to monitor the availability of the backend system.

The suitable re-connection strategy to be used in the event of a heartbeat loss (e.g. the number of acceptable missed heartbeats) is completely dependent on the third party’s API client implementation.

3.2.2 Market Data Broadcasts

A durable topic exchange is setup per client on the AMQP Server.

- `m7.broadcastExchange.<login-id>`

The backend system sends messages to the specific trader broadcast exchange whenever a respective action has been performed. Those messages will be delivered to the trader’s private queue that will also be setup on the AMQP Server. The private clients queue will be named

- `m7.broadcastQueue.<login-id>`

The backend system sends broadcasts messages whenever a change has occurred, either initiated by traders, or the backend system itself. Based on the current data model used in the backend system, different routing keys are used to deliver broadcast messages to the trader’s private exchanges and queue. See the “Message Format” section to find out what routing key is used by the backend system upon a market data change.

The Broadcast queue is created automatically before the UserReport is sent out. A client can subscribe to broadcasts by binding a consumer to its private queue. Clients cannot subscribe to all existing queues. The access is controlled by configuring privileges for broadcast queues based on the trader’s assignments. If access to a broadcast queue is not granted, the client will not be able to bind a consumer to its private queue.

Example A message (formatted according to schema 6.0) containing information about an order entry done by *TM001-BG1------X* for product *Intraday_Power_D* and delivery area *10YDE-RWENET---I* will be sent using the routing key:

```
6_0.Intraday_Power_D.10YDE-RWENET---I.TM001-BG1------X
```

The message will be pushed to all private queues bound to exchanges using the same routing key.

3.2.3 Reference Data Broadcasts

As each trader has its private broadcast queue, reference data messages will also be sent via the private queue, using specific routing keys. Only messages sent by the backend system that match the routing key will be delivered to the trader's private queue.

The backend system sends messages to the traders broadcast queue whenever it needs to publish a reference data change to the clients.

A client can subscribe to broadcasts by binding a consumer to its private queue.

Not all traders can receive all broadcasts. For each trader, the access to its private queue is controlled by configuring privileges. If access to a broadcast queue is not granted, the client will not be able to bind a consumer to its private queue.

Example A message (v6.0 format) containing information about the suspension of trader CXDBSX01 will be sent using the routing key:

```
6_0.CXDBSX01
```

The message will be pushed to the trader's queue that is bound to the exchange using the same routing key:

```
m7.broadcastQueue.CXDBSX01
```

3.2.4 Sequence counting for Broadcast Messages

A sequence number is used to identify the order of the broadcasts, and to find out if some broadcasts have been lost. The sequence number is not part of the payload, but it is stored within the header of the AMQP message as an attribute called `x-m7-group-sequence`.

The sequence will be always increased in increments of one for the next broadcast. It will be stored in-memory only (NOT persisted), which means that when the backend system shuts down or terminates, the sequence will be reset to 0 (e.g. in case of a failover event when the original master node went down ungracefully). Whenever the client gets a value which is not expected (i.e. value different than last_value+1) it should request the market data from the backend system.

Rarely, it may happen that client gets a value which is lower than expected but still not necessarily a reason for the synchronization of market data. When the same message is received twice, for example, sequence 11 is received again after sequence 12, it may have been caused by incorrect message acknowledgment on the side of consumer application. In such case, it is safe to ignore the duplicated message, however it is recommended to introduce some reasonable threshold (i.e. > 0) to be able to distinguish this situation from the standard sequence reset.

Note a discontinuous sequence and user disconnection are two independent things, therefore a sequence reset does not necessarily lead to a user being disconnected. A discontinuous sequence indicates inconsistent data, therefore the reaction of the client shall be to perform its re-synchronisation. In case of a sequence reset, all sequence IDs will be reset to 0.

The sequence number is counted based on the routing keys (attribute `x-m7-group-id` in message header). For each routing key there will be a different sequence number. All queues that are bound to the default broadcast exchange with the same routing key will receive the same sequence id. The order of broadcasts are guaranteed only on the level of routing key or the attribute `x-m7-group-id`. M7 or RabbitMQ does not guarantee any sending or reception order of messages across several routing keys.

3.2.5 Broadcast Distribution from M7 Back-End to AMQP Server

All broadcasts from M7 back-end to AMQP server go through seven different connections. The distribution of broadcast messages into each connection is performed based on the routing key:

- One connection is for broadcasts with the routing key containing the string `'.bg.'` (i.e. Order Execution Report and Settlement Process Information Report).
- Four connections are for broadcasts with the routing key containing the string `'.prddlvr.'` (i.e. Public Order Books Delta Report and Implied Order Report).
- One connection is for all remaining broadcasts.
- One connection is for heartbeats. The heartbeats are sent through the dedicated connection in order to minimize the risk of slowing down their delivery to the heartbeat broadcast exchange in Rabbit MQ.

Please note the information contained in this paragraph is for informative purposes only and may be subject to changes. This includes also the number of connections used for the broadcast distribution from the M7 back-end to the AMQP Server as well as the distribution rules for each such connection.

3.2.6 How to Receive Broadcasts

As the trader's private broadcast queues are durable queues and are already setup in the AMQP Server any client that wants to subscribe to broadcasts simply registers a consumer for its private queue.

Whenever a message is broadcast by the backend system, a copy of the message will be placed into the trader's private queue and the client's call back method will be invoked on the registered consumer.

To stop receiving broadcasts, the client should remove the consumer from its private queue.

The broadcast queues within the AMQP Server are setup as durable queues to avoid the loss of any messages due to connection problems. The broadcast queues are also configured with a time to live for messages that are put into the queue (e.g. 60 seconds). This will protect the AMQP Server from out of memory issues in case a client consumes the messages too slow or because of any other connection problems.

3.2.7 Acknowledgement of Broadcasts

The client must not leave any unacknowledged messages on the AMQP server. Client applications are requested to use an auto acknowledgement on RabbitMQ channels to acknowledge the receipt of all response messages, as soon as the message has been received (before processing the message). It is strongly recommended to implement a gap-detection mechanism as described in chapter 3.2.4 "Sequence counting for Broadcast Messages" to prevent any data-loss.

If the server side queue gets filled with unacknowledged messages, it might lead to high memory consumption. The private broadcast queues on server side are constantly monitored and in case one of the queues hit a certain threshold, the connection might be actively disconnected by the server and the Application ID gets deactivated.

3.2.8 Failover Processing

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

In case of a backend system failure, the client subscriptions will stay active, but clients will not receive any broadcasts until the backend system is restarted. Once the backend system is restarted, delivery of broadcasts will resume without any further actions being required on the client side. The client will recognise that the backend system is down because no more heartbeat broadcasts will be sent. Note that in the case of a backend restart, the sequence numbering for the requests will be restarted.

Any broadcast messages published by the backend system whilst a client was disconnected, will be lost for that client if the client does not reconnect within the specified time to live time for the messages within the queue.

3.2.9 Flow Control

When a client is consuming messages too slowly, a backlog of messages waiting for processing may build up in the private queue(s) owned by this trader.

If messages exceed the time to live limit, the AMQP server will start deleting those messages from the trader's private queues. Thus, a slow client may lose messages.

The clients are therefore required to consume and acknowledge each message as soon as possible. If the processing of a message is a non-trivial task, the receiver must handle the receipt of the message separately from the actual processing of the message: the message must be consumed, acknowledged and stored in a memory buffer of the client, where it awaits processing.

It is important that clients consume messages as quickly as possible. Should a client fail to adhere to these rules and consume messages too slowly, it may lose messages and display a non-up to date market state.

3.2.10 Message type

The M7 system uses the AMQP message attribute named `type` to provide information about the content of each message. Example: `ContractInfoRprt`.

This attribute can be used by a client application to determine the content of each message even before unpacking/processing.

4 Security

All communication between the client and the AMQP server is encrypted using the Transport layer security (**TLS) version 1.2 only**. Older TLS versions are not supported anymore. Client and server certificates are used to establish a trusted connection. The usage of asymmetric encryption ensures confidentiality, authentication, message integrity assurance and non-repudiation of origin.

4.1 Server Certificate

The backend system uses signed Server Certificates.

Usually the CA root certificates are known and trusted by the software development frameworks like Java or .NET. If not, clients need to add the CA root certificate to a list of trusted certificates. The needed CA root certificate files can be downloaded directly from the CAs Website[3].

The exact way this is done depends on the platform and programming language in which the client application is developed. Please note that the CA root certificate has to be imported into a location used by the client application's runtime environment, not into a web browser.

4.1.1 Using CA Root Certificate in Java

For Java clients, the CA root certificate must be imported into a *keystore*, which will be used to initialise the *Trust Manager* used by the client application. The certificate can be imported using the `keytool` utility, which is part of the Java runtime environment:

```
keytool -import -alias m7-ca -file <cacert> -keystore <keystore>
```

In the command above, `<cacert>` is path to the CA certificate file in PEM format (`cacert.pem`) and `<keystore>` is path to the keystore file. You have to confirm that you trust the certificate being imported.

The keystore file will be created if it does not exist already. Java keystores are protected by a password; choose a password when first creating the keystore and then use it whenever accessing the keystore.

Refer to Java documentation for more information about the `keytool` utility. Example code can be found in the reference client implementation.

4.2 Client Certificate

An organisation that wants to use the Public Message Interface of the backend system has to apply for an account on the AMQP server. The following is provided when a new account is set up:

- AMQP server host names, port number and virtual host name.
- The trader's login ID if it doesn't already exist.
- A client certificate and private key that will be used by the client when connecting to the AMQP server.
- A CA root certificate that has to be imported as a trusted certificate on the client side.

[3] VeriSign: http://www.verisign.com/support/roots.html
Comodo: https://support.comodo.com/index.php?_m=downloads&_a=view&parentcategoryid=1

The client certificate:

- complies to the X.509v3 specification,
- uses a key length of 2048 bits,
- subject contains an unique identifier as the *Common Name*,
- is signed by Deutsche Börse CA,
- is provided in format PKCS#12 (`.p12`).

The client's private key is used to verify the client's identity when communicating with the AMQP server. Should a third party get hold of the client's private key, it could forge client's identity and access the encrypted communication with the server.

It is therefore extremely important to keep the private key secure.

4.2.1 Using a Client Certificate in Java

To be able to use the client certificate and private key in a Java client, they need to be loaded into a keystore, which is used to initialise a *Key Manager*. Java supports the PKCS#12 format (`.p12`), where the private key is protected by a passphrase. The passphrase is provided to the client along and it must be used when loading the `.p12` file into the keystore.

Please refer to the reference client implementation for sample Java code. For further details regarding TLS security, please consult the TLS specification, AMQP specification and https://www.rabbitmq.com/ssl.html.

4.3 Authentication

When a client connects to the AMQP server using the TLS protocol, both peers are mutually authenticated by exchanging their certificates during the TLS handshake. All subsequent communication between the client and the server is encrypted using a key and encryption algorithm agreed on during the handshake. To create a connection to the AMQP Server, the username and password are required. The AMQP Server is connected to an internal LDAP Server which will verify the given credentials. In case of an unsuccessful authentication the client won't be able to connect to the AMQP Server.

Whenever a client sends a request to the AMQP server, it uses a client-specific exchange. The exchange name contains the client login id, which can be extracted by the backend system when processing the request to identify from which client the request came.

4.4 Authorisation

Authorisation of client actions occurs on two levels:

4.4.1 Authorisation by an AMQP server

The AMQP server verifies client's privileges when a client accesses resources stored on the AMQP server. This includes exchanges and queues that the client tries to configure, read or write. This applies to:

- binding queues to private broadcast exchanges
- sending messages to request exchanges

In the event of insufficient privileges, the attempt to access the resource is immediately (synchronously) rejected by AMQP.

4.4.2 Authorisation by the backend system

The backend system verifies privileges when processing requests from clients.

In the event of insufficient privileges, the request is rejected by the backend system. From the client point of view, this rejection occurs synchronously for inquiry requests as a negative response message is sent to the traders response queue

and asynchronously for management requests as a negative response message is sent to the traders private exchange using the routing key <schema-version>.trdr.<login-id> .

5 Message Format

All messages sent between the backend system and clients have an XML-encoded payload and specific AMQP message properties. This section describes the xml payload format and the AMQP message properties in detail. The xml payload format specification comes in the form of XML schemas, which specify the allowed structure and format of XML elements and attributes.

5.1 General Information

5.1.1 AMQP Message Properties

The following message properties have to be set when sending a request to the backend system:

AMQP Message Property	Description
content-type	Contains information about the XML payload version used, as well as the used message type. Valid content-type definitions are (the version number has to be filled with the used version): ▪ x-m7/request; version=x (Used by the clients when sending requests) ▪ x-m7/response; version=x ▪ x-m7/broadcast; version=x ▪ x-m7/heartbeat; version=x ▪ x-m7/error; version=x
reply-to	Contains the predefined trader's queue name that a response has to be sent to (See 3.1)
user-id	Contains the login-id of the logged in trader
app-id	Contains the application id given by the granting authority
correlation-id	Contains the request message id generated by each client
expiration	Contains an optional entry specifying if the request should be deleted if it is not executed within the specified time
contentEncoding	Contains ***gzip***, if messages are compressed (content is encrypted using gzip method); the property is null if messages are not compressed
header	Can contain connecting application version in format (optional) **app-version**:version where version must be in the integer(s) format, optionally separated by dots (*i.e. 10 or 4.1.5*)

If an app-version is provided, M7 checks if it's higher or equal than the minimum version for the given app-id. If the app-version is lower, the connection is refused.

If an app-version is not provided, the check is not performed and the connection is allowed.

5.1.2 XML Convention

- Element tags are only used for structure reasons.
- Data is only carried in attributes, never in element tags.
- Types:
    - All Elements are bold, Attributes are in regular text

- **SE:** Structure Element. No Data is embedded between tags, but it may contain attributes. Contains no data. (grey background and bold)
- **CE:** Content Element. Data is embedded between tags, and can also contain attributes.(bold)
- **A:** Attribute of an Element.

- The sorting of elements and attributes is not guaranteed and might change in the future.
- **The m/o** column specifies if the presence of the element/attribute is mandatory or optional
  - Value "**c**" means conditional. Such an attribute is marked as optional in the API, but may be required from the service depending on system settings or the particular state of reference data. Such a condition is always specified.

5.1.3 Standard Message Header

Every message will contain a header at the beginning of the message.

XML Tag	Type	m/o	No.	Data Type	Short description
StandardHeader	SE	m		Structure	
marketId	A	m		Char(4)	Market Identification Code (MIC) of the market to which the request is sent or from which the request originates.
onBehalfUserId	A	o		Integer	UsrId of the target user in the case of On-behalf trading (see 5.1.11).

Table 1: The message layout of the Standard Message Header.

5.1.4 XML Validity and Data Binding

It is important that clients produce valid XML code when sending requests to the backend system. The backend system uses a validating parser which rejects any requests that do not strictly conform to the supported XML schemas.

To make parsing and the creation of XML data easier, and to avoid problems with the validity of the XML code, clients are encouraged to use XML data binding.

5.1.5 Schema Version

Every message sent or received by the backend system must specify the XML schema version that was used to encode its payload in its metadata. This information is stored in the message properties in the contentType field as a string.

The contentType should contain value that corresponds to the major schema version.

Example: For XSD schema version 6.5.3 the correct contentType = 6.0 because the major schema version in this case is 6.0.

This information can be used to validate that both peers use the same version of the payload format. The backend system rejects requests with an unexpected schema version.

5.1.6 M7 Heartbeat

The heartbeat contains the message SYSTEM_ALIVE:<interval> and the message attribute "server-timestamp" within the header property of each message.

In case the message with the interval has not yet been received, the client shall wait for 5 seconds to receive the M7 back-end heartbeat after its first connection to the broadcast queue.

5.1.7 Quantity values in Messages

Quantity values in all messages (requests and responses) are given as integer values which exactly represent the database values stored in the backend. The interpretation/display of these values depends on the product attributes *decShftQty*, minQty and *qtyUnit* (see 6.4.19).

The attribute *decShftQty* defines the position of the decimal point within the integer value (e.g. the integer value 1000 with a *decShftQty* of 3 means a decimal number of 1.000).

The attribute *minQty* defines the possible quantity steps which can be entered into the system and at the same time the smallest possible quantity value (e.g. a *minQty* of 100 means, quantities can be entered in 100 steps starting with 100: 100, 200, 300, etc. – the value of 50 would be rejected). The *minQty* can also be used to limit the number of displayed decimal places for a quantity value: With a *minQty* of 100, the last two zeros might be cropped when displaying quantities, because they are always zero.

The attribute *qtyUnit* contains a string with the unit of the quantity values (e.g. “MW” for megawatt in the power market).

Table 2 shows some examples.

decShftQty	smallestTrdUnit	qtyUnit	Value in the messages	Possible display value
3	100	MW	1300	1.3 MW
3	100	MW	700	0.7 MW
3	1000	MW	34000	34 MW
0	100	Ltr.	700	700 Ltr.

Table 2: An example of displaying quantity values

5.1.8 Price values in Messages

Price values in all messages (requests and responses) are given as long values which exactly represent the database values stored in the backend. The interpretation/display of these values depends on the product attributes *decShftPx* and *currency* (see 6.4.19).

The attribute *decShftPx* defines the position of the decimal point within the long value (e.g. the long value 1276 with a *decShftPx* of 2 means a decimal number of 12.76).

The attribute *currency* contains a 3 character long identifier for the currency. A *decShftPx* of 2 for the currency *Euro* means that the values are given in Eurocents (3499 = 34.99 EUR = 3499 Eurocents).

5.1.9 Date time values in Messages

Date time values in messages (data type in the message layout tables is “DateTime”) are given as XML Schema DateTime values in the following format:

`YYYY-MM-DD`**`T`**`hh:mm:ss.sss`**`Z`** (2012-09-14T12:34:23.351Z)

Symbol	Description	Example
YYYY	The year	2012
MM	The month	09
DD	The day of the month	14
T	Separator between the date and the time section	T
hh	The hour of the day (24 h)	12
mm	The minute of the hour	34
ss	The seconds of the minute	23
sss	The milliseconds of a second	351

Z	Time zone - Zulu time zone = UTC time	Z

All Date/Time values are given in the UTC time zone. To get local times, the client has to add or remove hours based on the local time zone.

Contract naming patterns are affected by the time zone set at the product level.

The market time zone can be obtained using the SystemInfoResp message (see 6.1.5)

5.1.10 Date values in Messages

Date values in messages (data type in the message layout tables equalling “Date”) are given as a Date value in ISO format:

YYYY-MM-DD (2018-09-24)

5.1.11 On-behalf Trading

The backend system supports trading on-behalf of another user. There are three different levels of on-behalf trading:

- *On-behalf trading on a member level*: Traders having the right to trade on-behalf are allowed to perform actions for all traders belonging to the same member, the same balancing group and product configuration.
- *On-behalf trading on a broker level*: Broker users are allowed to trade on-behalf for all traders that are assigned as the assigned members for their account regarding the user product configuration.
- *On-behalf trading on an admin level*: All Admin users are allowed to trade on-behalf for all traders in the system. In relation to the user product configuration (only products assigned to the trader can be traded on behalf, even if the admin himself has wider products assignments).

In order to submit requests to the backend as on-behalf requests, the standard message header field *onBehalfUserId* must be filled with the user ID of the target user, for whom the request is sent on-behalf (see 5.1.3). Although the standard message header is part of every message in the Public Message Interface, the field *onBehalfUserId* is only relevant for the following request types:

- Order Entry Request (OrdrEntry)
- Order Modify Request (OrdrModify)
- Order Request (OrdrReq)
- Order Execution Report (OrdrExeRprt)
- Pre-Arranged Order Processing (PreArrangedOrdrProcess)
- Trade Recall Request (TradeRecallReq)
- Message Request (MsgReq)
- Message Report (MsgRprt)
- Trade Capture Request (TradeCaptureReq)
- Trade Capture Report (TradeCaptureRprt)
- Delete Quotes Request (DeleteQuotesReq)
- Order Limit Request (OrdrLmtReq)

If the field is filled in for a request which does not support on-behalf trading, it will be ignored by the backend.

In case of an on-behalf request, the user with the *onBehalfUserId* will be retrieved on the backend side and the user context of the request is changed to this user. This means that the request is treated by the backend like a normal request directly from the user.

5.1.12 Message properties summary table

The description for every message starts with a table that summarises the key properties of the message. The following table describes the different properties and their meaning:

The message classname is present in the table header.

Property	Description
Type	Type of the message: • **Inquiry Request**: A message to retrieve information out of the backend system during the initial load phase or in case of a detected gap. • **Inquiry Response**: A response message to an Inquiry Request. • **Management Request**: A message to send new business information to the backend system to change business objects. • **Management Response**: An response message to a Management Request. • **Broadcast**: The message is sent as a broadcast, initiated by the backend system. One message can have several types.
Roles	User roles that are allowed to send or receive the message. Possible values: Trader, Market Operations, Market Makers, Brokers, Sales, Data Vendor, Settlement Operations and <ALL>
Routing Keys	The routing key(s) used to decide the message queue destination. (request messages only)
Response To	Request message to which the response is a reply message (response messages only). If not specified, the message is a broadcast-only message.
Broadcast	Indicates if the response message can be sent as a broadcast (response messages only). If 'NO' then the message is only sent as a reply to a request.
Broadcast Routing Keys	Routing keys specified for a broadcast message. These are empty for non-broadcast responses (response messages only) Broadcast routing keys will be deprecated and may be decommissioned in future releases. For documentation purposes they will be replaced by "Broadcast audience" information.
Broadcast Audience	Contains an audience scope for the Broadcast message.
Request Limits	Request limit values for Inquiry Request (see also 3.1.9.1): <shortTermLimit>/<longTermLimit> (Example: 1/10; max 1 request every minute and 10 requests per hour) The limits given in this document are the default values. They can be changed during runtime of the backend system, if necessary.

5.1.13 Any elements and attributes

For forward compatibility reasons, each entity in PMI is now expandable with "any" element and "anyAttribute" in the XSD schema. For more details please refer to chapter 9.

6 Public Requests and Responses

The requests and responses described in this chapter are used for users without administrative privileges in order to communicate with the backend.

6.1 General Requests and Responses

The general requests and responses described in this chapter are used to login and logout of the backend system, as well as for responding to management requests.

6.1.1 Login Request (LoginReq)

LoginReq	
Type:	Inquiry Request
Roles:	<All>
Routing Keys:	`m7.request.inquiry`
Request Limits:	14/70

The Login Request is sent to the backend system to participate in the market. As a response to this request, either a User Response (successful login – see 6.6.1) or an Error Response is returned. The Error Response might indicate that a trader with same credentials is already logged into the backend system. The Login Request has to be sent at least 5 seconds after a successful Logout Request (see 6.1.2).

XML Tag	Type	m/o	No.	Data Type	Short description
LoginReq	SE			Structure	
user	A	m		String(255)	The Login ID of the user that wants to login to the backend system.
force	A	m		Boolean	Flag that indicates if this user want to force a login even if a user with the same credentials is already logged in into the backend system.
disconnectAction	A	m		String	An action that will be executed in case of an unexpected connection loss. The following values are allowed: "**NO**": No action is executed. "**DEACT_USER_ORDRS**": All orders of this user will be deactivated. "**DEACT_ACCT_ORDRS**": All orders entered into the backend system of the assigned trader accounts will be deactivated.
StandardHeader	*SE*	*m*		*Structure*	*Standard header of each message. Please see 5.1.3.*

Table 3: The message layout of a Login Request.

6.1.2 Logout Request (LogoutReq)

LogoutReq	
Type:	Inquiry Request
Roles:	<All>
Routing Keys:	`m7.request.inquiry`
Request Limits:	14/70

The Logout Request has to be sent by a client application if the trader logs out of the backend system manually. The queue deletion or channel closing is handled by the backend system. The maximum amount of time needed for Logout Request associated activities (e.g. RabbitMQ queue deletion) is 5 seconds.

XML Tag	Type	m/o	No.	Data Type	Short description
LogoutReq	SE			Structure	
sessionId	A	m		Long	Session id of the trader's session which is passed to the client upon login.
StandardHeader	*SE*	*m*		*Structure*	*Standard header of each message. Please see 5.1.3.*

Table 4: The message layout of a Logout Request.

6.1.3 Logout Report (LogoutRprt)

LogoutRprt	
Type:	Inquiry Response, Broadcast
Response to:	LogoutReq (sent to the private response queue, see 3.1)
Broadcast:	Yes
Broadcast Routing Keys:	`<schema-version>.trdr.<user-id>`
Broadcast audience	`Only the user with the usrId involved.`
Roles:	<All>

The Logout Report is used

- as a response to the Logout Request (to the private response queue see 3.1) in which the attribute usrId and sessionId will be set to respective values (see table below).
- as a broadcast Logout Report to the user who is logged out, as a consequence of a concurrent forced login with the same user credentials. The attribute forced is set to 'true' (routing key: <schema-version>.trdr.<user-id>). The usrId and sessionId will be set respectively (see table below) in this case as well.
- as a broadcast in case of any other event leading to the logout of the user

The following table shows the format of the Logout Report:

XML Tag	Type	m/o	No.	Data Type	Short description
LogoutRprt	SE			Structure	
usrId	A	m		Integer	The internal trader id used in the backend system.
sessionId	A	m		Long	The session id of the trader's session passed to the client upon login.
txt	A	o		String(255)	A text field containing information about the reason of the logout.
forced	A	o		Boolean	Indicates whether a user has been forced to log out. Is 'true' as a consequence of a concurrent login with the same user credentials where subsequently the user is forced to logout. Is 'false' as a response to a regular LogoutReq.
StandardHeader	*SE*	*m*		*Structure*	*Standard header of each message. Please see 5.1.3.*

Table 5: The message layout of a Logout Report.

6.1.4 System Info Request (SystemInfoReq)

SystemInfoReq	
Type:	Inquiry Request
Roles:	<All>
Routing Keys:	`m7.request.inquiry`
Request Limits:	14/70

The System Info Request is used to get general information about the backend system (see the response message description below).

XML Tag	Type	m/o	No.	Data Type	Short description
SystemInfoReq	SE			Structure	
StandardHeader	*SE*	*m*		*Structure*	*Standard header of each message. Please see 5.1.3.*

Table 6: The message layout of a System Info Request.

6.1.5 System Info Response (SystemInfoResp)

SystemInfoResp	
Type:	Inquiry Response
Response to:	SystemInfoReq (sent to the private response queue see 3.1)
Broadcast:	No
Broadcast Routing Keys:	---

Roles:	<All>

The System Info Response returns general information about the backend system as reply to a System Info Request.

XML Tag	Type	m/o	No.	Data Type	Short description
SystemInfoResp	SE			Structure	
backendVersion	A	m		String(255)	The version number of the backend system.
backendTimeZone	A	m		String(255)	The time zone identifier of the time zone the backend system runs in.
backendMarketTimeZone	A	m		String(255)	The time zone identifier of the time zone the market is operated in. This time zone is used on backend side to generate time zone related information like contract short names.
contractStoreTimeInDays	A	m		Integer	The number of days that contracts are available in the system (today included). If the parameter equals 7 for example, the users are able to display their contracts and respective trades from the last 7 days, including today. The contracts, and the related trades are marked for removal from the system at the date and time calculated as (current date and time – contractStoreTimeInDays). Afterwards they are removed in bulk during the next few hours. Therefore some contracts/trades may be visible for several hours longer than e.g. 7 days exactly.
appVersionActual	A	o		String(32)	Contains the actual version for the application identified by the app-id message property in the requesting message (SysInfoReq). This may be used on the client side to check if an up-to-date application is used.
maxOrders	A	m		Integer	The maximum number of orders that are allowed to be sent within one order entry/modify message
maxQuotes	A	m		Integer	The maximum number of quotes that are allowed to be sent within one order entry/modify message (order types Q and W)
capabilities	A	o	0..n	String(255)	A list of the backend features available. If the feature is not in the list, it is disabled on backend side. Possible values: **"SETTLEMENT"**: The Update Settlement Information message is supported by the system (see 7.2.3) and trade settlement status functionality is available. **"LOCAL-EXCHANGE"**: The exchange is acting as a local exchange. **"PNC-ORDERS":** Private and confidential entry of pre-arranged trades is available (see 6.4.13).6.4.13). **"QUOTE-ORDERS"**: Mass Quote requests are supported by the system. **"QUOTE-REQUESTS"**: Order Quote requests are supported by the system (see 6.4.34). **"OPEN-CLOSE-INDICATOR"**: The Open-close indicator is supported in Order entry and the following messages (see 6.2.1). **"TRADING-LIMIT"**: Cash limit requests are supported by the system (see 6.4.5).
allowedClearingAcctTypes	A	o		String(255)	Comma separated valid values for the "clearingAcctType" attribute in e.g. OrdrEntry message. Example (spot markets): "**A,P**"
StandardHeader	*SE*	*m*		*Structure*	*Standard header of each message. Please see 5.1.3.*
RequestLimitList	SE	m	1	SE	List of request limits
RequestLimit	SE	m	0..N	SE	Request limit
message	A	m		String(255)	Message name (e.g. AllUsersReq)
duration	A	o		Integer	Limit duration in seconds (e.g. 60 for short term limits)
rate	A	o		Integer	The value of the limit (e.g. 1)

Table 7: The message layout of a System Info Response.

6.1.6 Acknowledgement Response (AckResp)

AckResp	
Type:	Management Response
Response to:	OrdrEntry; OrderModify; PreArrangedOrdrProcess; ModifyAllOrdrs: TradeRecallReq: (sent to the private response queue see 3.1)
Broadcast:	No
Routing Keys:	---

Roles:	Trader, Market Operation

The Acknowledgement Response is sent upon receipt of any management request. If an acknowledgement response is not sent, the client has to send an inquiry request to figure out if the previously sent management request has been executed.

The response is sent back using the same correlation id within its message attributes. Acknowledgement Responses are always sent to the private response queue of the client.

M7 guarantees to send the AckResp before any other broadcast. However, Rabbit MQ operates two asynchronous queues for the AckResp and other responses. Due to the load balancing in Rabbit MQ, the response may be delivered earlier than the AckResp. M7 cannot use only one queue for capacity reasons. Thus, M7 can only guarantee the behaviour that regards the order of sending. In theory, it is possible that, due to the above stated reasons other types of message are received earlier than the AckResp.

Note: because of the new architecture constraints, the clOrdrId was removed from this message.

XML Tag	Type	m/o	No.	Data Type	Short description
AckResp	*SE*			Structure	
StandardHeader	*SE*	*m*		*Structure*	*Standard header of each message. Please see 5.1.3.*

Table 8: The message layout of an Acknowledgement Response.

6.1.7 Error Response (ErrResp)

ErrResp	
Type:	Inquiry Response; Management Response; Broadcast
Response to:	<All> (sent to the private response queue see 3.1
Broadcast:	Yes
Broadcast Routing Keys:	`<schema-version>.trdr.<user-id>`
Broadcast audience:	`The connected user whose action caused the asynchronous error (i.e. OrdrEntry whilst not having a sufficient cash limit).`
Roles:	<ALL>

The Error Response is sent whenever the backend system determines a business exception has occurred, based on wrong entries in the request.

Error Responses can be sent synchronously or asynchronously (broadcast). The backend system sends the Error Response in a synchronous way whenever the received xml contains syntactical errors. In this case the requests have not been processed by the backend system yet. Error Responses are sent by the backend system in an asynchronous way if any errors happen during processing of the request. Synchronous Error Responses get sent to the private response queue of each trader, asynchronous get sent to the private trader's broadcast queue.

Any connected application may use English text provided in the "err" attribute, or can use the provided (as part of the reference implementation) text resource with the transferred variables. This way the localisation of the error text can be completed on the client side (the server side provides no localisation in the err texts).

Example *Message contains:*

```
<Error errCode="12345" err="ACCT1 - account not found">
  <VarList>
    <Var id="0" value="ACCT1"/>
  </VarList>
```

```
</Error>
```

Provided resource file contains:

errCode	*Error string English*
12345	*{0} – account not found*

The client application can either use the err attribute from the message which will always be in English, or it can use the provided resource file to translate to any other language and fill the provided variable itself.

XML Tag	Type	m/o	No.	Data Type	Short description
ErrResp	SE			Structure	
StandardHeader	*SE*	*m*		*Structure*	*Standard header of each message. Please see 5.1.3.*
Error	SE	m	1..n	Structure	
err	A	m		String(255)	The error message for this error. This is always in the English language with variables already replaced by their values.
errCode	A	m		Integer	Predefined errors for better client handling/validation/markup. Possible values are numeric and indicate the specific reasons: A complete list of error codes is provided as an attachment to this document. Some error messages do not have a specific error code. In this case the value is 0.
clOrdrId	A	o		Char(40)	The client order ID (only applicable for errors sent during order management processes)
VarList	SE	o		Structure	A list of variables used in the err text field
Var	SE	o	0..n	Structure	Structure containing variable information
id	A	m		Integer	In Error Response, it is the identifier of a variable within the err resource text (eg. 0). In MsgRprt, it is the identifier of a variable within the message resource text (e.g. 6).
value	A	m		String(255)	Value of the variable (e.g. “Acct1”)

Table 9: The message layout of an Error Response.

6.1.8 Change password request (ChgPwdReq)

ChgPwdReq	
Type:	Management Request
Roles:	<All>
Routing Keys:	`m7.request.management`
Request Limits:	1/10

This message can be used by any user to change his password. It is not possible to change a password on behalf.

As a result, one of following actions is performed

1) ErrResp: The change of password was not successful. The err attribute will contain a detailed error message from LDAP/M7.
2) If the change was successful, a shutdown signal from the broker with the reason PWD_CHANGE is sent, and the user then can relogin with the new password.

The password must comply with the LDAP server password policy; otherwise, an error message from LDAP will appear in the err attribute.

Passwords shall be at least 8 characters long and shall fulfil 3 out of the 4 requirements:

- At least one upper case letter
- At least one lower case letter
- At least one number
- At least one special character.

Passwords can expiry after a certain length of time[4] according to the LDAP settings. If the expiration time is configured in LDAP, M7:

- sends a reminder to the e-mail address set in the user's profile in M7 Admin GUI X[5] days before the password expiry, informing them about the expiration date:

  Login ID: ${loginId}

  Password Expiration Date: ${expiryDate}

  This notice has been sent to you because your password will expire soon.

  Please change your password at your earliest convenience or reset your password ${resetLink}.
- sends a notification on the day of password expiry, containing a link to the password reset page:

  Login ID: ${loginId}

  Password Expiration Date: ${expiryDate}

  This notice has been sent to you because your password expires today. Please change your password at your earliest convenience or reset your password ${resetLink}.

If a wrong password is used 5 times for the login, the Login ID will be suspended.

XML Tag	Type	m/o	No.	Data Type	Short description
ChgPwdReq	SE			Structure	
currentPwd	A	m		String(64)	Current password
newPwd	A	m		String(64)	New password
StandardHeader	*SE*	*m*		*Structure*	*Standard header of each message. Please see 5.1.3.*

Table 10: The message layout of a Change Password Request

6.2 Order Entry and Maintenance

The messages described in this section are used to enter and maintain orders. These messages can be used only by trading participants and are not available for Market Operation users.

6.2.1 Order Entry (OrdrEntry)

OrdrEntry	
Type:	Management Request
Roles:	Trader, Market Operation
Routing Keys:	`m7.request.management`

The Order Entry message enables a trader to send up to 100 orders at once to the backend for execution. The orders are contained in a so called basket. It is possible to define the execution restriction on a basket level.

An order entry triggers the following message flow:

[4] By default, the parameter is set to 90 days.

[5] By default, the parameter is set to 10 days.

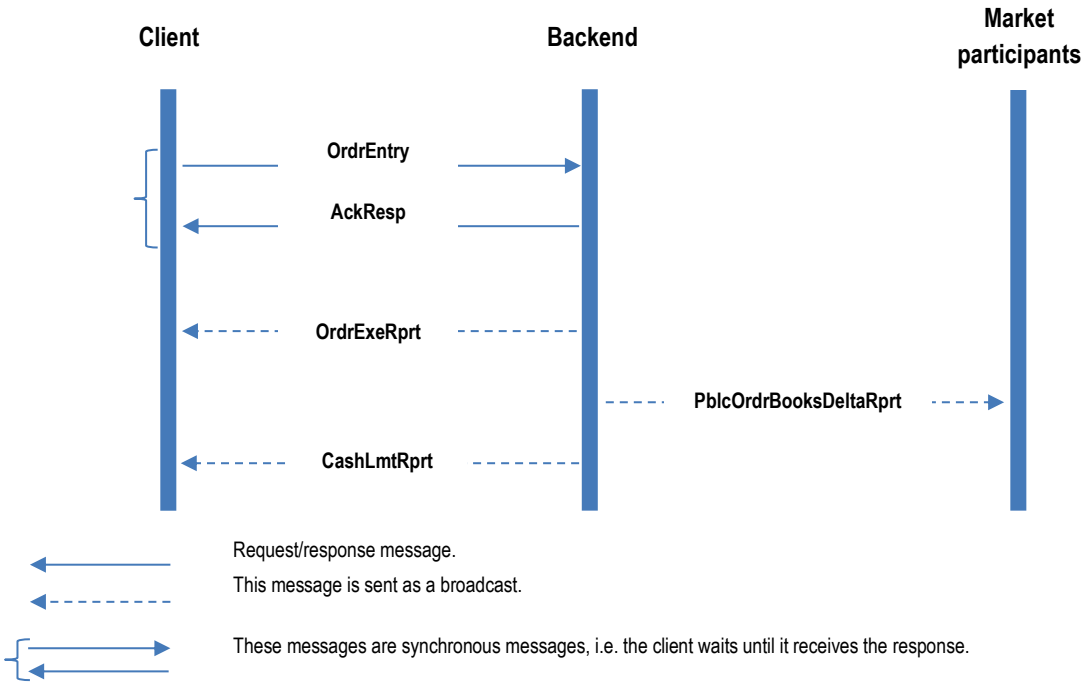


Figure 1: The The message flow triggered by an Order Entry. [6]
Actions caused by partial or full execution or invalid order parameters are not part of this diagram.

Only one acknowledgement (AckResp) or one Order Execution Report (OrdrExeRprt) is sent back, even if the OrdrEntry request contains several orders.

In case of order executions (trades) public and private messages will be broadcast (PblcOrdrBooksDeltaRprt). In addition to the OrdrExeRprt, a Trade Capture Report will be sent to the affected parties, the public order books will be updated (giving rise to Public Order Books Delta Reports), and the connected data vendors will also receive updated data.

When an OrdrEntry request is received for a remote product which has the local trading end time before the XBID trading end, M7 will enrich the OrdrEntry to contain the local trading end in the exGTD field and forward it to XBID. The enrichment will be made for all orders except ones with Execution Restriction codes FOK and IOC.

A Cash Limit Delta Report is sent only in case the order entry has an impact on the cash limit, otherwise the message is not sent.

The Q order type cannot be mixed with other order types in one message (otherwise ErrResp is returned).

The W order type cannot be mixed with other order types in one message (otherwise ErrResp is returned).

The message layout is defined in the following table.

[6] **Note**: The order of broadcasts are guaranteed only on the level of the routing key or the attribute x-m7-group-id. For more information on the order of the broadcast messages see 3.2.4.

XML Tag	Type	m/o	No.	Data Type	Short description
OrdrEntry	SE	m	1	Structure	
StandardHeader	*SE*	m		*Structure*	*Standard header of each message. Please see 5.1.3.*
listExecInst	A	o		String(5)	Defines the execution instruction for the whole list of orders: **"NONE"**: All orders are treated independently. - This value is also to be used if the OrdrEntry message contains only **one order**. **"VALID"**: All orders must be valid, meaning they must pass the order validation of the backend 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. **"LNKD"**: Linked orders - the orders in the basket are linked and must be executed either all at once, or the whole list is rejected. A basket with the basket order restriction LNKD can only contain orders with the order restriction FOK. All orders in such a basket must be entered either for local products, or all for remote products (a combination of remote and local orders is not allowed). It is possible to submit orders for different products in a basket. The Linked Orders feature is configurable and might be turned off. Please check its availability with the System Info Request (see 6.1.4) and the Product Information Request (see 6.4.18). **"IMPL"**: Implied order
OrdrList	SE	m	1	Structure	List of all orders contained in the basket.
Ordr	SE	m	1.. N	Structure	
clearingAcctType	A	m		String (2)	Defines if the order is entered on a trader's own account, or as an agent. For valid values please refer to the values from the attribute allowedClearingAcctTypes in the SystemInfoResp message (i.e."A,P" for spot markets). See chapter 6.1.5
acctId	A	m		String(32)	Account for which the order is entered. The order is rejected if the trader tries to enter an account to which he is not assigned.
contractId	A	c		Long	Defines the underlying contract of the order. This value must be set for all pre-defined contracts. It may be omitted only in the case of an order in a user-defined contract.
prod	A	c		String(255)	Product identifier. This is mandatory in case that the contract Id is omitted.
side	A	m		Char(4)	Defines on which side of the market the order is entered ("BUY", "SELL").
px	A	m		Long	Limit price of the order. The price format is described in 5.1.8.
stopPx	A	c		Long	Stop price for stop limit orders. Mandatory if the type="S"
ppd	A	o		Long	Peak price delta for Iceberg orders. • The ppd of the buy orders must be smaller or equal than zero. • The ppd of the sell orders must be greater or equal than zero. If it is omitted the system will assume a value of "0,00".
qty	A	m		Integer	Contains the total quantity of the order. In case of an Iceberg order this field corresponds to the hidden quantity + display quantity.
displayQty	A	c		Integer	Used to define the display quantity of an Iceberg Order. This field is only required when the type='I'.

XML Tag	Type	m/o	No.	Data Type	Short description
ordrExeRestriction	A	c		Char(3)	Execution restriction of the order. Valid values: **"NON":** No restriction. This is the default setting. **"FOK"** (Fill or Kill): The order is immediately fully executed or deleted. **"IOC" (**Immediate and cancel): The order is executed immediately to its maximum extent. In the event of a partial execution, the remaining volume is removed from the order book. **"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. **"AU"** (Auction): The order was entered in the auction phase (no restriction is applied) If the product has an execution restriction of NON, then NON, FOK, IOC are allowed. Only NON is allowed for order type = "I". AON is allowed for type = "B". If the product has an execution restriction of AON, then FOK or AON are allowed. IOC is allowed for market sweep for order type = "B".
txt	A	o		String(250)	Text entered by the client. This text will not be modified by the backend system. The maximum possible length is 250 characters.
dlvryAreaId	A	c		String(16)	Defines the delivery area of the order. This is mandatory for exchanges with a multiple delivery area setup.
clOrdrId	A	o		String(40)	Client Order Id with a maximum length of 40 characters. This value is not modified by the backend and may be used by client applications to identify orders.
preArranged	A	m		Boolean	This flag indicates if the entered order is a pre-arranged order or not.
preArrangedAcct	A	c		String(16)	This is required in case of a pre-arranged order. It contains the account of the counterpart.
type	A	m		Char(1)	**"O":** Regular limit order. **"B":** User defined block order. **"I":** Iceberg order. **"L":** Balance order. **"C":** Indicative order. **"S:** Stop limit order. **"E:** On exchange prearranged trade **"N:** Private and confidential trade **"H:** Lifting order for products with the Hit & lift matcher "**Q**": Quote order "**W**": Indicative quote order
dlvryStart	A	o		DateTime	The start of delivery for the underlying contract.
dlvryEnd	A	o		DateTime	The end of delivery for the underlying contract.
validityDate	A	c		DateTime	This field is mandatory in the event that validityRes equals "GTD". It is used to define the date until which the order is valid. The remaining part of the order will be removed from the order book after this point in time.
validityRes	A	o		Char(3)	Validity restriction of the order. If this field is omitted, the order will be treated as a "Good for Session" order. Valid values: **"GFS" (**Good for trading session): The order rests in the order book until it is either executed, removed by the user or the current trading session (trading phase) of the underlying contract ends. **"GTD"** (Good till date, will be introduced with CX 3.5): The order rests in the order book until the date specified in the vldtyDate field. **"NON"** (No validity restriction): Mandatory for orders with execution restriction codes "FOK" or "IOC".
state	A	o		Char(4)	**"ACTI"**: The order is entered and immediately exposed to the market for execution. This is the default value. **"HIBE"**: The order is entered into the backend system but is not exposed to the market.
openCloseInd	A	c		Char(1)	Mandatory for Futures product and Cross product spreads. For other Commodities is not present. Valid values: **"O":** Open position indicator **"C":** Close position indicator

XML Tag	Type	m/o	No.	Data Type	Short description
exGTD	A	o		DateTime	This attribute is deprecated and should not be used. Its value is ignored and will be deleted in a future version.
aot	A	o		Boolean	The indicator whether the order shall be automatically transferred to the corresponding linked contract after the trading in the specific delivery area ends in XBID. Default value: false
ClgHse	SE	c	0..n	Structure	List of the Clearing House elements The priority order will be the same as the order of the Clearing House in the xml message. The Clearing House specified first will have the top priority, and the Clearing House specified at the end of the file will have the least priority. This is mandatory if the clearing house functionality is set-up on the exchange. See SystemInfoResp(chapter 6.1.5)
clgHseCode	A	m		String(255)	Clearing House Code
clgAcctId	A	m		Integer	Clearing account Id

Table 11: The message layout of an Order Entry message.

6.2.2 Order Modify (OrdrModify)

OrdrModify	
Type:	Management Request
Roles:	Trader, Market Operation
Routing Keys:	`m7.request.management`

This message is used to modify one or several orders. If the order modification has an impact on the order execution priority, the old order will be removed by the system and a new order with a new order id will be entered instead. As a result, after an order modification is made which has an impact on the execution priority, two OrdrExeRprt messages are returned:

- First the deletion of the modified order is reported.
- Secondly, an Execution Report detailing the newly added order is also sent.

In case of remote products which have a local trading end time that is before the XBID trading end time, the value of the local trading end time contained in the exGTD field is persisted during the order modification. Modifications performed to local schedules or their assignments, however, do not trigger the sending of OrdrModify with updated values of the exGTD field for the existing remote orders.

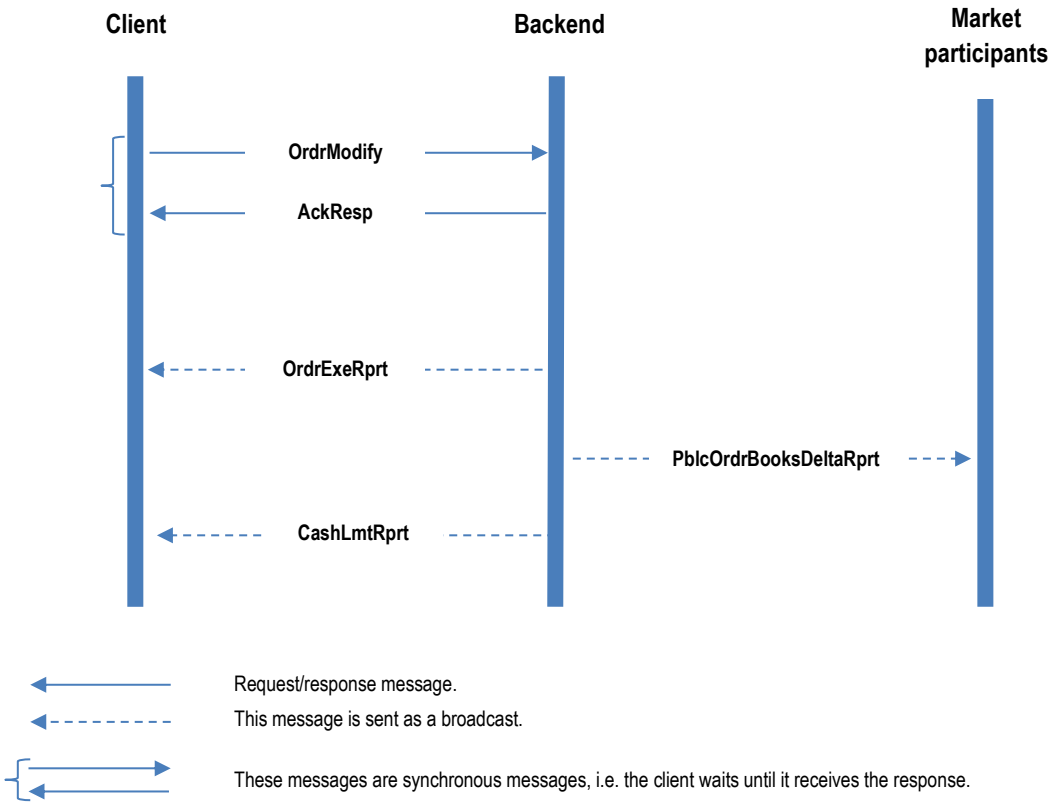


Figure 2: The message flow triggered by an Order Modification.

The The message layout of a the order modification message (OrdrModify) is described in the following table.

XML Tag	Type	m/o	No.	Data Type	Short description
OrdrModify	SE	m	1	Structure	
StandardHeader	*SE*	*m*		*Structure*	*Standard header of each message. Please see 5.1.3.*
ordrModType	A	m		Char(4)	Offers the possibility to activate, deactivate, modify or delete all orders contained in the basket. **"ACTI"**: Activate all orders contained in this basket. Orders that are already active are ignored. **"DEAC"**: Deactivates (hibernates) all orders contained in the basket. Hibernated orders are removed from the order book but are still available for modification or activation in the own orders list. **"MODI"**: Modifies all orders in the basket. **"DELE"**: Deletes all orders in the basket.
OrdrList	SE	m	1	Structure	List of all orders contained in the basket.
Ordr	SE	m	1..N	Structure	Definition of a single order.
ordrId	A	m		Long	Order Id is created by the backend system. This value is used to identify the order to be modified.
clOrdrId	A	o		String(40)	The Client Order Id has a maximum length of 40 characters. This value is not modified by the backend and may be used by client applications to identify orders.
px	A	m		Long	Limit price of the order. The price format is described in 5.1.8.
stopPx	A	c		Long	Stop price for stop limit orders. Mandatory if the type="S"
ppd	A	o		Long	Peak price delta for Iceberg orders. • The ppd of buy orders must be smalle than or equal to zero. • The ppd of sell orders must be greater than or equal to zero. If it is omitted, the system will assume a value of "0,00".
qty	A	m		Integer	Contains the total quantity of the order. In the case of an Iceberg order, this field corresponds to the hidden quantity + the display quantity.
displayQty	A	c		Integer	Used to define the display quantity of an Iceberg Order.
ordrExeRestriction	A	o		Char(3)	Execution restriction of the order. Valid values: **"FOK"** (Fill or Kill): The order is immediately fully executed or deleted. **"IOC" (**Immediate and cancel): The order is executed immediately to its maximum extent. In the case of a partial execution, the remaining volume is removed from the order book. **"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. **"NON":** Any restriction. **"AU"** (Auction): The order was entered in the auction phase (no restriction is applied) If the product has an execution restriction code of NON, then NON.FOK,IOC are allowed If the product has an execution restriction code of AON, then FOK or AON are allowed
txt	A	o		String(250)	Text entered by the client. This text will not be modified by the backend.
type	A	m		Char(1)	Order type. Valid values: **"O":** Regular limit order. **"B":** User defined block order. **"I":** Iceberg order. **"L":** Balance order. **"C":** Indicative order. **"S:** Stop limit order. "**Q**": Quote order "**W**": Indicative quote order Note: types H,E,N cannot be used in OrdrModify

	validityDate	A	c		DateTime	This field is mandatory in the event that validityRes equals "GTD". It is used to define the date until which the order is valid. The remaining part of the order will be removed from the order book after this point in time.
	validityRes	A	o		Char(3)	Validity restriction of the order. If this field is omitted, the order will be treated as a "Good for Session" order. Valid values: **"GFS" (**Good for trading session): The order rests in the order book until it is either executed, removed by the user or the current trading session (trading phase) of the underlying contract ends. **"GTD"** (Good till date, will be introduced with CX 3.5): The order rests in the order book until the date specified in the vldtyDate field. **"NON"** (No validity restriction): Mandatory for orders with the execution restriction code "FOK" or "IOC".
	revisionNo	A	m		Long	The latest revision number of the order must be provided by the client. In case the backend has another revision number, it will reject the request with an ErrResp.
	openCloseInd	A	c		Char(1)	Mandatory for Futures product and Cross product spreads. For other Commodities it is not present. Valid values: **"O":** Open position indicator **"C":** Close position indicator
	exGTD	A	o		DateTime	Exchange – Good-till-Date: The XBID SOB instance will delete an order at this time Only Exchange Users and Central Admins are allowed to enter date into this attribute Certain resolution may be required for this attribute (default is 5 minutes)
	aot	A	o		Boolean	Indicates whether the order has been automatically transferred to the corresponding linked contract after the trading in the specific delivery area ended in XBID.
	ClgHse	SE	c	0..n	Structure	List of the Clearing House elements The priority order will be the same as the order of the Clearing House in the xml message. The Clearing House specified first will have the top priority, and the Clearing House specified at the end of the file will have the least priority. This is mandatory if the clearing house functionality is set-up on the exchange. See SystemInfoResp(chapter 6.1.5)
	clgHseCode	A	m		String(255)	Clearing House Code
	clgAcctId	A	m		Integer	Clearing account Id

Table 12: The message layout of an Order Modify message.

6.2.3 Order Request (OrdrReq)

OrdrReq	
Type:	Inquiry Request
Roles:	Trader, Market Operation
Routing Keys:	`m7.request.inquiry`
Request Limits:	1/10

The Order Request is used to request all own orders, which are currently active or hibernated for the given account and products. The Order Request is acknowledged by an Order Execution Report (see 6.2.4).

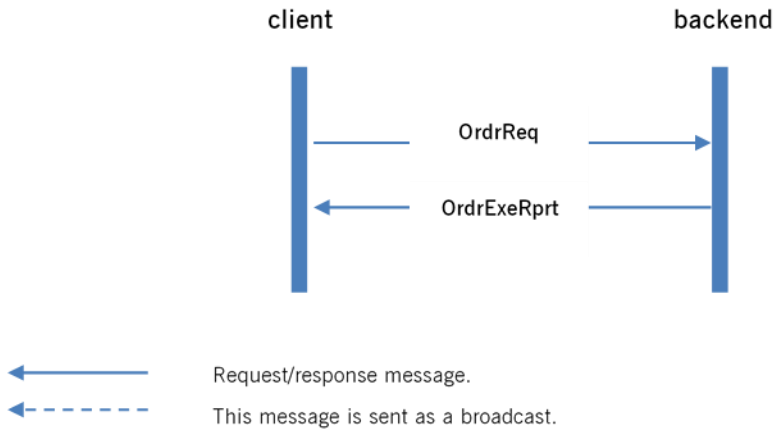


Figure 3 – The message flow triggered by an Order Request.

The following table lists the message layout for the Order Request

XML Tag	Type	m/o	No.	Data Type	Short description
OrdrReq	SE	m	1	Structure	
StandardHeader	SE	m		Structure	*Standard header of each message. Please see 5.1.3.*
prodName	CE	o	0.. 1000	String(255)	List of product names of which own orders should be returned. If no product name is given, the own orders for all products assigned to the requesting user are returned.
acctId	A	c		String(32)	Account Id of which own orders should be returned. This account should match an account assigned to the user sending the request. This element is optional for Market Operation users. If it is not supplied, it will return the orders for all accounts.
inclPreArranged	A	o		Boolean	Include pre-arranged orders in the response or not. Default value: false

Table 13: The message layout of an Order Request.

6.2.4 Order Execution Report (OrdrExeRprt)

OrdrExeRprt	
Type:	Management Response; Broadcast
Response to:	OrdrEntry; OrdrModify; OrdrReq; ModifyAllOrdrs; (sent to the private response queue see 3.1)
Broadcast:	Yes
Broadcast Routing Keys:	`<schema-version>.bg.<acctId>`

Broadcast audience:	`Trader (owner of the order) and traders from his Balancing groups.`
	`Admins`
	`Brokers with an assignment to Trader (owner of the order).`

Roles:	`Trader, Market Operation`

The Order Execution Report is sent in the following cases:

- After a successful Order Entry.
- After an Order Modify request.
- After a partial or full execution of the order.
- In case of the execution of a Pre-arranged order.

As a response to an Order Request, the Order Execution Report is sent to the private response queue of the requesting user see 3.1.

In both cases, traders will receive only the orders entered in the account they are assigned to, whereas market operations users will receive all orders

XML Tag	Type	m/o	No.	Data Type	Short description
OrdrExeRprt	SE	m	1	Structure	
StandardHeader	*SE*	*m*		*Structure*	*Standard header of each message. Please see 5.1.3.*
listExecInst	A	o		String	Original execution instruction for a list of orders. Please see 6.2.1.
listId	A	o		Long	ID for the basket, determined by the backend.
OrdrList	SE	o	0..1	Structure	
Ordr	SE	o	0..n	Structure	
ordrId	A	m		Long	Order Id as returned by the backend system.
initialOrdrId	A	m		Long	In the event of an order modification, this value contains the Id of the first order in the modification chain.
parentOrdrId	A	o		Long	In the event of an order modification, this field contains the Id of the modified order.
clearingAcctType	A	m		Char(2)	Defines if the order is entered on its own account, or as an agent. For valid values please refer to values from attribute allowedClearingAcctTypes in the SystemInfoResp message (i.e."A,P" for spot markets). See chapter 6.1.5
acctId	A	m		String(32)	Account for which the order was entered.
contractId	A	m		Integer	Defines the underlying contract of the order. This value must be set for all pre-defined contracts. It may be omitted only in the event of an order in a user-defined contract.
side	A	m		String(4)	Defines on which side of the market the order is entered. Valid values: **"BUY":** Buy order. **"SELL"**: Sell order.
px	A	m		Long	Limit price of the order. The price format is described in 5.1.8.
stopPx	A	o		Long	Stop price for stop limit orders. The price format is described in 5.1.8.
ppd	A	o		Long	The peak price delta for Iceberg orders.
qty	A	m		Integer	Contains the quantity exposed to the market. In the event of an Iceberg Order, this is the rest of the display quantity.
hiddenQty	A	o		Integer	Contains the hidden quantity of the Iceberg order. The total executable quantity may be calculated by adding the hiddenQty to the qty.
displayQty	A	o		Integer	Used to define display the quantity of an Iceberg Order.
initialQty	A	m		Integer	The initial quantity entered with this order. If the order is partially matched, the initialQty still contains the original quantity value. The initialQty value is related to the current order and not to the value of the initial order in the order modification chain (identified by the initialOrdId).
ordrExeRestriction	A	o		Char(3)	Execution restriction of the order. Valid values: **"FOK"** (Fill or Kill): The order is immediately fully executed or deleted. **"IOC" (**Immediate and cancel): The order is executed immediately to its maximum extent. In the case of a partial execution, the remaining volume is removed from the order book. **"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. **"NON":** Any restriction. **"AU"** (Auction): The order was entered in the auction phase (no restriction is applied)
txt	A	o		String(250)	Text entered by the client. This text will not be modified by the backend.
dlrvyAreaId	A	o		String(16)	Defines the delivery area of the order.
clOrdrId	A	o		String(40)	The Client Order Id has a maximum length of 40 characters. This value is not modified by the backend and may be used by client applications to identify orders.
preArranged	A	m		Boolean	A flag that indicates if the order is a pre-arranged order or not.
preArrangedAcct	A	o		String(16)	The account of the counterparty in case of a pre-arranged order.
type	A	m		Char(1)	Order type. Valid values: **"O":** Regular limit order. **"B":** User defined block order. **"I":** Iceberg order. **"L":** Balance order. "**Q**": Quote order "**W**": Indicative quote order **"C":** Indicative order. **"S:** Stop limit order. **"E:** On exchange prearranged trade

XML Tag		Type	m/o	No.	Data Type	Short description
						"N: Private and confidential trade **"H:** Lifting order for products with Hit&lift matcher
	state	A	m		Char(4)	The current state of the order in the system. Valid values: **"HIBE":** The order is entered into the backend system but is not exposed to the market. "HIBE" status is also set when LTS is disconnected from XBID (see DFS160a). **"ACTI":** The order is entered and is immediately exposed to the market for execution **"IACT":** The order is deleted.
	aggressorIndicator	A	o		Char(1)	Indicates whether the executed order was a trade aggressor or trade originator. The field will only be present in case the Order Execution Report was triggered by a partial or full execution. **Valid values:** •"Y" - Trade aggressor •"N" - Trade originator •"U" - Unknown for executed orders of remote products and data before migration
	usrCode	A	m		String(6)	The user code of the user performing the last successful action on the order.
	revisionNo	A	m		Long	When an order is entered, the revision is set to 1. The revision is increased by 1 in case of a partial execution, hibernation or a modification without an execution priority change. In case of a modification with an execution priority change, the revision number for the deleted order ("UDEL" action) is increased by 1 and the revision number for the newly entered order is reset to 1 ("UADD" action). The behaviour is the same for local and remote orders.
	timestmp	A	m		DateTime	The timestamp of the order entry, as determined by the backend. This timestamp determines the execution priority in case of identical limit prices.
	validityDate	A	o		DateTime	This field is mandatory in the event that validityRes equals "GTD". It is used to define the date until which the order is valid. The remaining part of the order will be removed from the order book after this point in time.
	validityRes	A	o		Char(3)	Validity restriction of the order. If this field is omitted, the order will be treated as a "Good for Session" order. Valid values: **"GFS" (**Good for trading session): The order rests in the order book until it is either executed, removed by the user or the current trading session (trading phase) of the underlying contract ends. **"GTD"** (Good till date, will be introduced with CX 3.5): The order rests in the order book until the date specified in the vldtyDate field. **"NON"** (No validity restriction): Mandatory for orders with the execution restriction "FOK" or "IOC".
	action	A	m		String(255)	Lists the action code. Valid values are: **"UADD"**: Order added by the user. **"UHIB"**: Order deactivated by the user. **"UMOD"**: Order modified by the user. **"UDEL"**: Order deleted by the user. **"UREJ"**: Pre-arranged order rejected by the user. **"AADD"**: Order added by market operations on behalf. **"AHIB"**: Order deactivated by market operations on behalf. **"AMOD"**: Order modified by market operations on behalf. **"ADEL"**: Order deleted by market operations on behalf. **"AREJ"**: Pre-arranged order rejected by market operations on behalf. **"SADD"**: Order added by the system. **"SHIB"**: Order deactivated by the system. **"SMOD"**: Order modified by the system. **"SDEL"**: Order deleted by the system. **"SREJ"**: Pre-arranged order rejected by system. **"FEXE"**: Order is fully executed. **"PEXE"**: Partial execution of order. **"IADD"**: A new slice of an Iceberg order was added to the service.

XML Tag	Type	m/o	No.	Data Type	Short description
					"SERR": The order validation failed on SOB side, or the request sent to SOB timed out. This is only valid for remote orders. "**QADD**": Quote was added "**QFEX**": Quote was fully executed "**QPEX**": Quote was partially executed
lastUpdateUsrInfo	A	m		String(255)	Information (concatenation of accountId and usrCode) about the user who last updated the order, either on their own or on behalf. If the "action" is any of the system actions ("FEXE", "SADD" etc.), then the update has been created by the system and the lastUpdateUsrInfo contains the system user ("7777777777777777SYSTEM").
openCloseInd	A	o		Char(1)	Mandatory for Futures and Cross product spreads. For other Commodities the value is not used. Valid values: **"O":** Open position indicator **"C":** Close position indicator
brokerUserId	A	o		Integer	UsrId of the broker user
counterOrder	A	o		Boolean	Denotes if the order is an artificial counterOrder generated by the cross product matching process. The default value is false.
remoteOrdrId	A	o		Long	The Order Id as returned by the remote backend system (i.e. XBID SOB)
lastUpdateTm	A	m		DateTime	The timestamp of the last modification of the order object in the back-end. Is also updated for modifications which do not affect the execution priority (like the "text" field).
exGTD	A	o		DateTime	**NOT** to be used by a trader.
preAotId	A	o		Long	Local order ID of the remote order that this order originated from during AOT. The field is populated if and only if this is a local order transferred from a remote order during AOT.
aot	A	o		Boolean	Indicates whether the order has been automatically transferred to the corresponding linked contract after the trading in the specific delivery area ended in XBID Default value: false
clgHse	SE	o	1	Structure	List of the Clearing House elements The priority order will be the same as the order of the Clearing House in the xml message. The Clearing House specified first will have the top priority, and the Clearing House specified at the end of the file will have the least priority. For lifting orders, (Order type = H) only one clearing house can be entered.
clgHseCode	A	m		String(255)	Clearing House Code
clgAcctId	A	m		Integer	Clearing account Id

Table 14: The message layout of an Order Execution Report.

6.2.5 Pre-Arranged Order Processing (PreArrangedOrdrProcess)

PreArrangedOrdrProcess	
Type:	Management Request
Roles:	Trader, Market Operation
Routing Keys:	`m7.request.management`

The pre-arranged order processing request is used to accept or reject a pre-arranged trade.

The The message flow generated by this request is depicted in the following graph.

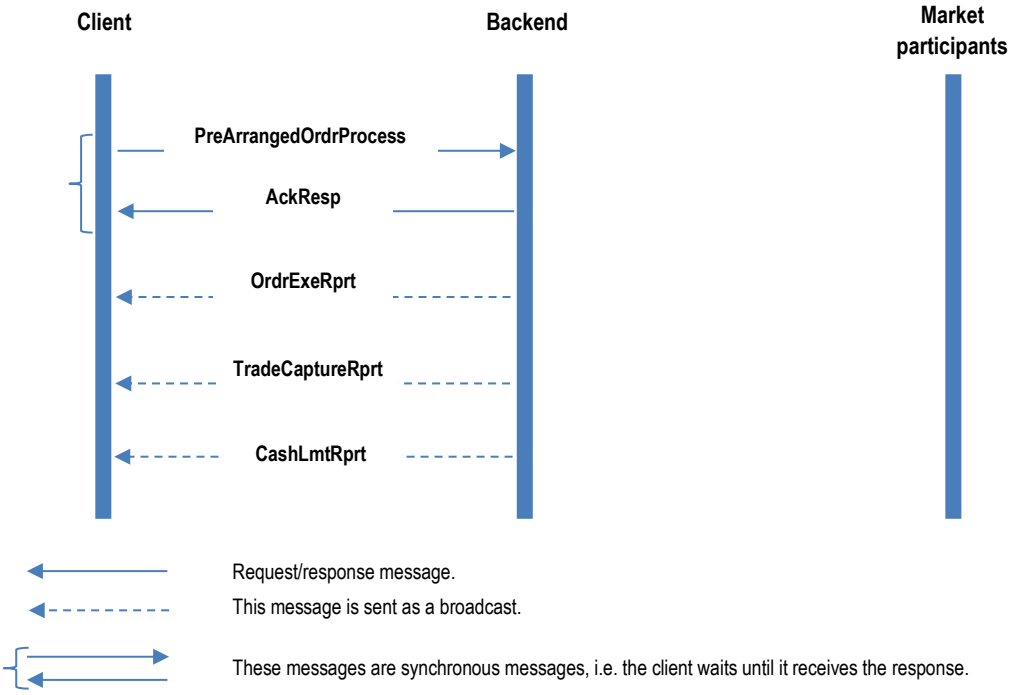


Figure 4: The message flow initiated by a pre-arranged order processing request.

XML Tag	Type	m/o	No.	Data Type	Short description
PreArrangedOrdrProcess	SE	m	1	Structure	
clearingAcctType	A	m	2	Char(2)	Defines if the order is entered on its own account, or as an agent. For valid values please refer to values from attribute allowedClearingAcctTypes in the SystemInfoResp message (i.e."A,P" for spot markets). See chapter 6.1.5
dlvryAreaId	A	c	1	String(16)	Defines the delivery area of the order.
txt	A	o	1	String (250)	Text entered by the client. This text will not be modified by the backend.
clOrdrId	A	o	1	String(40)	Client Order Id. This value is not modified by the backend and may be used by client applications to identify orders.
ordrId	A	m	1	Long	Order Id as returned by the backend system.
action	A	m	1	Char(3)	Defines if the pre-arranged order is accepted or rejected by the reacting party. Valid values: **"ACC"**: The pre-arranged trade is accepted **"REJ"**: The pre-arranged trade is rejected
revisionNo	A	m		Long	The latest revision number of the order must be provided by the client. In case the backend system has another revision number, it will reject the request with an ErrResp.
clgAcctId	A	c		Integer	Clearing account Id Mandatory if the clearing house capability is set-up on exchange. See SystemInfoResp(chapter 6.1.5)
StandardHeader	*SE*	*m*		*Structure*	*Standard header of each message. Please see 5.1.3.*

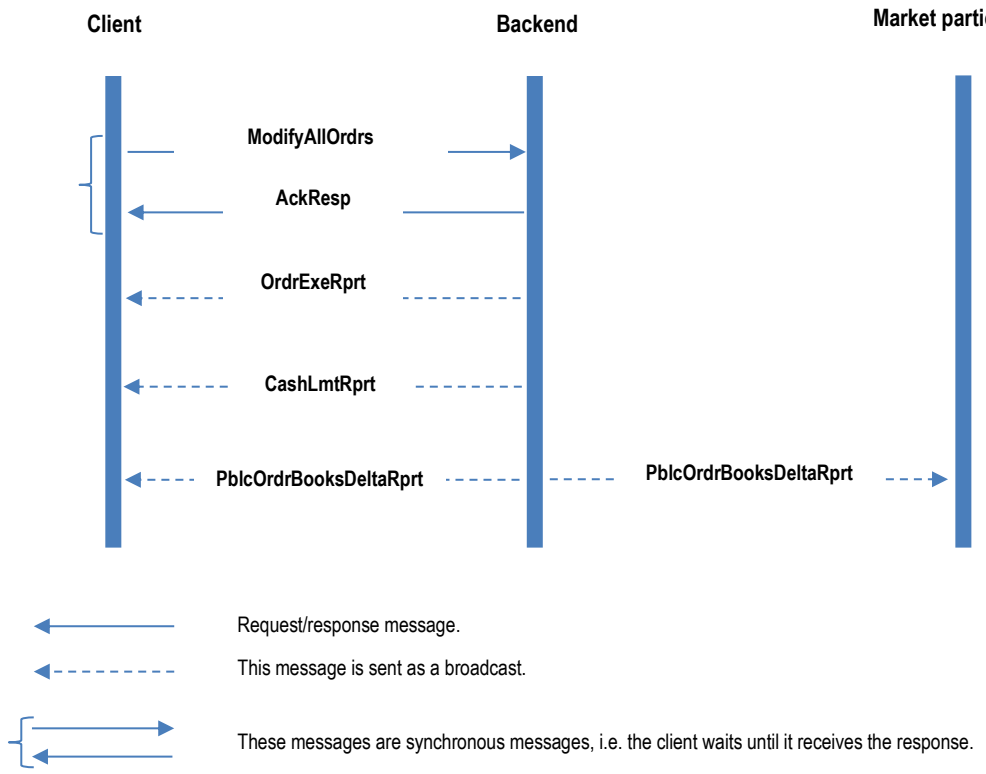
Table 15: The message layout of a Pre-Arranged Order Processing message.

6.2.6 Modify All Orders (ModifyAllOrdrs)

ModifyAllOrdrs	
Type:	Management Request
Roles:	Trader, Market Operation
Routing Keys:	`m7.request.management`

The Modify All Orders message is used to delete or deactivate all orders belonging to a member, an account or a trader. Only one of those attributes should be filled with a proper value. In case of an account also specific products can be defined to be taken into account.

The message flow generated by this request is depicted in the following diagram:



XML Tag	Type	m/o	No.	Data Type	Short description	
ModifyAllOrdrs	SE			Structure		
mbrId	A	c		String(5)	Unique identifier of a member.	One and only one of these attributes must be supplied.
usrId	A	c		Integer	Unique identifier of a user.	
acctId	A	c		Char(32)	Unique identifier of an account.	
dlvryAreaId	CE	o	O..n	String	Orders for the given usrId and list of DA's in dlvryAreaId will be Deactivated or Deleted This element can only be supplied when usrId is provided in the message. If left out all delivery areas assigned to usrId are affected	
ordrModType	A	m		Char(4)	Modification type for the orders: **"ACTI"**: Activate all orders. Already active orders are ignored. **"DEAC"**: Deactivates (hibernates) all orders. Hibernated orders are removed from the order book but are still available for modification or activation in the own orders list. **"DELE"**: Deletes all orders.	
inclPreArranged	A	m		Boolean	Specifies, if pre-arranged orders should be modified or not.	
StandardHeader	*SE*	*m*		*Structure*	*Standard header of each message. Please see 5.1.3.*	
prodName	CE	o	0.. 1000	String(255)	Only orders for the given products will be modified. This element can only be supplied when an account is provided.	

Table 16: The message layout of a Modify All Orders message.

6.2.7 Order Limit Request (OrdrLmtReq)

OrdrLmtReq	
Type:	Inquiry Request
Roles:	Trader, Market Operation, Broker, Market Maker
Routing Keys:	`m7.request.inquiry`

The OrdrLmtReq is used to retrieve the current Order Limits valid for the member to which the logged in trader is assigned.

XML Tag	Type	m/o	No.	Data Type	Short description
OrdrLmtReq	SE	m	1	Structure	
StandardHeader	*SE*	*m*		*Structure*	*Standard header of each message. Please see 5.1.3.*
prodName	CE	o	0.. 50	String(255)	ProductName
mbrId	A	o		String(5)	Member Id. In case of no member ID is passed, the appropriate member will be user will be returned. - for trader user his current member (= member the user belongs to) - for broker user his current member and his assigned members - for admin member all active members

Table 17: The message layout of an Order Limit Request message

6.2.8 Order Limit Report (OrdrLmtRprt)

OrdrLmtRprt	
Type:	Inquiry Response; Broadcast
Response to:	OrdrLmtReq (sent to the private response queue see 3.1)
Broadcast:	Yes
Broadcast Routing Keys:	`<schema-version>.mbr.<mbrId>`
Broadcast audience	`All traders from a particular member` `Admins`
Roles:	`Trader, Market Operation, Broker, Market Maker`

The OrdrLmtResp is returned as a response to the OrdrLmtReq or as a broadcast as a result of an event that changes an Order Limit. The message lists all the Order entry limits of the inquired member.

These limits are then checked at every order entry process for appropriate member.

XML Tag	Type	m/o	No.	Data Type	Short description
OrdrLmtResp	SE	m	1	Structure	
StandardHeader	*SE*	*m*		*Structure*	*Standard header of each message. Please see 5.1.3.*
MbrList	SE	m	1	Structure	List of the members
Mbr	SE	m	0..N	Structure	Member element
mbrId	A	m		String(5)	Member Id.
OrdrLmtList	SE	m	1	Structure	List of order limits
OrdrLmt	SE	m	0..N	Structure	Order limit element
lmtId	A	m		Long	The (internal) limit ID, the primary key.
revisionNo	A	m		Long	This value is increased every time the order limit is changed.
prodName	A	m		String(255)	product name
maxAmount	A	m		Long	Maximum amount allowed for order entry. The price decimal shift of the product applies. The value 0 is returned if amount validation has to be skipped during order entry.
maxQty	A	m		Long	Maximum quantity allowed for orders submitted in contracts belonging to the product

Table 18: The message layout of an Order Limit Response message

6.3 Trade Maintenance

6.3.1 Trade Recall Request (TradeRecallReq)

TradeRecallReq	
Type:	Management Request
Roles:	Trader, Market Operation
Routing Keys:	`m7.request.management`

This message is used to request a recall of a miss-trade. Market Operation will be informed about the trade recall request after successful submission. The The message flow is shown in the chart below.

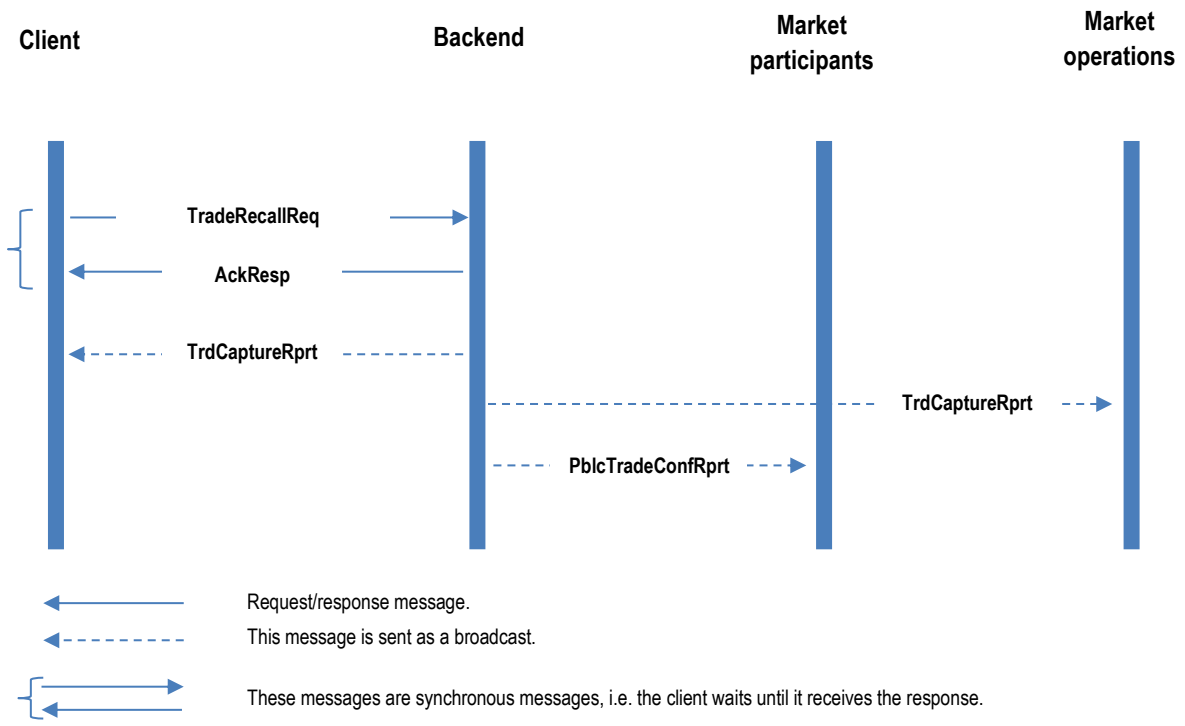


Figure 5: The message flow triggered by a Trade Recall Request.

The message required for a trade recall request contains only the trade identifier and revision number of the affected trade.

Note: if trade for which recall is requested has filled parentTradeId, the trade with tradeId=parentTradeId and also all other trades with the same parentTradeId will be recalled simultaneously.

XML Tag	Type	m/o	No.	Data Type	Short description
TradeRecallReq	SE	m		Structure	
StandardHeader	*SE*	*m*		*Structure*	*Standard header of each message. Please see 5.1.3.*
tradeId	A	m		Long	Trade Id of the trade to be recalled.
revisionNo	A	m		Long	The latest revision number of the trade must be provided by the client. In case the backend system has another revision number, it will reject the request with an ErrResp.

Table 19: The message layout of a Trade Recall Request.

6.3.2 Prearranged Trade Entry (PrearrangedTradeEntry)

PrearrangedTradeEntry	
Type:	Management Request
Roles:	Market Operation, Broker
Routing Keys:	`m7.request.management`

The message is used by broker to enter the prearranged trade. It is used for On Exchange prearranged trades and private and confidential trades.

Order Execution Reports and Trade Capture Reports are generated as a result of the trade.

XML Tag	Type	m/o	No.	Data Type	Short description
PrearrangedTradeEntry	SE			Structure	
contractId	A	m		Long	Contract Id.
px	A	m		Long	Price of the trade. (see 5.1.8)
qty	A	m		Integer	Quantity of the trade (see 5.1.7)
type	A	m		Char(1)	"**E**": On exchange prearranged trade "**N**": Private and confidential trade Note: other order types are not allowed in this message.
txt	A	o		String(250)	Optional text
clgHseCode	A	c		String(255)	Clearing House code of the trade Mandatory if the clearing house functionality is set-up on exchange. See SystemInfoResp(chapter 6.1.5)
Buy	SE	m	1	Structure	Defines buy side of the order
traderUserId	A	m		Integer	trader identification
acctId	A	m		String(32)	Account identification
clearingAcctType	A	o		Char(2)	Defines if the order is entered on its own account, or as an agent. For valid values please refer to values from attribute allowedClearingAcctTypes in the SystemInfoResp message (i.e."A,P" for spot markets). See chapter 6.1.5

	dlvryAreaId	A	o		String(16)	Delivery area identification
	openCloseInd	A	c		Char(1)	Mandatory for Futures and Cross product spreads. For other Commodities the value is not used. Valid values: "**O**": Open position indicator "**C**": Close position indicator
	brokerUserId	A	o		Integer	Broker user identification
	clgAcctId	A	c		Integer	Clearing account id Mandatory if the clearing house functionality is set-up on the exchange. See SystemInfoResp(chapter 6.1.5)
Sell		SE	m	1	Structure	Defines sell side of the trade
	traderUserId	A	m		Integer	Trader identification
	acctId	A	m		String(32)	Account identification
	clearingAcctType	A	o		Char(2)	Defines if the order is entered on its own account, or as an agent. For valid values please refer to values from attribute allowedClearingAcctTypes in the SystemInfoResp message (i.e."A,P" for spot markets). See chapter 6.1.5
	dlvryAreaId	A	o		String(16)	Delivery area identification
	openCloseInd	A	o		Char(1)	Mandatory for Futures and Cross product spreads. For other Commodities, the value is not used. Valid values: "**O**": Open position indicator "**C**": Close position indicator
	brokerUserId	A	o		Integer	Broker user identification
	clgAcctId	A	c		Integer	Clearing account id Mandatory if the clearing house functionality is set-up on exchange. See SystemInfoResp(chapter 6.1.5)

Table 20: The message layout of a Prearranged Trade Entry.

6.4 Market Information

6.4.1 Retrieval of Public Order Book information

The public order book of a contract is built up by performing the following steps:

- Retrieve the initial data set at the start of a user session using the Public Order Books Request (PblcOrdrBooksReq). All active orders in the order book at the time of request are returned.
- During the session, process the Public Order Books Delta Responses (PblcOrdrBooksDeltaRprt) that contain all of the updates to the order books.

PblcOrdrBooksDeltaRprt messages with an order book revision number smaller than the ones received in the PblcOrdrBooksResp must be ignored.

6.4.2 Public Order Books Request (PblcOrdrBooksReq)

PblcOrdrBooksReq	
Type:	Inquiry Request
Roles:	<ALL>
Routing Keys:	`m7.request.inquiry`
Request Limits:	14/70

The Public Order Books Request is used to retrieve the local view of the public order book.

It is possible to request the order book for a dedicated contract, in a given delivery area, or for all active contracts of a list of products. Either the contract and delivery area, or the list of products must be defined in the request.

A Public Order Book Request always returns a Public Order Books Response containing the complete information of the requested order books. M7 filters out orders for remote contracts from XBID that cannot be traded locally.

Note that the purpose of the request is to get the initial state of the public order books. Subsequent updates of the public order books are sent using the Public Order Books *Delta* Report message, listing all modified orders. Client applications should process these delta reports to keep their public order book view up to date. Any Public Order Books *Delta* Report messages in which the order book revision numbers are smaller than those in the Public Order Books Request must be ignored.

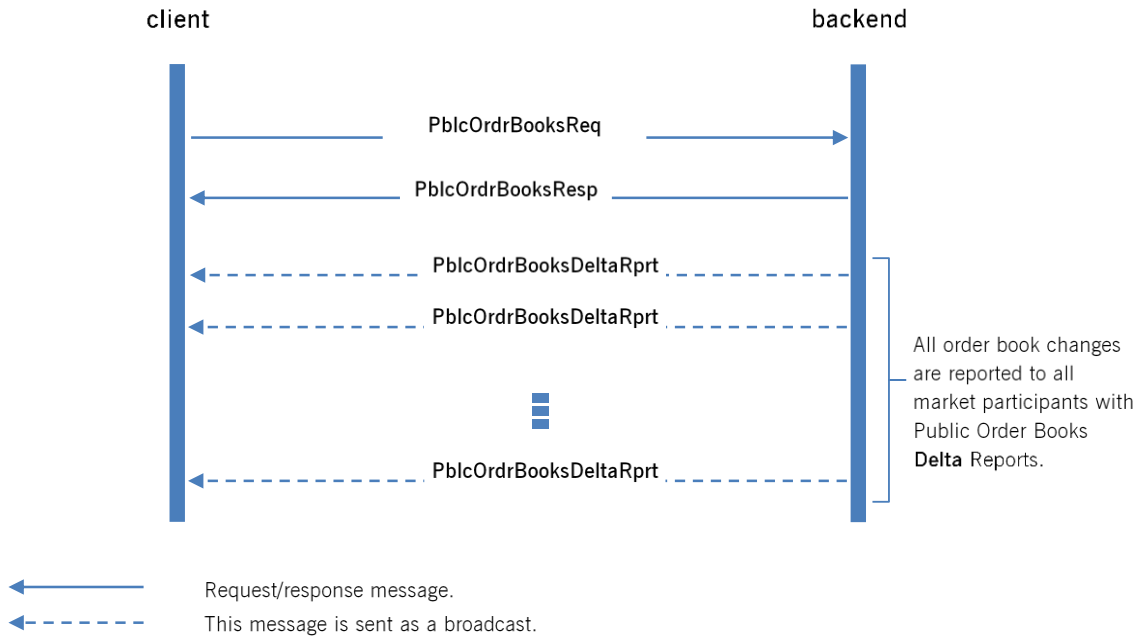


Figure 6: The message flow after a public order book request.

XML Tag	Type	m/o	No.	Data Type	Short description
PblcOrdrBooksReq	SE		1	Structure	
StandardHeader	*SE*	*m*		*Structure*	*Standard header of each message. Please see 5.1.3.*
contractType	A	o		Char(3)	Defines which kind of contracts should be retrieved: Possible values are: **"ALL"** – All kind of contracts (pre-defined and user-defined) **"PDC"** – Only pre-defined contracts **"UDC"** – Only user-defined contracts This attribute is ignored when a contractId is specified.
openCloseInd	A	o		Char(1)	Valid values: **"O":** Open position indicator **"C":** Close position indicator
prodName	CE	c	0.. 1000	String(255)	List of product names. All order books for these products are returned. The delivery area and/or contract type may be specified to filter the result. If no contract ID is given, at least one product name must be provided.

XML Tag	Type	m/o	No.	Data Type	Short description
contractId	CE	c	0..1	Integer	The Contract ID, which defines the order book to be retrieved. If specified, a delivery area must be specified and the contractType attribute is ignored. If no product name is given, at least one contract ID must be provided.
dlvryAreaId	CE	c	0.. 1000	String(16)	The delivery areas for which the order book(s) should be retrieved. Mandatory when a contractId is specified. If it is not specified, all applicable orderbooks will be returned.

Table 21: The message layout of a Public Order Books Request.

6.4.3 Public Order Books Response (PblcOrdrBooksResp)

PblcOrdrBooksResp	
Type:	Inquiry Response
Response to:	PblcOrdrBooksReq (sent to the private response queue see 3.1)
Broadcast:	No
Broadcast Routing Keys:	---
Roles:	<ALL>

The public order book is returned to the client as a result of a Public Order Books Request, and gives a complete overview of the requested public order book at the time of the request. In the case of remote products, M7 filters out orders using the messages that are broadcast by XBID for contracts that are not in-a trading phase according to the local schedule.

Subsequent changes to the public order books as a result of an order entry, modification, deletion or execution, or a change in cross border capacity are reported using a Public Order Books Delta Report. The Public Order Books Response and the Public Order Books Delta Report share the same Message Payload format.

The Public Order Books Response is sent to the private response queue of the requester, see 3.1.

XML Tag	Type	m/o	No.	Data Type	Short description
PblcOrdrBooksResp	SE	m	1	Structure	
StandardHeader	*SE*	*m*		*Structure*	*Standard header of each message. Please see 5.1.3.*
OrdrbookList	SE	o	0..1		
OrdrBook	SE	o	0..n	Structure	
contractId	A	m		Integer	The contract Id of the underlying contract.
dlvryAreaId	A	m		String(16)	Delivery Area to which the attached order books refer.
lastPx	A	o		Long	Last traded price (see 5.1.8).
lastQty	A	o		Integer	Last traded quantity.
totalQty	A	o		Long	The total quantity traded during this trading session.
lastTradeTime	A	o		DateTime	Timestamp of the last execution.
pxDir	A	o		Integer	Defines the direction of the price movement. Valid values are: -1: Price has decreased 0: Price is unchanged 1: Price has increased
revisionNo	A	m		Long	This value is increased in case of any change in the order book. **Please note**: the revision numbers of the order books are stored in memory only (not persisted) on backend side. After a restart of the backend system, the revision numbers of the order books will start again from 0.
highPx	A	o		Long	The highest traded price since the start of the trading period. The price format is described in 5.1.8.
lowPx	A	o		Long	The lowest traded price since the start of the trading period. The price format is described in 5.1.8.
surplusBid	A	o		Integer	The surplus determined during the auction phase on the Bid Side
surplusAsk	A	o		Integer	The surplus determined during the auction phase on the Ask Side
indicativePx	A	o		Long	The price determined during the auction phase of the contract. The price format is described in 5.1.8.
SellOrdrList	SE	o	0..1	Structure	

XML Tag	Type	m/o	No.	Data Type	Short description
OrdrBookEntry	SE	o	0..n	Structure	
ordrId	A	m		Long	The Order Id as determined by the backend system. If the order was entered for a remote product, the field contains: • the local Order ID (not the one from the XBID system) for the orders of own PX, or • the remote Order ID (the one from the XBID system) for the orders of other PXs.
qty	A	m		Integer	The quantity of the order which is exposed in that delivery area. This value may be different for one order depending on the observed delivery area and the available cross border capacity.
px	A	m		Long	The limit price of the order. The price format is described in 5.1.8.
ordrEntryTime	A	m		DateTime	The timestamp of the order.
ordrExeRestriction	A	o		Char(4)	Execution restriction of the order. This attribute is set only in the case of AON orders (value = "AON"). See 6.2.1 for more information and the possible values.
ordrType	A	o		Char(1)	**"O":** Regular limit order(default value) **"B":** User defined block order. **"L":** Balance order. **"C":** Indicative order.
openCloseInd	A	o		Char(1)	Mandatory for Futures and Cross product spreads. For other Commodities this value is not used. Valid values: **"O":** Open position indicator **"C":** Close position indicator
clgHse	SE	o	0..n	Structure	List of the Clearing House elements The priority order will be the same as the order of the Clearing House in the xml message. The Clearing House specified first will have the top priority, and the Clearing House specified at the end of the file will have the least priority.
clgHseCode	A	m		String(255)	Clearing House Code
BuyOrdrList	SE	o	0..1	Structure	
OrdrBookEntry	SE	0	0..n	Structure	
ordrId	A	m		Long	The Order Id as determined by the backend system. If the order was entered for a remote product, the field contains: • the local Order ID (not the one from the XBID system) for the orders of own PX, or • the remote Order ID (the one from the XBID system) for the orders of other PXs.
qty	A	m		Integer	The quantity of the order which is exposed in that delivery area. This value may be different for one order depending on the observed delivery area and the available cross border capacity.
px	A	m		Long	The limit price of the order. The price format is described in 5.1.8.
ordrEntryTime	A	m		DateTime	The timestamp of the order.
ordrExeRestriction	A	o		Char(4)	Execution restriction of the order. This attribute is set only in the case of AON orders (value = "AON"). See 6.2.1 for more information and the possible values.
ordrType	A	o		Char(1)	**"O":** Regular limit order (default value) **"B":** User defined block order. **"L":** Balance order. **"C":** Indicative order.
openCloseInd	A	o		Char(1)	Mandatory for Futures and Cross product spreads. For other Commodities this value is not used.. Valid values: **"O":** Open position indicator **"C":** Close position indicator
clgHse	SE	o	O..n	Structure	List of the Clearing House elements

XML Tag	Type	m/o	No.	Data Type	Short description
					The priority order will be the same as the order of the Clearing House in the xml message. The Clearing House specified first will have the top priority, and the Clearing House specified at the end of the file will have the least priority.
clgHseCode	A	m		String(255)	Clearing House Code

Table 22: The message layout of a Public Order Books Response and a Public Order Books Delta Report.

6.4.4 Public Order Books Delta Report (PblcOrdrBooksDeltaRprt)

PblcOrdrBooksDeltaRprt	
Type:	Broadcast
Response to:	n/a
Broadcast:	Yes
Broadcast Routing Keys:	`<schema-version>.prddlvr.<prodName>.<dlvryAreaId>`
Broadcast audience	`All users with the assignment of a particular product and delivery area.`
Roles:	`<ALL>`

The Public Order Books Delta Report is sent to the client as a result of any change in the order book as a result of an order entry, modification, deletion or execution or a change in cross border capacity. In the event that local trading for a remote product is closed, M7 does not forward the Public Order Books Delta Report received from XBID.

It contains a list of orders that have been added to the market, or changed as a result of the above mentioned actions. A quantity of 0 indicates that the order has to be removed from the order book.[7]

The message layout is shared with the Public Order Books Response (see 6.4.3 above).

6.4.5 Cash Limit Request (CashLmtReq)

CashLmtReq	
Type:	Inquiry Request
Roles:	Trader, Market Operation, Broker, Market Maker, Clearing user
Routing Keys:	`m7.request.inquiry`
Request Limits:	14/70

The Cash Limit Request is used to retrieve the current cash trading limit. The cash limit is used to limit the open financial risk position of a member. It is calculated on a member level and is valid for all traders belonging to that member.

The principal use of the Cash Limit Request is to obtain the cash limit at the start of a session, or after a communication breakdown. Subsequent changes to the cash limit are broadcast automatically by the backend. The diagram below shows the typical The message flow during a user session in respect of the Cash Limit Request and Cash Limit Report messages.

[7] The behaviour of the message depends on which timer (contract expiry timer vs. delivery interval closure timer) runs first. In the event that the delivery interval closes before the contract has expired, a Public Order Books Delta Report will be sent for the delivery areas in which orders can no longer match, with the quantity of 0 indicating the order's removal from these delivery areas. Once the contact expires, Public Order Books Delta Reports for these delivery areas with a quantity of 0 will not be re-distributed.

In case the contract expires before delivery interval closes, a Public Order Books Delta Report with the quantity of 0 for all orders will be sent out for all delivery areas. The subsequent delivery interval closure will not trigger the distribution of the Public Order Books Delta Reports.

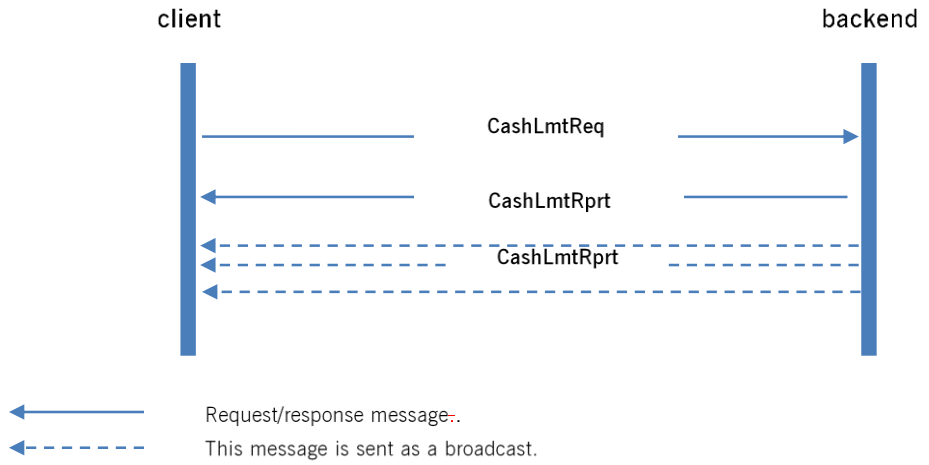


Figure 7: The message flow as a result of a trading limit request.

The trading limit request requires only the member Id as a parameter.

XML Tag	Type	m/o	No.	Data Type	Short description
CashLmtReq	SE			Structure	
StandardHeader	*SE*			*Structure*	*Standard header of each message. Please see 5.1.3.*
mbrId	A	m		String(5)	Member Id for which the trading limit is requested.

Table 23: The message layout of a Cash Limit Request.

6.4.6 Cash Limit Report (CashLmtRprt)

CashLmtRprt	
Type:	Inquiry Response; Broadcast
Response to:	CashLmtReq (sent to the private response queue see 3.1)
Broadcast:	Yes
Broadcast Routing Keys:	`<schema-version>.mbr.<mbrId>`

Broadcast audience:	`All users from a particular member`
	`Admins`
	`Brokers with assigned members`
	`Clearing users`

Roles:	`Trader, Market Operation, Broker, Market Maker, Clearing user`

The Cash Limit Report is provided in response to a Cash Limit Request or a broadcast that is a result of an event that changes the cash limit (Limit creation, modification, deletion or reset), as well as after the housekeeping of cash limits (deleting limits that were set up for past dates).

It is not sent during the order management requests (order entry, order execution). See the The message flow diagram above (Cash Limit Request).

The message lists all of the cash limits of the desired Member, as well as the current cash limit with its revision.

When the Cash Limit Report is sent as a response, it is sent to the private response queue of the requesting user, see 3.1. All subsequent updates are sent as a broadcast so that all traders of the member are up to date.

XML Tag	Type	m/o	No.	Data Type	Short description
CashLmtRprt	SE	m		Structure	
StandardHeader	*SE*	m		*Structure*	*Standard header of each message. Please see 5.1.3.*
mbrId	A	m		String	Member Id.
CurrentCashLmtList	SE	m		Structure	
CurrentCashLmt	SE	o	0..N	Structure	
currentCashLmtRevision	A	m		Long	The revision of the current cash limit.
currentCashLmt	A	m		Long	The value of the current cash limit.
currency	A	m		Currency Type (string)	The currency of the cash limit (EUR/GBP). The default value is EUR.
CashLmtList	SE	m		Structure	
CashLmt	SE	o	0..50	Structure	
revisionNo	A	m		Long	This value is increased every time the cash limit is changed.
cashLmt	A	m		Long	The cash limit that is available for the member.
decShft	A	m		Integer	The decimal shift that has been used to determine the cash limit in EUR. If this value equals to 2, the value in the cash limit corresponds to Eurocent (EURO/100). If this value equals to 4, the value in the cash limit corresponds to 1/100th Eurocent.
currency	A	m		Currency Type (string)	The currency of the cash limit (EUR/GBP). The default value is EUR.
lmtId	A	m		Long	The (internal) limit ID, the primary key.
state	A	m		String(4)	The status of the limit entry. Allowed values: "**ACTI**" - The limit is active
startDate	A	o		Date	The first date when the limit is active.
endDate	A	o		Date	The last date when the limit is active.
externalLmtId	A	o		Long	The external identifier of a limit.
externalVersion	A	o		Long	The external version of a limit.

Table 24: The message layout of a Cash Limit Report.

6.4.7 Cash Limit Delta Report (CashLmtDeltaRprt)

CashLmtDeltaRprt	
Type:	Broadcast
Response to:	--
Broadcast:	Yes
Broadcast Routing Keys:	`<schema-version>.mbr.<mbrId>`

Broadcast audience:	`All users from a particular member`
	`Admins`
	`Brokers with assigned members`
	`Clearing users`

Roles:	`Trader, Market Operation, Broker, Market Maker, Clearing user`

The Cash Limit Delta Report broadcasts the new current cash limit value in the event that the limit changed (e.g. order entry, order execution). All traders/brokers of the Member receive this message.

XML Tag	Type	m/o	No.	Data Type	Short description
CashLmtDeltaRprt	SE	m		Structure	
StandardHeader	*SE*	m		*Structure*	*Standard header of each message. Please see 5.1.3.*
mbrId	A	m		String(5)	The Member ID for whom the limit is set.
revisionNo	A	m		Long	This value is increased every time the current cash limit is changed.
decShft	A	m		Integer	The decimal shift that has been used to determine the cash limit in EUR. If this value equals to 2, the value in the cash limit corresponds to Eurocent (EURO/100). If this value equals to 4, the value in the cash limit corresponds to 1/100th Eurocent.
cashLmt	A	m		Long	The cash limit available to the member.
currency	A	m		Currency Type (string)	The currency of the cash limit (EUR/GBP). The default value is EUR.

Table 25: The message layout of a Cash Limit Delta Report.

6.4.8 Commodity Limit Request (CommodityLmtReq)

CommodityLmtReq	
Type:	Inquiry Request
Roles:	Trader, Market Operation, Broker, Market Maker
Routing Keys:	`m7.request.inquiry`
Request Limits:	1/10

The Commodity Limit Request is used to retrieve the current commodity trading limits of a member. The commodity limit is used to prohibit short selling of a member. The commodity limit is product specific, not all products offer this feature. It is calculated on a member level for every product and is valid for all traders belonging to that member.

The principal use of the Commodity Limit Request is to obtain the commodity limits at the start of a session or after a communication breakdown. Subsequent changes to the commodity limit are broadcast automatically by the back end.

The following diagram shows a typical The message flow during a user session for the Commodity Limit Request and Commodity Limit Report messages.

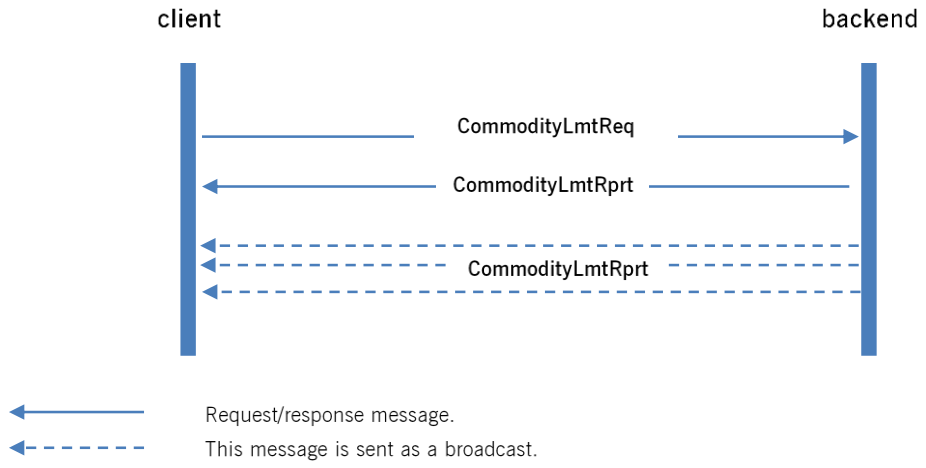


Figure 8: The message flow as a result of a commodity limit request.

The commodity trading limit request only requires the member Id as a parameter.

XML Tag	Type	m/o	No.	Data Type	Short description
CommodityLmtReq	SE			Structure	
StandardHeader	*SE*			*Structure*	*Standard header of each message. Please see 5.1.3.*
mbrId	A	m		String(5)	The member Id for which the trading limit is requested.

Table 26: The message layout of a Commodity Limit Request.

6.4.9 Commodity Limit Report (CommodityLmtRprt)

CommodityLmtRprt	
Type:	Inquiry Response; Broadcast
Response to:	CashLimitReq (sent to the private response queue see 3.1)
Broadcast:	Yes
Broadcast Routing Keys:	`<schema-version>.mbr.<mbrId>`

Broadcast audience:	`All users from a particular member`
	`Admins`
	`Brokers with assigned members`

Roles:	`Trader, Market Operation, Broker, Market Maker`

The Commodity Limit Report is returned in response to a Commodity Limit Request, or a broadcast that is sent as a result of an event that changes the commodity limit (for products eligible for the commodity risk control functionality). See the diagram in the previous subsection (6.4.8 Commodity Limit Request (CommodityLmtReq))

When the Commodity Limit Report is sent as a response, it is sent to the private response queue of the requesting user, see 3.1. All updates are sent as a broadcast so that all of the traders of the member are up to date.

XML Tag	Type	m/o	No.	Data Type	Short description
CommodityLmtRprt	SE	m	1	Structure	
StandardHeader	*SE*	*m*		*Structure*	*Standard header of each message. Please see 5.1.3.*
CommodityLmtList	SE	o	0..1	Structure	
CommodityLmt	SE	o	0..n	Structure	
commodityLmt	A	m		Integer	Commodity Limit.
mbrId	A	m		String(5)	The member Id of the requesting member.
contractId	A	m		long	The contract to which the commodity limit applies
revisionNo	A	m		Long	This value is increased every time the trading limit is changed and a new Trading Limit Response is issued.

Table 27: The message layout of a Commodity Limit Report.

6.4.10 Message Request (MsgReq)

MsgReq	
Type:	Inquiry Request
Roles:	<ALL>
Routing Keys:	`m7.request.inquiry`
Request Limits:	14/70

This inquiry message is used to retrieve the public and/or private messages issued by the backend system in the past.

The principal use of this request is to obtain a list of recent messages at login, or after a communication breakdown between the client application and the backend. After the login, the client application will receive all messages (private and public) automatically as they are broadcast by the backend.

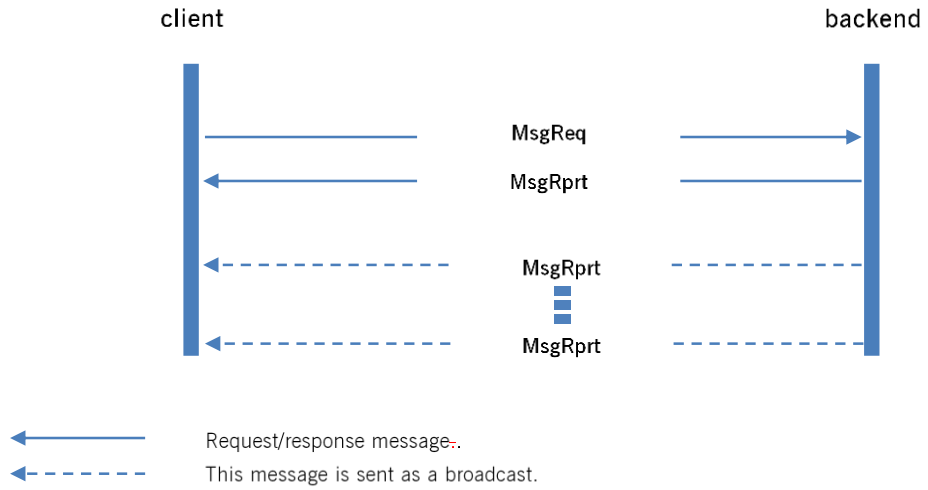


Figure 9: The message flow induced by a message request.

It is necessary to define a period for which the messages are to be retrieved, and to filter the message type by setting the appropriate value in the type field.

XML Tag	Type	m/o	No.	Data Type	Short description
MsgReq	SE		1	Structure	
StandardHeader	*SE*	*m*		*Structure*	*Standard header of each message. Please see 5.1.3.*
acctId	A	c		String(32)	The accountId (balancingGroupEIC) to which the messages are sent. The response will also include messages which are sent to the product and the delivery area routing keys that have been assigned to this account. For Market Operation users, this attribute is optional, in which case the messages sent to all accounts will be returned. For other users this attribute is mandatory.
endDate	A	m		DateTime	The timestamp defining the point in time that the messages should be retrieved.
startDate	A	m		DateTime	The timestamp defining from which point in time the messages should be retrieved. It is possible to only retrieve messages from the last 5 days: (current date - 5 days) < frm
type	A	m		Char(7)	This defines what kind of messages are returned, allowing the messages to be filtered on a request level. Valid Values: **"ALL":** Return all messages. **"PUBLIC":** Return only public messages. **"PRIVATE":** Return only private messages.

Table 28: The message layout of a Message Request.

6.4.11 Message Report (MsgRprt)

MsgRprt	
Type:	Inquiry Response, Broadcast
Response to:	MsgReq (sent to the private response queue see 3.1)
Broadcasted:	Yes

Broadcast Routing Keys:	`<schema-version>.prvt.bg.msg.<acctId>`
	`<schema-version>.prd.<prodName>`
	`<schema-version>.dlvr.<dlvryAreaId>`
	`<schema-version>.mkt.<marketAreaId>`
	`<schema-version>.mbr.<memberId>`

	<schema-version>.public
Broadcast audience:	Depending on particular message type.
Roles:	<All>

The Message Report is sent as a response to a Message Request to the private response queue of the requesting user, but it can also be broadcast without a prior request, for example

- when a user is logged out from an exchange;
- when a user is re-connecting after a connection loss;
- when an Admin is sending a text message to market participants
- when an order failed to be transferred during an automated order transfer (see *DFS160a*).

In case MsgRprt is response to MsgReq, the list of messages can be truncated given startDate and endDate exceeds the maximum number of messages configured in the application. In such a situation, refine startDate and endDate in MsgReq.

XML Tag	Type	m/o	No.	Data Type	Short description
MsgRprt	SE	m	1	Structure	
truncated	A	o	0..1	Boolean	If set to true, the MsgList has been truncated to the maximum number of messages (specified in the configuration).
StandardHeader	*SE*	*m*		*Structure*	*Standard header of each message. Please see 5.1.3.*
MsgList	SE	o	0..1		
Msg	SE	o	0..n		
msgId	A	m		Long	The message Id assigned by the backend system.
type	A	m		String(7)	Defines the message type. Valid Values: **"PUBLIC":** The message is a public message. **"PRIVATE":** The message is a private message.
messageCode	A	m		Integer	Message code of the message
prod	A	o		String(255)	Underlying product.
acctId	A	o		String(32)	Id of an account.
timestmp	A	m		DateTime	The timestamp of the message as assigned by the backend system.
svrty	A	m		String(3)	Severity of the message: **"URG":** Urgent message. **"ERR":** Error. **"HIG":** High priority message. **"MED":** Medium priority message. **"LOW":** Low priority message.
txt	A	m		String(300)	Message text.
sellDlvryAreaId	A	o		String(16)	In case of an order execution, this field contains the delivery area of the sell side.
buyDlvryAreaId	A	o		String(16)	In case of an order execution, this field contains the delivery area of the buy side.
mktSupervisionMsg	A	m		Boolean	This determines if the message has been sent by market supervision
nonPersistent	A	o		Boolean	If set to true, the message is not persisted.
VarList	SE	o..1		Structure	List of variables used in message txt attribute
Var	SE	0..n		Structure	Structure containing variable information
id	A	m		Integer	Identifier of variable within the message resource text (e.g. 6)
value	A	m		String(255)	Value of the variable (e.g. “T18-19”)

Table 29: The message layout of a Message Report.

6.4.12 Trade Capture Request (TradeCaptureReq)

TradeCaptureReq	
Type:	Inquiry Request
Roles:	Trader, Market Operation, Broker, Market Maker, Sales
Routing Keys:	m7.request.inquiry

Request Limits:	14/70

This message is used to retrieve trades created during the last five days:

- For *Traders,* the account must be specified and all private trades are returned. If not specified, empty TradeCaptureRprt is returned.
- For a *Market Operation* user or *Sales* user, the account is not needed (a given value will be ignored by the backend) and all trades in the system for the observation period are returned.

The observation period extends from 5 days in the past until the day after the current trading day

XML Tag	Type	m/o	No.	Data Type	Short description
TradeCaptureReq	SE				
StandardHeader	*SE*	*m*		*Structure*	*Standard header of each message. Please see 5.1.3.*
acctId	CE	c	0..n	String(32)	The selected account. For Market Operation and Sales users this attribute is optional and ignored by the backend.
startDate	A	m		DateTime	The start of the period for which the trades are retrieved. This value must fulfil the following conditions: • endDate – startDate <= 25 hours
endDate	A	o		DateTime	The end of the period for which the trades are retrieved. The following condition must be fulfilled: • endDate – startDate <= 25 hours If no end date is given, the backend will return all trades until midnight (CET/CEST for all products) of the start date.
dateMeaning	A	o		String(11)	This defines the meaning of the startDate and endDate parameters Valid values: **MATCH** (default): trades with a match event occurring between the startDate and endDate are inquired **LAST_UPDATE:** trades with a last update that has occurred between the startDate and endDate are inquired
includeCancelledAndRecalled	A	o		Boolean	If set to true, cancelled and recalled trades are also returned. The default value if the attribute is not present is **false**

Table 30: The message layout of a Trade Capture Request.

6.4.13 Trade Capture Report (TradeCaptureRprt)

TradeCaptureRprt	
Type:	Inquiry Response, Broadcast
Response to:	TradeCaptureReq (sent to the private response queue see 3.1)
Broadcasted:	Yes
Broadcast Routing Keys:	<schema-version>.hlftrd.<acctId> <schema-version>.trade.<mktAreaId>
Broadcast audience	Half trade: - Trader (owner of the order) and the other traders from his Balancing groups. - Broker with an assignment to a trader (owner of the order) - Market maker (owner of the order) and other traders from his Balancing groups. Full trade: - Admins - Sales - Settlement users - Clearing users
Roles:	Trader, Market Operation, Broker, Market Maker, Sales

The Trade Capture Report broadcasts private trade information as a result of the following actions:

- Order Execution
- Trade cancellation
- Trade recall request
- Trade recall request processing

The broadcasts are sent to the traders assigned to the accounts for which the related orders were entered as well as to the market operations users.

Trading participants will always receive half-trades containing only the information of the order entered by that trading participant:

- If the trading participant entered a SELL order she will see only the SELL side of the trade.
- If the trading participant entered a BUY order she will see only the BUY side of the trade.

The Trade Capture Report is sent to the trading participants with the routing key <schema-version>.hlftrd.<acctId>. Market Operation users will receive the complete trade information (both sell side and buy side). The Trade Capture Report is sent to Market Operation users with the routing key <schema-version>.trade.<mktAreaId>. In the case of a local trade (between 2 delivery areas in the same market area) the message will be sent only once.

When the Trade Capture Report is sent as a reply to the Trade Capture Request, it will be sent to the private response queue of the requesting user (a trade or market operations user), see 3.1.

The Trade Capture Report contains at most the trades created during the last X days, even if trades for a longer or different period are requested in the Trade Capture Request. The number of days represented by X is set using the system parameter contractStoreTimeInDays.

The visibility rules of the data are the same for both the broadcast and response messages.

XML Tag	Type	m/o	No.	Data Type	Short description
TradeCaptureRprt	SE	m	1	Structure	
StandardHeader	*SE*	*m*		*Structure*	*Standard header of each message. Please see 5.1.3.*
TradeList	SE	o	0..1		
Trade	SE	o	0..n	Structure	
tradeId	A	m		Long	The trade ID of the trade. In XBID-connected environments remote trades and LTS trades share the same trade ID sequence. Therefore a gap in LTS trade IDs may occur in the course of messages.
state	A	m		Char(4)	The current state of the trade. Valid values are: **"CNCL": The t**rade was cancelled by market operations. **"RREJ":** The requested Recall was rejected by market operations. **"RGRA": The** requested Recall was granted by market operations. **"RREQ": The r**ecall of this trade was requested. **"ACTI": The t**rade is active (this is the default value). **"CREQ"**: A cancellation was requested from the local market operations. **"CREJ"**: A cancellation was rejected by the global market operations. **"RSFA"**: The request was sent for approval to SOB (XBID).
contractId	A	m		Integer	Id of the underlying contract.
qty	A	m		Integer	The executed quantity.
px	A	m		Long	The execution price. The price format is described in 5.1.8.
execTime	A	m		DateTime	The execution date as assigned by the backend system.
revisionNo	A	m		Long	The revision number of the trade. With every change to the trade, the revision number is increased by one.
preArranged	A	m		Boolean	This defines if the trade was pre-arranged. "false": No pre-arranged trade "true": Pre-arranged trade
prearrangeType	A	o		String(3)	This attribute is mandatory for prearranged trades. "**OTC**": Over the counter trade "**OPT**": On exchange prearranged trade "**PNC**": Private and confidential trade
recallReqTime	A	o		DateTime	Date and time of a recall request.
recallGrantedTime	A	o		DateTime	Date and time when market operations granted the recall. Also used when the trade was cancelled.
recallRejectedTime	A	o		DateTime	Date and time when market operations rejected the recall.
latestRecallProcessTime	A	o		DateTime	Informs until when a recall request can be processed by market operations.
contractPhase	A	m		String	Indicates, in which contract phase the trade was executed: "CONT": Continuous trading "BALA": Balancing phase "AUCT": Auction phase "CLSD": Closed Phase "SDAT » : Same Delivery Area Treading phase
clgHseCode	A	o		String(255)	Clearing House code of the Trade
parentTradeId	A	o		Long	Filled in case of crosscontract trades – points to the parent trade id
decomposed	A	o		Boolean	Indicates whether the trade was decomposed to trades in underlying or parent products.
remoteTradeId	A	o		Long	Trade Id as returned by remote backend system (i.e. XBID SOB)
selfTrade	A	o		Boolean	Indicates whether the trade was done as a self trade (inside one balancing group or between two different balancing groups within one member) or not. A cross-NEMO trade is not marked as a self-trade.
Buy	SE	o	1	Structure	
clearingAcctType	A	o		Char(2)	Defines if the order is entered on its own account, or as an agent. For valid values please refer to values from attribute allowedClearingAcctTypes in the SystemInfoResp message (i.e."A,P" for spot markets). See chapter 6.1.5
dlvryAreaId	A	o		String(16)	Delivery Area to which the attached order books refer to.
acctId	A	o		String(32)	Account for which the order is entered.
ordrId	A	o		Long	Order Id of the buy side order.
txt	A	o		String(250)	Text of the buy side order.

XML Tag	Type	m/o	No.	Data Type	Short description
aggressorIndicator	A	o		Char(1)	Indicates whether the executed order was a trade aggressor or trade originator. **Valid values:** •"Y" - Trade aggressor •"N" - Trade originator •"U" - Unknown, for executed orders of remote products and data before migration. Default value.
usrCode	A	o		String(6)	User code of the user who entered the buy side order.
clOrdrId	A	o		String(40)	Client Order Id with a maximum length of 40 characters. This value is not modified by the backend and may be used by client applications to identify orders.
mbrId	A	o		String(5)	MemberID of the member who entered the buy side order.
openCloseInd	A	o		Char(1)	Mandatory for Futures product and Cross product spreads. For other Commodities is not present. Valid values: **"O":** Open position indicator **"C":** Close position indicator
brokerUserId	A	o		Integer	UsrId of the broker user
clgAcctId	A	o		Integer	Clearing account Id of the buy Order
remoteOrdrId	A	o		Long	Order Id as returned by remote backend system (i.e. XBID SOB)
Sell	SE	o	1	Structure	
clearingAcctType	A	o		Char(2)	Defines if the order is entered on its own account, or as an agent. For valid values please refer to values from attribute allowedClearingAcctTypes in the SystemInfoResp message (i.e."A,P" for spot markets). See chapter 6.1.5
dlvryAreaId	A	o		String(16)	The delivery Area to which the attached order books refer to.
acctId	A	o		String(32)	The account for which the order is entered.
ordrId	A	o		Long	The Order Id of the buy side order.
txt	A	o		String(250)	The text of the buy side order.
aggressorIndicator	A	o		Char(1)	This indicates whether the executed order was a trade aggressor or a trade originator. **Valid values:** •"Y" - Trade aggressor •"N" - Trade originator •"U" - Unknown, for executed orders of remote products and data before migration. Default value.
usrCode	A	o		String(6)	The user code of the user who entered the buy side order.
clOrdrId	A	o		String(40)	The client Order Id has a maximum length of 40 characters. This value is not modified by the backend and may be used by client applications to identify orders.
mbrId	A	o		String(5)	The MemberID of the member who entered the buy side order.
openCloseInd	A	o		Char(1)	This is mandatory for Futures and Cross product spreads. For other Commodities it is not used. Valid values: **"O":** Open position indicator **"C":** Close position indicator
brokerUserId	A	o		Integer	UsrId of the broker user
clgAcctId	A	o		Integer	The clearing account Id of the buy Order
remoteOrdrId	A	o		Long	The Order Id as returned by remote backend system (i.e. XBID SOB)

Table 31: The message layout of a Trade Capture Report.

6.4.14 Public Trade Confirmation Request (PblcTradeConfReq)

PblcTradeConfReq	
Type:	Inquiry Request
Roles:	Trader, Broker, Data Vendor
Routing Keys:	`m7.request.inquiry`
Request Limits:	14/70

This message is used to retrieve a list of public trades executed during a specified period. This period can only be 5 days in the past. A user is also able to send this request for products that are not tradable (both remote and local products).

XML Tag	Type	m/o	No.	Data Type	Short description
PblcTradeConfReq	SE	m		Structure	
StandardHeader	*SE*	*m*		*Structure*	*Standard header of each message. Please see 5.1.3.*
prodName	CE	o	0.. 1000	String(255)	Products for which the public trade confirmations are requested. If this is not entered, all products for which the user has access rights are returned.
startDate	A	m		DateTime	The start of the period for which the trades are retrieved. This value must fulfil the following conditions: endDate – startDate <= 25 hours
endDate	A	m		DateTime	The end of the period for which the trades are retrieved. The following condition must be fulfilled: endDate – startDate <= 25 hours

Table 32: The message layout of a Public Trade Confirmation Request.

6.4.15 Public Trade Confirmation Report (PblcTradeConfRprt)

PblcTradeConfRprt	
Type:	Inquiry Response, Broadcast
Response to:	PblcTradeConfReq (sent to the private response queue see 3.1)
Broadcasted:	Yes
Broadcast Routing Keys:	`<schema-version>.public.trade.<prodName>`
Broadcast audience	`All traders, brokers and data vendors with an assignment to traded product Admins`
Roles:	Trader, Market Operation

This message is broadcast to all trading participants who are assigned to the product in the event of a trade execution or cancellation, or as a response to a Public Trade Confirmation Request.

As a broadcast, it contains a list of the public trade confirmations for the triggering event (trade execution, recall or cancellation) where exchange is on buy or sell side.

As a response, it contains the list of all public trades where exchange is on buy or sell side for the requested product(s) and period. It is sent to the private response queue of the requester, see 3.1.

Pre-arranged trades are not part of the Public Trade Confirmation Report.

In the case of remote products, M7 filters out XBID broadcast messages for remote contracts that cannot be traded according to the local schedule.

XML Tag		Type	m/o	No.	Data Type	Short description
PblcTradeConfRprt		SE	m	1	Structure	
StandardHeader		*SE*	*m*		*Structure*	*Standard header of each message. Please see 5.1.3.*
TradeList		SE	o	0..1	Structure	
	PblcTradeConf	SE	o	0..n	Structure	
	tradeId	A	m		Long	Trade Id of the underlying trade.
	state	A	m		Char(4)	The current state of the trade. Valid values are: **"CNCL": The t**rade was cancelled by market operations. **"RREJ":** The requested Recall was rejected by market operations. **"RGRA": The r**equested Recall was granted by market operations. **"RREQ": The r**ecall of this trade was requested. **"ACTI": The t**rade is active (this is the default value). **"CREQ"**: The cancellation was requested from local market operations. **"CREJ"**: The cancellation was rejected by global market operations. **"RSFA"**: The request has been sent for approval to SOB (XBID).

XML Tag		Type	m/o	No.	Data Type	Short description
	contractId	A	m		Long	The contract Id of the traded contract.
	qty	A	m		Integer	The traded quantity.
	px	A	m		Long	The execution price in Eurocents (1 Euro = 100).
	tradeExecTime	A	m		DateTime	The trade execution time.
	revisionNo	A	m		Long	The revision number of the trade. This is increased by one every time the trade is changed.
	buyDlvryAreaId	A	o		String(16)	The delivery area of the buy side.
	sellDlvryAreaId	A	o		String(16)	The delivery area of the sell side.
	clgHseCode	A	o		String(255)	The Clearing House Code of the trade.
	selfTrade	A	o		Boolean	Indicates whether the trade was done as a self trade (inside one balancing group or between two different balancing groups within one member) or not. A cross-NEMO trade is not marked as a self-trade.

Table 33: The message layout of a Public Trade Confirmation Report.

6.4.16 Contract Information Request (ContractInfoReq)

ContractInfoReq	
Type:	Inquiry Request
Roles:	<ALL>
Routing Keys:	`m7.request.inquiry`
Request Limits:	14/70

This message is used to retrieve contract related information from the backend system.

If ContractInfoReq that has a startDate and endDate set, then ACTI, HIBE and IACT contracts are returned where the delivery interval or tradingDate are in an interval defined by the startDate and endDate request. The selected time interval when setting the startDate and endDate is limited to 25h.

If the requesting user does not have the proper rights for the requested contract, an ErrResp is returned.

XML Tag	Type	m/o	No.	Data Type	Short description
ContractInfoReq	SE			Structure	
StandardHeader	*SE*	*m*		*Structure*	*Standard header of each message. Please see 5.1.3.*
prodName	CE	o	0.. 1000	String(255)	The contract information for all contracts belonging to the given products is requested. If specified, the contractId element cannot be specified and the startDate and endDate attributes are mandatory.
contractId	CE	o	0..1	Long	The contract information for the given contractID is requested. If specified, the prodName element cannot be specified and the startDate and endDate attributes are ignored.
startDate	A	c		DateTime	Start date for requested contract mandatory if contractId is not specified
endDate	A	c		DateTime	End date for requested contract mandatory if contractId is not specified

Table 34: The message layout of a Contract Information Request.

6.4.17 Contract Information Report (ContractInfoRprt)

ContractInfoRprt	
Type:	Inquiry Response, Broadcast
Response to:	ContractInfoReq (sent to the private response queue see 3.1)
Broadcasted:	Yes
Routing Keys:	<schema-version>.prd.<prodName>
Broadcast audience:	All users with an assignment to a particular product.
Roles:	<ALL>

This returns detailed information linked to the requested contract(s) as a result of a Contract Information Request, see 3.1.

If the ContractInfoReq that has a startDate and an endDate are set, then ACTI, HIBE, STBY and IACT contracts are returned where the delivery interval or tradingDate are in an interval defined by the startDate and endDate from the request.

This message is also broadcast to indicate contract state changes (i.e. without a prior Contract Information Request).

XML Tag	Type	m/o	No.	Data Type	Short description
ContractInfoRprt	SE	m	1	Structure	
StandardHeader	*SE*	m		Structure	*Standard header of each message. Please see 5.1.3.*
ContractList	SE	o	0..1	Structure	
Contract	SE	o	0..n		
contractId	A	m		Long	The contract Id as defined by the backend system.
prod	A	m		String(255)	The underlying product.
name	A	m		String(255)	The contract name. This is used for display purposes.
longName	A	m		String(255)	The contract long name, containing additional information (delivery period).
dlvryStart	A	m		DateTime	The start of the delivery period.
dlvryEnd	A	m		DateTime	The end of the delivery period.
predefined	A	m		Boolean	A flag that indicates if a contract has been automatically created by the backend system, or if the contract was generated via the entry of a user-defined block order.
revisionNo	A	m		Long	The revision number of the contract. This value is increased by one every time the contract is modified.
state	A	m		Char(4)	The current state of the contract. The following values are allowed: **"HIBE":** Hibernate: The contract was manually deactivated by market operations. **"IACT":** The contract is inactive and not available for trading. **"ACTI":** The contract is active and available for trading. **"STBY":** The contract is waiting on an external event to become available for trading.
tradingPhase	A	m		Char(4)	**Note: This field is deprecated. Only the Trading Phase (tradingPhase) in the <DlvryAreaState> substructure should be regarded.** "**CLSD**": Trading in the contract is closed for the current trading day. The members will not be able to submit any new orders **"CONT"**: Trading in the contract is in continuous mode. **"AUCT"**: The contract is in the Auction phase. **"BALA"**: The contract is in the Balancing phase. **"SDAT":** The contract is in the Same Delivery Area Trading phase
duration	A	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.
actPoint	A	o	1	DateTime	The parameter gives the contract activation point (e.g. delivery start).
expPoint	A	o	1	DateTime	The parameter gives the contract expiration point (e.g. delivery end).
strikePx	A	o	1	Long	In case that the bespoke contract is an option, this field contains the strike price of the contract. The price format is described in 5.1.8.
CallPut	A	o	1	String(1)	In case that the bespoke contract is an option, this field contains the contract type. The following values are available: • '**C**' : Call • '**P**' : Put
optionExpPoint	A	o	1	DateTime	In case that the bespoke contract is option, this parameter gives an option expiration point.

XML Tag	Type	m/o	No.	Data Type	Short description
bespoke	A	o	1	Boolean	This defines if the contract is a bespoke contract.
delUnits	A	m		Double	This is the delivery unit of the respective product. In case of a product with the type User-Defined Delivery Period, this attribute is stored only on a contract level.
remoteContractId	A	o		Long	The contract Id as defined by the external backend system (i.e. XBID SOB)
DlvryAreaState	SE	o	0..n	Structure	This is used to define the state of an auto generated contract for a specific delivery area.
dlvryAreaId	A	m		String(16)	The Delivery Area ID.
state	A	o		Char(4)	The current state of the contract in the delivery area. The following values are allowed: **"HIBE":** Hibernated, the contract was manually deactivated for the delivery area by market operations. **"IACT":** The contract is inactive and not available for trading in the delivery area. **"ACTI":** The contract is active and available for trading in the delivery area. **"STBY":** The contract is waiting for a triggering event to become available for trading. Triggering events are: • The end of a trading phase of a global product (i.e. ContractInfoRprt has been received from XBID), in case there are no remote orders and its linked product has a longer trading phase than the remote product. No automatic order transfer is performed. • An automatic order transfer for a delivery area has been performed, or skipped/reverted due to the valid reasons (e.g. a disconnection from XBID). For more details on the AOT process please refer to *DFS160a*.
tradingPhase	A	o		Char(4)	"**CLSD**": Trading in the contract is closed for the current trading day. The members will not be able to submit any new orders **"CONT"**: Trading in the contract is in continuous mode. **"AUCT"**: The contract is in the Auction phase. **"BALA"**: The contract is in the Balancing phase. **"SDAT":** The contract is in the Same Delivery Area Trading phase
tradingPhaseStart	A	m		DateTime	The start date and time of the current/next trading phase in the delivery area.
tradingPhaseEnd	A	o		DateTime	The end date and time of the current/next trading phase in the delivery area.
UnderlyingContracts	SE	o	0..1	Structure	For bespoke contracts, this structure is used to define the underlying contracts of the bespoke product.
contractId	CE	m	0..n	Integer	The contract Id of the underlying contract leg, as defined by the backend system.
SpreadContracts	SE	o	0..1	Structure	
leg1ContractId	A	m	1	Long	The contract Id of the first underlying contract for Auto Combination contracts
leg2ContractId	A	m	1	Long	The contract Id of the second underlying contract for Auto Combination contracts

Table 35: The message layout of a Contract Information Report.

6.4.18 Product Information Request (ProdInfoReq)

ProdInfoReq	
Type:	Inquiry Request
Roles:	<ALL>
Routing Keys:	`m7.request.inquiry`
Request Limits:	14/70

This message is used to request the product details. It is possible to pass a list of product Ids in one message. If the user is requesting a product he does not have a right to, an ErrResp is returned.

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

StandardHeader	*SE*	*m*		*Structure*	*Standard header of each message. Please see 5.1.3.*
prodName	CE	o	0.. 1000	String(255)	

Table 36: The message layout of a Product Information Request.

6.4.19 Product Information Report (ProdInfoRprt)

ProdInfoRprt	
Type:	Inquiry Response
Response to:	ProdInfoReq (sent to the private response queue see 3.1)
Broadcasted:	Yes
Broadcast Routing Keys:	<schema-version>.prd.<prodName>
Broadcast audience:	All users with an assignment to a particular product.
Roles:	<ALL>

This report is used to return detailed product information. It is provided as a response to the Product Information Request, as well as being broadcast in the event of product modification.

If the prodName in the Product Information Request is set then:

For admin role, all products with a requested prodName(s) are contained in the message.

For other roles, the user's balancing group(s) product assignment are checked (see message AccntInfoRprt in chapter 6.6.8).

If the prodName in the Product Information Request is not set then:

For admin role, all products are contained in the message.

For other roles, the products that have assignment to the user's balancing group(s)are returned (see the message AccntInfoRprt in chapter 6.6.8).

In the *ProdCfgs* structure, the following key-value pairs are available:

Prod types:

- FUT: Futures
- CRO: Cross product spreads
- COM: Commodity, energy

Key	Value	Prod Type	Description
isin	true	FUT	The product has an ISIN.
isinValue		FUT	The value of the ISIN.
blockOrderProduct	true	COM	User defined block orders are allowed for this product.
blockMaximumHours		COM	Defines how many base contracts may be grouped together for user defined blocks.
dstBlockProduct	true	COM	Defines the treatment of contracts during the 25 hour DST switch.
crossBorderProduct	true	all	The product is traded on the Consolidated Order Book.
icebergOrderProduct	true	all	Iceberg orders are supported for this product.
icebergMinPeakSize	true	all	The minimum peak size of an iceberg order
icebergPriceDeltaRange	true	all	The maximum value for the peak price delta of an iceberg order.
stopOrderProduct	true	FUT, CRO	The product supports stop orders.
indicativeOrderProduct	true	FUT, CRO	Indicative orders are supported for this product.
linkedOrderProduct	true	all	Orders can be linked together for this product.
quoteOrderProduct	true	FUT, CRO	Quotes are supported for this product.
quoteMinQuantity		FUT, CRO	Defines the minimum quantity that a quote is allowed to have.
otcAllowed	true	all	OTC trade registration is allowed for this product.
otcOnly	true	all	The product is for OTC trading only.
onExchangePrearrangedTrade	true	FUT, CRO	The product supports on-exchange prearrange trades.
privateAndConfidential	true	FUT, CRO	Private and confidential trades are allowed for the product.
volatilityInterruption	true	FUT, CRO	Volatility interrupt is relevant for the product.
volatilityPrice		FUT, CRO	Defines the percentage of the price, above which volatility interrupt is triggered.
commodityLimitEnabled	true	COM	The commodity limit is enabled for the product.
intraProductSpreads	true	FUT, CRO	The product supports intra-product spreads.
intraProductSpreadContractCount		FUT, CRO	Defines how many contracts are used for intra-product spreads.
productIsWithinDelivery	true	COM	Trading of the product is allowed when delivery has started.
leadTime		COM	Defines the lead time in minutes for products within delivery.
autoOrderMatcher	true	all	The product uses the automatcher (regular price-time priority matcher)
liftOrderMatcher	true	FUT, CRO	The product is associated with the Hit & Lift matcher.
continuousAONProduct	true	COM	AON orders are supported in continuous trading for this product.
productCommodityId		COM	
referencePrice		all	The initial reference price of the product.
clgHouses	true/false	all	The product matcher has clearing house restrictions
crossProductMatchingEnabled	true	all	The product has cross product matching feature enabled
tradeDecomposition	true	all	The product has trade decomposition feature enabled
groupName		all	The product group name
aotEnabled	true	COM - Linked products only	An automated order transfer for the linked product is enabled.

XML Tag	Type	m/o	No.	Data Type	Short description
ProdInfoRprt	SE	m	1	Structure	
StandardHeader	*SE*	*m*		*Structure*	*Standard header of each message. Please see 5.1.3.*
ProdList	SE	o	0..1	Structure	
Prod	SE	o	0..n	Structure	
prodType	A	m		String (3)	Indicates the type of the product. Valid values are: "**ENG**": Energy products "**COM**": Commodities "**FUT**": Futures "**CRO**": Cross Product Spreads "**UDD**": User defined delivery
prodName	A	m		String(255)	The unique identifier name of the product.
dsplName	A	m		String(255)	The string used to display the product.
revisionNo	A	m		Long	The revision number of the product. This value is increased by one every time the product is modified by the system.

XML Tag	Type	m/o	No.	Data Type	Short description
base	A	m		Boolean	Defines if the product is a base product
baseReference	A	c		String(255)	If the product is not a base product, this attribute contains the reference to the base product
groupName	A	o		String(255)	A group name of the product as set up in the system
cashLmtEnabled	A	m		Boolean	Indicates whether the cash limit is enabled for the product
riskSetId	A	c		Long	The ID of the default set of risk parameters for a cash limit calculation for this product. The attribute is mandatory when cashLmtEnabled is true; otherwise it is not set. Allowed values: An existing set of parameters
currency	A	m		String(3)	The currency of the product (e.g. “EUR”).
minQty	A	m		Integer	Defines the minimum tradable quantity of the product.
maxQty	A	m		Integer	Maximum allowed quantity for orders entered in contracts belonging to this product.
maxAmount	A	o		Long	Maximum value of an order permitted on this product (price*quantity*delivery unit)
lotSize	A	m		Integer	Defines the minimum increment of the quantity in this product. The value is entered as an integer, but the decimal quantity shift is applied.
decShftQty	A	m		Integer	The decimal shift of the quantity information. A value of 2 results in a display of 100 Kw.
qtyUnit	A	m		String(255)	Defines the quantity unit.
delUnits	A	o		Double	Defines delivery units of a product in relation to the basic period (e.g. If the basic period is 1 month, for 3 month products is set to 3.) The attribute is not set for type User-Defined Delivery Period It is mandatory for other types. For XBID style user defined blocks, this attribute contains the delivery length expressed in number of the original product canonical contracts (e.g. 20-23 has delUnits = 3).
minPx	A	m		Long	The minimum price allowed for orders entered in contracts belonging to this product.
maxPx	A	m		Long	The maximum price allowed for orders entered in contracts belonging to this product.
tickSize	A	m		Integer	Defines the minimum increment for limit prices for this product. The value is entered as an integer, but the decimal price shift is applied (please refer to the example in minimum peak size).
decShftPx	A	m		Integer	The decimal shift of the price information. A value of 2 results in a display in 1/100 of specified currency.
exchangeId	A	m		String(255)	The exchange id of the exchange where the orders for this product are matched.
assetType	A	o		String(3)	This field is used to define the type of the underlying asset Valid values: '**COM**': Commodity '**STO**': Stock '**IDX**': Index
assetName	A	o		String(64)	A text input field used to define the asset name.
assetExchangeId	A	o		String(4)	A text input field used to define the exchange on which the underlying asset is traded
executionRestriction	A	o		String(3)	Defines whether orders referencing the contracts of this product can be partially matched or if only with the full quantity. NON: No restriction (partial matching allowed) AON : All or nothing
contractsGenerationNumber	A	m		Integer	Denotes how many contracts are generated in advance
contActBusinessDay	A	o		Boolean	Defines if the contract activation day should be on a business day or on any calendar day.
contActDay	A	o		String(3)	The week day at which the contract is activated. Valid Values: MON, TUE, WED, THU, FRI, SAT, SUN
contActTime	A	o		DateTime	The time at which the contract is activated.

XML Tag	Type	m/o	No.	Data Type	Short description
contExpiryBusinessDay	A	o		Boolean	Defines if the contract expiry day should be on a business day or on any calendar day.
contExpiryDay	A	o		String(3)	The week day at which the contract expires. Valid Values: MON, TUE, WED, THU, FRI, SAT, SUN
contExpiryTime	A	o		DateTime	The time at which the contract expires.
state	A	m		Char(4)	The current state of the product. The following values are allowed: **"HIBE":** The product is inactive and is not available for trading. **"ACTI": The** product is active and is available for trading. "**IACT**": The product is inactive and is not tradable.
timeZone	A	m		String(32)	The time zone identifier of the time zone that the product is operated in. Note that only the following valid values are available for the TimeZone : •CET (default value) •Europe/London
masterProd	A	o		String(255)	A reference to the corresponding master product
linkedProd	A	o		String(255)	A reference to the corresponding linked product.
ProdCfgs	SE	o	0..n	Structure	Used to list exchange specific attributes of the product. The product attributes are given as key-value pairs.
cfgKey	A	m		String(255)	Exchange specific product attribute name
cfgVal	A	m		String(255)	Exchange specific product attribute value
DlvryAreaAssignment	SE	m		String	The list of delivery areas assigned to the product.
dlvryAreaId	A	m		String	Delivery Area Id
riskSetId	A	o		Long	The Id of the risk set that was used for the cash limit calculation for this product in this delivery area. The attribute is mandatory when cashLmtEnabled is true; otherwise it is not set. The default selection is the product default risk set from the Attributes panel. Allowed values: •An existing set of parameters
ContractName Pattern	CE	o	0..1	String(255)	A format string for the contract name.

Table 37: The message layout of a Product Information Response.

6.4.20 Market State Request (MktStateReq)

MktStateReq	
Type:	Inquiry Request
Roles:	<ALL>
Routing Keys:	`m7.request.inquiry`
Request Limits:	14/70

This message is used to request information about the current market state. In the current version, the message has no content.

XML Tag	Type	m/o	No.	Data Type	Short description
MktStateReq	SE	m	1	Structure	
StandardHeader	*SE*			*Structure*	*Standard header of each message. Please see 5.1.3.*

Table 38: The message layout of a Market State Request.

6.4.21 Market State Report (MktStateRprt)

MktStateRprt	
Type:	Inquiry Response, Broadcast
Response to:	MktStateReq (sent to the private response queue see 3.1)
Broadcasted:	Yes
Broadcast Routing Keys:	`<schema-version>.public`
Broadcast audience:	`All users`
Roles:	<ALL>

This message is broadcast when a change in the current market state occurs.

It is also sent as a response to the Market State Request message, and is sent to the private response queue of the requesting user, see 3.1.

XML Tag	Type	m/o	No.	Data Type	Short description
MktStateRprt	SE	m	1	Structure	
StandardHeader	*SE*	*m*		*Structure*	*Standard header of each message. Please see 5.1.3.*
state	A	m		Char(4)	Contains the current market state. The following values are allowed: **"HIBE":** Hibernated; no trading is possible, and the order books are empty. **"ACTI": The m**arket is active and trading is possible.
revisionNo	A	m		Long	The revision number of the market. With every change of the market state, this value is increased by one.

Table 39: The message layout of a Market State Report.

6.4.22 Hub-to-Hub ATC Matrix Request (HubToHubReq)

HubToHubReq	
Type:	Inquiry Request
Roles:	Trader, Data Vendor, Balance User, Admin, Sales User. Must have the additional right 'Capacity info'.
Routing Keys:	`m7.request.inquiry`
Request Limits:	50/500

This request is used to retrieve the Hub-to-Hub ATC matrix from XBID for a given delivery day. By default, the data can be retrieved for the current day and the (current day + 1). This time interval can be parametrised. To retrieve ATC values, the user must have the "Capacity information" right.

XML Tag	Type	m/o	No.	Data Type	Short description
HubToHubReq	SE	m		Structure	
StandardHeader	*SE*			*Structure*	*Standard header of each message. Please see 5.1.3.*
mktArea	A	o		String(16)	Market Area
area	A	o		String(16)	Area
dlvryDay	A	m		Date	Delivery date. ATC values are valid for this date.

Table 40: The message layout of a Hub-to-Hub ATC Matrix Request.

6.4.23 Hub-to-Hub ATC Matrix Response (HubToHubResp)

HubToHubRprt	
Type:	Inquiry Response
Response to:	HubToHubReq (sent to the private response queue see 3.1)
Broadcasted:	No
Broadcast Routing Keys:	--
Roles:	Trader, Data Vendor, Balance User, Admin, Sales User. Must have the additional right 'Capacity info'.

This message is returned as a response to the Hub-to-Hub ATC Matrix Request. It is sent to the private response queue of the user requesting the Hub-to-Hub ATC values, see 3.1.

To retrieve or receive ATC values, the user must have the “Capacity information” right.

Errors that can occur when requesting Hub-to-Hub information are listed in DFS200. In such case, an ErrResp will be returned.

XML Tag	Type	m/o	No.	Data Type	Short description
HubToHubResp	SE	m	1	Structure	
StandardHeader	*SE*	*m*		*Structure*	*Standard header of each message. Please see 5.1.3.*
HubToHubAtcList	SE	o	1..n	Structure	
revisionNo	A	m	1	Long	The H2H matrix version. The version number is generated internally and increases upon any updates of the matrix.
dlvryStart	A	m	1	DateTime	The delivery start date
dlvryEnd	A	m	1	DateTime	The delivery end date
timestmp	A	m	1..*	DateTime	The timestamp when the ATC data was received from the Capacity system.
HubFrom	SE	o	0..n	Structure	
frm	A	m		String(16)	The outgoing Area.
Atc	SE	o	0..n	Structure	
to	A	m	1	String(16)	The target Area.
in	A	m	1	Long	The ATC value ‘IN’ for target Area.
out	A	m	1	Long	The ATC value ‘OUT’ for target Area.

Table 41: The message layout of a Hub-to-Hub ATC Matrix Report.

6.4.24 Hub-to-Hub Area Info Request (H2HAreaInfoReq)

H2HAreaInfoReq	
Type:	Inquiry Request
Roles:	All users with the additional right ‘Capacity Info’.
Routing Keys:	`m7.request.inquiry`
Request Limits:	50/500

The H2H Area Info Request is used to retrieve detailed information about the remote market/delivery areas. To retrieve or receive Hub-to-Hub Area Info, the user must have the “Capacity Info” right.

XML Tag	Type	m/o	No.	Data Type	Short description
H2HAreaInfoReq	SE	m		Structure	
StandardHeader	*SE*			*Structure*	*Standard header of each message. Please see 5.1.3.*

Table 42 The message layout of a Hub-to-Hub Area Info Request

6.4.25 Hub-to-Hub Area Info Response (H2HAreaInfoResp)

H2HAreaInfoResp	
Type:	Inquiry Response
Response to:	H2HAreaInfoReq (sent to a private response queue)
Broadcasted:	No
Broadcast Routing Keys:	– –
Roles:	The user must have the additional right ‘Capacity Info’

This message is returned as a response to the Hub-to-Hub Area Info Request. It is sent to the private response queue of the user requesting the Hub-to-Hub Area Info, see 3.1.

To retrieve or receive Hub-to-Hub Area Info, the user must have the “Capacity Info” right.

Errors that can occur when requesting Hub-to-Hub Area Info are listed in DFS200. In such case, an ErrResp will be returned.

XML Tag	Type	m/o	No.	Data Type	Short description
H2HAreaInfoResp	SE	m	1	Structure	
StandardHeader	*SE*	*m*		*Structure*	*Standard header of each message. Please see 5.1.3.*
H2HAreaList	SE	o	0..n	Structure	
H2HArea	SE	o	0..n	Structure	Market/Delivery Area information
areaId	A	m	1	String(255)	Market/Delivery Area Id. In the power market, this corresponds to the EIC code of the market/delivery area.
name	A	m	1	String(255)	The name of the market/delivery area, usually used for display purposes.
longName	A	m	1	String(255)	The long name of the market/delivery area.

Table 43: The message layout of a Hub-to-Hub Area Info Response

6.4.26 Hub-to-Hub Notification (HubToHubNtf)

HubToHubNtf	
Type:	Broadcast
Broadcasted:	Yes
Broadcast Routing Keys:	`<schema-version>.hubToHub`
Broadcast audience:	`All users with the additional right 'Capacity info'`
Roles:	<ALL>

This message is broadcasted every 30 seconds to inform about any changes in the Hub-to-Hub ATC matrix data. The time interval between HubToHubNtf messages can be set in the system configuration.

The message contains the part of the matrix that has changed since the previous broadcast. If there was a change for the outgoing delivery area Elia and the target area RTE for a delivery period 12.00 – 13.00, the HubToHubNtf message will retrieve the part of the ATC matrix that corresponds to the outgoing delivery area Elia, delivery period 12.00 – 13.00 and all target areas (not only RTE).

The routing key is shared with HubToHubHeartbeat message. If the client consumes only HubToHubNtf messages (and not HubToHubHeartbeat messages), sequence gaps may appear.

XML Tag	Type	m/o	No.	Data Type	Short description
HubToHubNtf	SE	m		Structure	
HubToHubAtcList	SE	o	1..n	Structure	
revisionNo	A	m	1	Long	The H2H matrix version. The version number is generated internally and increases upon any updates of the matrix. For consistency check purposes it verifies that: • The revisionNo of the last broadcast is higher than the revisionNo of the previous one, • Possible gaps between the broadcasts should be detected using a standard broadcast sequence.
dlvryStart	A	m	1	DateTime	The delivery start date
dlvryEnd	A	m	1	DateTime	The delivery end date
timestmp	A	m	1..*	DateTime	The timestamp when the ATC data was received from the Capacity system.
HubFrom	SE	o	0..n	Structure	
frm	A	m		String(16)	The outgoing Area.
Atc	SE	o	0..n	Structure	
to	A	m	1	String(16)	The target Area.
in	A	m	1	Long	The ATC value 'IN' for target Area
out	A	m	1	Long	The ATC value 'OUT' for target Area

Table 44: The message layout of a Hub-to-Hub Notification

6.4.27 Hub-to-Hub Heartbeat (HubToHubHeartbeat)

HubToHubHeartbeat	
Type:	Broadcast
Broadcasted:	Yes
Broadcast Routing Keys:	`<schema-version>.hubToHub`
Broadcast audience:	`All users with the additional right 'Capacity info'`
Roles:	<ALL>

This message is broadcasted every 5 seconds to provide information that the Hub-to-Hub module, that provides information about the ATC matrix, is connected to XBID and is running. If the system is disconnected from XBID or three heartbeats are missed in a row, it is necessary to collect the whole ATC matrix again using HubToHubReq message.

The routing key is shared with HubToHubNtf message. If the client consumes only HubToHubHeartbeat messages (and not HubToHubNtf messages), sequence gaps may appear.

XML Tag	Type	m/o	No.	Data Type	Short description
HubToHubHeartbeat	SE	m	1	Structure	
StandardHeader	*SE*	*m*		*Structure*	*Standard header of each message. Please see 5.1.3.*
sobConnectionState	A	m	1	String(128)	Information about the state of the connection to XBID. Possible values are CONNECTED/DISCONNECTED. When M7 disconnects from XBID, the last heartbeat sent by the system contains sobConnectionState =DISCONNECTED.

6.4.28 Settlement Process Information Request (StlmntProcessInfoReq)

StlmntProcessInfoReq	
Type:	Inquiry Request
Roles:	Market Operation, Sales, Access for Traders is configurable
Routing Keys:	`m7.request.inquiry`
Request Limits:	14/70

A user can send this request to obtain a history of the Settlement Process Info Reports. The backend system will respond with a StlmntProcessInfoRprt message to the user's private response queue, (see 3.1).

Note that this is only needed when a user logs into the system. After the login, all subsequent settlement process information updates for trades will be automatically broadcast to the user by the backend.

XML Tag	Type	m/o	No.	Data Type	Short description
StlmntProcessInfoReq	SE	m	1	Structure	
StandardHeader	*SE*			*Structure*	*Standard header of each message. Please see 5.1.3.*
startDate	A	m		DateTime	The start of the period for which the settlement process information is retrieved. This value must fulfil the following conditions: endDate – startDate <= 25 hours
endDate	A	m		DateTime	The timestamp of the end of the period for which the settlement process information is retrieved. This value must fulfil the following conditions: endDate – startDate <= 25 hours
unAcknOnly	A	o		Boolean	If set to "**true**", only settlement process information with stlmntState = "**SENT**" will be returned. Otherwise, all settlement process information for the requested period will be returned. Default ="**false"**
lastOnly	A	o		Boolean	If set to "**true**", only the last revision of the settlement process information is returned. Default = "**true**"
clientAcctId	A	o		String	This specifies for which clientAccountId is applicable for the request.

Table 45: The message layout of a Settlement Process Information Request.

6.4.29 Settlement Process Information Report (StlmntProcessInfoRprt)

StlmntProcessInfoRprt	
Type:	Inquiry Response, Broadcast
Response to:	StlmntProcessInfoReq (sent to the private response queue see 3.1)
Broadcasted:	Yes
Broadcast Routing Keys:	<schema-version>.settlement <schema-version>.bg.<acctId>
Broadcast audience:	Admins, Sales configurable (with the routing key <schema-version>.bg.<acctId>): - The owner of half of the trade and all of the users from his balancing group - A broker with assignment rights to the trader (owner of one of the orders).
Roles:	Market Operation, Sales, Access for Traders is configurable

This message reports on the various states of settlement processing as they are communicated by the settlement system. It is broadcast by the backend system to the clients whenever the backend receives updated settlement information.

In addition, this message is sent as a reply message to the private response queue of a trader that sends a Settlement Process Information Request (StlmntProcessInfoReq), see 3.1.

XML Tag	Type	m/o	No.	Data Type	Short description
StlmntProcessInfoRprt	SE	m	1	Structure	
StandardHeader	*SE*			*Structure*	*Standard header of each message. Please see 5.1.3.*
Trade	SE	m	0..n		
tradeId	A	m		Long	The trade ID of the trade.
revisionNo	A	m		Long	The revision number of this trade. With every trade change, the revision number is increased by one.
stlmntState	A	m		Char(4)	The settlement system status for the trade. Valid values are: "**INIT**": The initial status of a trade before being processed. The sole state if the settlement capability is not turned on see chapter 6.1.5. "**SNDG**": The trade information was accepted by the transfer system "**SENT":** The trade information was sent to the settlement system **"ACKN":** The trade information was received by the settlement system "**INFO**": Additional information has been received from the settlement system
stlmntRevisionNo	A	m		Long	The revision number of the settlement supplied by the settlement system.
stlmntInfo	A	o		String(255)	Additional information supplied by the settlement system.
remoteTradeId	A	o		Long	The remote Trade Id as defined by the remote backend system (i.e. XBID SOB)

Table 46: The message layout of a Settlement Process Information Report.

6.4.30 Reference Price Update (RefPxUpd)

RefPxUpd	
Type:	Management Request
Roles:	Market Operation
Routing Keys:	m7.request.management

This message contains the reference price of contracts for a particular date. However, the date quoted in the message cannot be greater than current business date.

Every update of a reference price will generate a Reference price report broadcast containing the updated prices.

XML Tag	Type	m/o	No.	Data Type	Short description
RefPxUpd	SE	m	1	Structure	
StandardHeader	*SE*	*m*		*Structure*	*Standard header of each message. Please see 5.1.3.*
RefPxList	SE	m	1		List of price elements
RefPx	SE	m	1..n	Structure	Reference price element
contractId	A	m		Integer	Id of the contract in the M7 application.
dlvryAreaId	A	m		Char(16)	Delivery area identification
refPxType	A	o		Char(1)	Type of the reference price updated Valid values are "**C**": Closing price (the default value if the refPxType attribute is omitted) "**O**": Opening price
refPx	A	m		Long	The reference price of the contract. The price format is described in 5.1.8.
date	A	m		Date	The date to which the reference price refers.

Table 47: The message layout of a Reference price update.

6.4.31 Reference Price Request (RefPxReq)

RefPxReq	
Type:	Inquiry Request
Roles:	<ALL>
Routing Keys:	`m7.request.inquiry`
Request Limits:	14/70

With this request message it is possible to inquire the reference price for a specific date or for all of the available reference prices of a contract.

- If the contractId and date are populated - then the specific reference price is returned in the report, if available.
- If the contractId is populated, and the date is empty - then all of the available reference prices for the contract are returned in the report.
- If the contractId is empty, and the date is populated - then all of the available reference prices for this date in all of the contracts assigned to the user are returned in the report.
- If the contractId is empty, and the date is empty- then all of the available closing prices for all of the contracts assigned to the user are returned in the report.
- If only refPxType is used, then only the reference prices of this type are returned. If it is omitted, all reference price types are returned.

XML Tag	Type	m/o	No.	Data Type	Short description
RefPxReq	SE	m	1	Structure	
StandardHeader	*SE*	*m*		*Structure*	*Standard header of each message. Please see 5.1.3.*
dlvryAreaId	A	o		Char(16)	The delivery area identification. If the field is omitted, the reference prices are returned for each Delivery area assigned to the user.
contractId	A	o		Integer	The contract id for which the information is required. If the field is left empty then all of the closing prices for all of the products assigned to the user are returned irrespective of if a date is specified or not.
refPxType	A	o		Char(1)	The type of reference price updated Valid values are "**C**": Closing price (the default value) "O": Opening price If the field is left empty, then all types are returned.
date	A	o		DateTime	The date for which the information is required. If the field is left empty, then all of the available closing prices are returned.

Table 48: The message layout of a Reference price request.

6.4.32 Reference Price Report (RefPxRprt)

RefPxRprt	
Type:	Inquiry Response; Broadcast
Response to:	Reference price request (sent to the private response queue see 3.1)
Broadcast:	Yes
Broadcast Routing Keys:	`<schema-version>.prd.<prodName>`
Broadcast audience:	`All users with an assignment to a particular product.`
Roles:	`All`

The Reference Price report is used as a response to the Reference Price Request, and is also used to broadcast information after receiving a Reference price update message.

XML Tag	Type	m/o	No.	Data Type	Short description
RefPxRprt	SE	m	1	Structure	
StandardHeader	*SE*	*m*		*Structure*	*Standard header of each message. Please see 5.1.3.*
RefPxList	SE	o	0..1		List of price elements
RefPx	SE	m	1..n	Structure	Reference price element
dlvryAreaId	A	m		Char(16)	The delivery area identification
contractId	A	m		Long	The Id of the contract in M7 application.
refPxType	A	o		Char(1)	The type of the reference price updated Valid values are "**C**": Closing price (the default value) "O": Opening price
refPx	A	m		Long	The reference price of the contract. The price format is described in 5.1.8.
date	A	m		Date	The date to which the reference price refers.

Table 49: The message layout of a Reference Price Report.

6.4.33 Implied Order Request (ImplOrdrReq)

ImplOrdrReq	
Type:	Inquiry Request
Roles:	Trader, broker, market maker, market operator
Routing Keys:	`m7.request.inquiry`
Request Limits:	1/10

The Implied order request is used to inquire for calculated implied orders. If the requested is allowed by the exchange, an ImplOrdrResp is sent, otherwise an ErrResp is returned.

XML Tag	Type	m/o	No.	Data Type	Short description
ImplOrdrReq	SE	m	1	Structure	
StandardHeader	*SE*	*m*		*Structure*	*Standard header of each message. Please see 5.1.3.*
contractID	CE	o	0..n	Long	The contract for which the implied orders are requested. If they are not present, the response will contain the implied orders for each contract that are available to the requesting user.

Table 50: The message layout of an Implied Order Request Price Report.

6.4.34 Implied Order Report (ImplOrdrRprt)

ImplOrdrRprt	
Type:	Inquiry Response; Broadcast
Response to:	Implied order request (sent to the private response queue see 3.1)
Broadcast:	Yes
Broadcast Routing Keys:	`<schema-version>.prddlvr.<prodName>.<dlvryAreaId>`
Broadcast audience	`All users with an assigned product and delivery area`
Roles:	`Trader, Broker, Market Maker, Market Operator`

The Implied order report is used as a response to the Implied order request and as a broadcast in case there is a change at the top of orderbook that causes a recalculation of such an implied order.

The trader can then match such an implied order by using the "basket" functionality – sending an OrdrEntry message with listExecInst set to IMPL (see 6.2.1) for the opposite side.

For details about implied orders, please consult DFS140.

XML Tag	Type	m/o	No.	Data Type	Short description
ImplOrdrRprt	SE	m	1	Structure	
StandardHeader	*SE*	*m*		*Structure*	*Standard header of each message. Please see 5.1.3.*
Orders	SE	o	0..1		List of implied orders
orders	SE	m	0..n	Structure	Implied order element
impliedOrderId	A	m		String(255)	The Id of an implied order.
contractId	A	m		Long	The Id of the contract
dlvryAreaId	A	m		Char(16)	The delivery area identification
side	A	m		Char(4)	Defines the side of which the implied order should be displayed The valid values are "**BUY**", "**SELL**".
qty	A	m		Integer	Contains the total quantity of the order (see 5.1.7)
price	A	m		Long	The limit price of the implied order. The price format is described in 5.1.8.
basketContent	SE	m	1	Structure	Describes how the basket should be prefilled in order to match the implied order.
contract	SE	m	1..n	Structure	
contractId	A	o		Long	The Id of the contract
price	A	m		Long	The price of the order.

Table 51: The message layout of an Implied Order Report.

6.5 Order Quotes

Order Quotes can be used by market participants to point the market to any currently open positions. This feature is configurable in the backend system and might be disabled. Its availability can be checked with the capabilities list in the System Info Response (see 6.1.5).

6.5.1 Order Quote Request (OrdrQuoteReq)

OrdrQuoteReq	
Type:	Management Request
Roles:	Trader
Routing Keys:	`m7.request.management`

This message is used to submit an order quote request.

XML Tag	Type	m/o	No.	Data Type	Short description
OrdrQuoteReq	SE	m		Structure	
StandardHeader	*SE*			*Structure*	*Standard header of each message. Please see 5.1.3.*
ordrId	A	m		Long	Unique identifier for an Order.

Table 52: The message layout of an Order Quote Request.

6.5.2 Order Quote Report (OrdrQuoteRprt)

OrdrQuoteRprt	
Type:	Broadcast
Response to:	---
Broadcasted:	Yes
Broadcast Routing Keys:	`<schema-version>.<prodName>`
Broadcast audience:	`All users with an assigned product.`
Roles:	<ALL>

A broadcast message about an order quote in the market.

XML Tag	Type	m/o	No.	Data Type	Short description
OrdrQuoteRprt	SE	m		Structure	
StandardHeader	*SE*	*m*		*Structure*	*Standard header of each message. Please see 5.1.3.*
ordrId	A	m		Long	The unique identifier for the order.
contractId	A	m		Integer	The underlying contract ID of the order.
px	A	m		Long	The limit price of the order. The price format is described in 5.1.8.
qty	A	m		Integer	The quantity of the order.
side	A	m		String(4)	Defines the side of which the implied order should be displayed Valid values are:- **"BUY":** Buy order. **"SELL"**: Sell order.
dlvryAreaId	A	m		Char(16)	The delivery area of the order.
timestmp	A	m		DateTime	The timestamp of when the order was entered.
reqTime	A	m		DateTime	The timestamp of when the order quote request was submitted.

Table 53: The message layout of an Order Quote Report.

6.5.3 Order Quote Setup Request (OrdrQuoteSetupReq)

OrdrQuoteReq	
Type:	Management Request
Roles:	Trader
Routing Keys:	`m7.request.management`

This request is used to setup the order quote functionality for one user.

XML Tag	Type	m/o	No.	Data Type	Short description
OrdrQuoteSetupReq	SE	m		Structure	
StandardHeader	*SE*	*m*		*Structure*	*Standard header of each message. Please see 5.1.3.*
usrId	A	m		Integer	The unique identifier for the user.
ordrReqActive	A	m		Boolean	Activate or deactivate the retrieval of order quotes.
sendMail	A	m		Boolean	Send out an email for order quotes.
sendSMS	A	m		Boolean	Send out an SMS for order quotes.
mailAddress	A	m		String(255)	The receiving mail address.
mobileNumber	A	m		String(255)	The receiving mobile number.

Table 54: The message layout of an Order Quote Setup Request.

6.5.4 Delete Quotes Request (DeleteQuotesReq)

DelQuotesReq	
Type:	Management Request
Roles:	Market Maker
Routing Keys:	`m7.request.management`

Used by the market maker for the mass deletion of quotes, these can be sent on behalf of other user.

The user may send the request for one or several products. For each product in the request, there is the possibility to delete all of the quotes by setting the Action parameter on the product level to All. If the value of the parameter is None, then only the quotes with contracts specified in the contract list for this product are deleted. On a contract level it is possible to delete either both sides of a quote by setting the actionOnQuoSide parameter to Both on the contract level. It is also possible to delete either the buy side of the quote or the sell side of the quote, by setting the Action parameter to Buy or Sell respectively.

XML Tag	Type	m/o	No.	Data Type	Short description
DeleteQuotesReq	SE	m	1	Structure	
StandardHeader	*SE*	*m*		*Structure*	*Standard header of each message. Please see 5.1.3.*
acctId	A	m		String(32)	The account identification
ProdList	SE	m	1	Structure	List of products for which the quote deletion is requested.
Prod	SE	m	1…n	Structure	
prodName	A	m		String(255)	The product identifier
action	CE	o		String	Valid values: **ALL** - All of the quotes for the product are deleted. **The action is not present** - Only quotes from the specified contracts below are deleted.
ContractList	SE	c	1	Structure	List of contracts. Mandatory if action ALL is not present
Contract	SE	o	0..n	Structure	The contract element
contractId	A	o		Long	The contract identifier.
action	A	o		String(4)	**BUY** - Delete only the buy quote side. **SELL** - Delete only the sell quote side. **BOTH** - Delete both of the quote sides

Table 55: The message layout of a Delete all quotes request

6.5.5 Delete Quotes Response (DeleteQuotesResp)

DeleteQuotesResp	
Type:	Management Response; Broadcast
Response to:	DeleteAllQuotesReq (sent to the private response queue see 3.1)
Broadcast:	Yes
Broadcast Routing Keys:	`<schema-version>.<prodName>.<dlvryAreaId>.<acctId>`
Broadcast audience:	In case of an on behalf deletion, the market maker and the user who has sent the request receive this message as a broadcast.
Roles:	`Market maker, (Market Operation trading on behalf)`

The Delete Quotes Response is sent to the user who has entered the Delete Quotes Request. In case this was sent as an on behalf deletion, the market maker and the user who has sent the request receive the broadcast message.

If there is one or more errors, the standard message (MsgRprt) is also sent.

XML Tag	Type	m/o	No.	Data Type	Short description
DeleteQuotesResp	SE	m	1	Structure	
StandardHeader	*SE*	*m*		*Structure*	*Standard header of each message. Please see 5.1.3.*
state	A	m		String	Possible values: **SUC**: Completed without errors **REJ**: The request was rejected **ERR**: There were errors during the processing of the request

ErrorList			SE	m	1	Structure	The list of Clearing houses for the quotes
	Error		SE	m	1…n	Structure	The grouping error element for each contract
	contractId		A	m		Long	Contract identifier
		Error	SE	m	0..n	Structure	The single error element
		err	A	m		String(255)	The Order Id as returned by the M7 backend.
		errCode	A	m		Integer	The error code of the error. In case an error message does not have a specific error code, th value 0 will be used.
		clOrdrId	A	o		String(40)	The client order id

Table 56: The message layout of a Delete Quotes Response.

6.5.6 Quote request (QuoteReq)

QuoteReq	
Type:	Inquiry Request
Roles:	Trader, Market operator trading on behalf
Routing Keys:	`m7.request.inquiry`
Request Limits:	1/10

The *Quote Request* functionality is used by a trader to request that market makers enter a quote for a specified contract. The requested entry is only allowed during *Continuous trading.* Traders can enter Quote Requests for all products that allow quotes.

The request contains the contractId, while the input of a quantity and the selection of the buy/sell side are optional. No set values mean that the quote is requested for both sides. Both sides can be sent simultaneously as well.

A public message is broadcast to the market and the RfQ report is sent to the Market Makers.

XML Tag		Type	m/o	No.	Data Type	Short description
QuoteReq		SE	m	1	Structure	
StandardHeader		*SE*	*m*		*Structure*	*Standard header of each message. Please see 5.1.3.*
	QuoteReq	SE	m	1	Structure	
	contractId	A	m		Long	The underlying contract ID of the quote.
	sellPx	A	o		Long	The bid price of the quote. The price format is described in 5.1.8.
	sellQty	A	o		Integer	The quantity of the bid side
	buyPx	A	o		Long	The ask price of the quote. The price format is described in 5.1.8.
	buyQty	A	o		Integer	The quantity of the ask side

Table 57: The message layout of a Request for a quote request

6.5.7 Quote response (QuoteResp)

QuoteResp	
Broadcast:	Yes
Broadcast Routing Keys:	`<schema-version>.<prodName>.<dlvryAreaId>.<acctId>`
Broadcast audience	Market Makers that have the relevant product assigned. Admins
Roles:	`Market maker, (Market Operation trading on behalf)`

A Quote Report is triggered as a result of a Quote Request. It is broadcast to Market Makers that have the relevant product assigned.

The report includes all of the information from the Quote Request.

XML Tag	Type	m/o	No.	Data Type	Short description
QuoteResp	SE	m	1	Structure	
StandardHeader	*SE*	*m*		*Structure*	*Standard header of each message. Please see 5.1.3.*
QuoteReq	SE	m	1	Structure	
contractId	A	m		Long	The underlying contract ID of the order.
sellPx	A	o		Long	The bid price of the quote. The price format is described in 5.1.8.
sellQty	A	o		Integer	The quantity of the bid side
buyPx	A	o		Long	The ask price of the quote. Price format is described in 5.1.8.
buyQty	A	o		Integer	The quantity of the ask side

Table 58: The message layout of a Request for a quote report

6.6 Reference Data

6.6.1 User Report (UserRprt)

UserRprt	
Type:	Management Response, Broadcast
Response to:	LoginReq (sent to the private response queue see 3.1
Broadcasted:	Yes
Broadcast Routing Keys:	`<schema-version>.mbr.<memberId>`
Broadcast audience	`Users from member of the affected user`
Roles:	<ALL>

The User Report will be returned as a response to a Login Request and broadcast after any changes made to the user data and user assignments in the backend system.

XML Tag	Type	m/o	No.	Data Type	Short description
UserRprt	SE	m		*Structure*	
StandardHeader	*SE*	*m*		*Structure*	*Standard header of each message. Please see 5.1.3.*
Usr	*SE*	*m*	1	*Structure*	
usrId	A	m		Integer	The unique identifier of a user.
revisionNo	A	m		Long	The revision number of the User. This always increases upon a change.
sessionId	A	m		Long	The current session id of the user given after the login to the system.
state	A	m		Char(4)	The current state of the User. The following values are allowed: "**ACTI**": The user is active. It is possible to trade using this user. "**DELE**": The user is deleted. Trading using this user is not possible. "**SUSP**": The user is suspended. Trading using this user is not possible.
mbrId	A	m		String(5)	The member id that the user belongs to.
usrCode	A	m		String(6)	The user's user code.
name	A	m		String(255)	The name of the user.
defaultAcctId	A	m		String(32)	The specified default account of the user.
connectionLossMsg	A	o		String(300)	In the event of a connection loss for the previous user session, this field is filled in with a connection loss message which details the connection loss event with the date and time, and the logout action executed by the backend system (see 6.1.1).
mbrName	A	m		String(255)	The name of member that the user belongs to
AssgAcctId	SE	m	0..n	Structure	Contains the accounts assigned to the user
UsrRole	SE	m	1..n	Structure	Contains the user roles assigned to the user

Table 59: The message layout of a User Report.

6.6.2 Member Change Report (MbrChangeRprt)

MbrChangeRprt	
Type:	Broadcast
Response to:	---
Broadcasted:	Yes
Broadcast Routing Keys:	`<schema-version>.public`
Broadcast audience	`All`
Roles:	<ALL>

The Member Change Report is sent for any changes that are made to this Member in the backend system. It is only delivered via a broadcast and cannot be received via an inquiry.

XML Tag	Type	m/o	No.	Data Type	Short description
MbrChangeResp	SE	m		Structure	
StandardHeader	*SE*	*m*		*Structure*	*Standard header of each message. Please see 5.1.3.*
mbrId	A	m		String(5)	The ID of the Member.
revisionNo	A	m		Long	The revision number of this member. This always increases upon a change.
state	A	m		Char(4)	The current state of the member. The following values are allowed: "**ACTI**": The member is active. It is possible to trade using this member. "**IACT**": The member is inactive. It is not possible to trade using this member. "**DELE**": The member is deleted. Trading using this member is not possible. "**SUSP**": The member is suspended. Trading using this member is not possible.
name	A	m		String(255)	The member's name

Table 60: The message layout of a Member Change Report.

6.6.3 Delivery Area Information Request (DlvryAreaInfoReq)

DlvryAreaInfoReq	
Type:	Inquiry Request
Roles:	<ALL>
Routing Keys:	`m7.request.inquiry`
Request Limits:	14/70

The Delivery Area Information Request is used to retrieve detailed information about the delivery areas assigned to a particular account. The request may optionally specify one or more products assigned to that account.

XML Tag	Type	m/o	No.	Data Type	Short description
DlvryAreaInfoReq	SE	m		Structure	
StandardHeader	*SE*			*Structure*	*Standard header of each message. Please see 5.1.3.*
acctId	A	m		String(32)	Account ID.
prodName	CE	o	0.. 1000	String(255)	List of products.

Table 61: The message layout of a Delivery Area Information Request.

6.6.4 Delivery Area Information Report (DlvryAreaInfoRprt)

DlvryAreaInfoRprt	
Type:	Inquiry Response, Broadcast
Response to:	DlvryAreaInfoReq (sent to the private response queue see 3.1)
Broadcasted:	Yes
Broadcast Routing Keys:	`<schema-version>.public` `<schema-version>.dlvr.<dlvryAreaId>`
Broadcast audience	`All`
Roles:	<ALL>

This message is broadcast whenever a change occurs in an attribute of a delivery area.

In addition, it is a response to a Delivery Area Information Request.

XML Tag	Type	m/o	No.	Data Type	Short description
DlvryAreaInfoRprt	SE	m	1	Structure	
StandardHeader	*SE*	*m*		*Structure*	*Standard header of each message. Please see 5.1.3.*
DlvryAreaList	SE	o	0..1	Structure	
DlvryArea	SE	o	0..n	Structure	
dlvryAreaId	A	m		String	The delivery Area Id.
revisionNo	A	m		Long	The revision number. With every change to the delivery area this value is increased by one.
name	A	m		String(255)	The name of the delivery area, usually used for display purposes. Example: RWE
longName	A	m		String(255)	The long name of the delivery area.
state	A	m		Char(4)	The current **Local Status** of the delivery area. The following values are allowed: "**IACT**": The delivery area is inactive and is thus not tradable. "**ACTI**": The delivery area is active. It is possible to trade in that area. "**HIBE**": The delivery area is deactivated (hibernated). Trading in that delivery area is not possible. **"BALA"**: The delivery area is active and in the balancing phase. "**DELE**": The delivery area was deleted. Trading is not possible.
remoteState	A	o		Char(4)	The current **Remote Status** of the delivery area. The following values are allowed: "**ACTI**": The delivery area is active and tradable. "**SUSP**": The delivery area is inactive. Trading is not possible. "**DELE**": The delivery area was deleted. Trading is not possible.
mktAreaId	A	m		String(16)	The ID of the Market Area that this delivery area belongs to.
prodName	CE	o	0..n	String(255)	The list of assigned products. In case there is a change of state for a delivery area, this list is not provided.

Table 62: The message layout of a Delivery Area Information Report.

6.6.5 Market Area Information Request (MktAreaInfoReq)

MktAreaInfoReq	
Type:	Inquiry Request
Roles:	<ALL>
Routing Keys:	`m7.request.inquiry`
Request Limits:	14/70

The Market Area Information Request is used to retrieve detailed information about the market areas assigned to a particular account. The request may optionally specify one or more products assigned to that account.

XML Tag	Type	m/o	No.	Data Type	Short description
MktAreaInfoReq	SE	m		Structure	
StandardHeader	*SE*			*Structure*	*Standard header of each message. Please see 5.1.3.*
acctId	A	o		String(32)	The ID of the account. If it is not specified, all of the Market areas of all delivery areas assigned to the requesting user are returned.
prodName	CE	o	0.. 1000	String(255)	The list of products.

Table 63: The message layout of a Market Area Information Request.

6.6.6 Market Area Information Report (MktAreaInfoRprt)

MktAreaInfoRprt	
Type:	Inquiry Response, Broadcast
Response to:	MktAreaInfoReq (sent to the private response queue see 3.1)
Broadcasted:	Yes
Broadcast Routing Keys:	`<schema-version>.public` `<schema-version>.mkt.<marketAreaId>`
Broadcast audience	`All`
Roles:	<ALL>

This message is broadcast whenever a change occurs in an attribute of a market area. In addition it is a response to a Market Area Information Request.

XML Tag	Type	m/o	No.	Data Type	Short description
MktAreaInfoRprt	SE	m	1	Structure	
StandardHeader	*SE*	*m*		*Structure*	*Standard header of each message. Please see 5.1.3.*
MktAreaList	SE	o	0..1	Structure	
MktArea	SE	o	0..n	Structure	
mktAreaId	A	m		String(16)	The Market Area Id.
name	A	m		String(255)	The name of the market area that is usually used for display purposes. Example: DE
longName	A	m		String(255)	Usually the long name of the market area.
state	A	m		Char(4)	The current state of the market area. The following values are allowed: "**IACT**": The market area is inactive and thus not tradable. "**ACTI**": The market area is active. It is possible to trade in that area. "**HIBE**": The market area is deactivated (hibernated). Trading in that market area is not possible. **"DELE**": The delivery area was deleted. Trading is not possible.
revisionNo	A	m		Long	The revision number. With every market area change this value is increased by one.

Table 64: The message layout of a Market Area Information Report.

6.6.7 Account Information Request (AcctInfoReq)

AcctInfoReq	
Type:	Inquiry Request
Roles:	<ALL>
Routing Keys:	`m7.request.inquiry`
Request Limits:	14/70

The Account Information Request is used to retrieve the list of accounts in the system. This list can be used to identify possible counterparties for a pre-arranged order entry.

XML Tag	Type	m/o	No.	Data Type	Short description
AcctInfoReq	SE	m		Structure	
StandardHeader	*SE*			*Structure*	*Standard header of each message. Please see 5.1.3.*
acctId	CE	m	0.. 1000	String(255)	The list of account IDs requested. If no acctId is provided, all of the accounts that the requesting user has the rights to see are returned.

Table 65: The message layout of an Account Information Request.

6.6.8 Account Information Report (AcctInfoRprt)

AcctInfoRprt	
Type:	Inquiry Response
Response to:	AcctInfoReq (sent to the private response queue see 3.1)
Broadcasted:	Yes
Broadcast Routing Keys:	<schema-version>.<acctId> <schema-version>.public
Broadcast audience	All
Roles:	<ALL>

This message is returned as a response to an Account Information Request. It is also sent when any changes are made to an Account in the backend system.

XML Tag	Type	m/o	No.	Data Type	Short description
AcctInfoRprt	SE	m		Structure	
StandardHeader	*SE*	*m*		*Structure*	*Standard header of each message. Please see 5.1.3.*
AcctList	SE	o	0..1	Structure	List of accounts
Acct	SE	o	0..n	Structure	Account element
acctId	A	m		String(32)	The account ID.
name	A	m		String(64)	The display name of the account.
mbrId	A	m		String(5)	The member identification of the user's member.
preArrangedAvlbl	A	m		Boolean	A flag that determines if this account can be used for entering pre-arranged trades.
state	A	m		Char(4)	The current state of the account. The following values are allowed: "**ACTI**": The account is active. It is possible to trade using this account. "**IACT**": The account is inactive. It is not possible to trade using this account. "**DELE**": The account is deleted. Trading using this account is not possible. "**SUSP**": The account is suspended. Trading using this account is not possible.
revisionNo	A	m		Long	The revision number of this account. This always increases when a change is made to the delivery area.
dlvryAreaId	CE	o	0..n	String(16)	The assigned delivery areas
prodName	CE	o	0..n	String(255)	The assigned products
assignedAcctIds	CE	o	0..n	String(32)	The list of assigned accounts for the broker
clgAcctId	CE	o	0..n	Integer	The assigned clearing accounts

Table 66: The message layout of an Account Information Response.

6.6.9 Account Change Report (AcctChangeRprt)

AcctChangeRprt	
Type:	Broadcast
Response to:	
Broadcasted:	Yes
Broadcast Routing Keys:	<schema-version>.public
Roles:	<ALL>

The Account Change Report is sent as a result of any changes that are made to an Account in the backend system. It is only delivered via a broadcast and cannot be received by sending an inquiry.. It does not contain any clearing accounts assignments or assigned Account Ids.

XML Tag	Type	m/o	No.	Data Type	Short description
AcctChangeRprt	SE	m		Structure	
StandardHeader	*SE*			*Structure*	*Standard header of each message. Please see 5.1.3.*
AcctList	SE	m	1	Structure	The list of accounts
Acct	SE	m	1..n	Structure	The account element
acctId	A	m		Char(32)	The unique identifier of the account.
revisionNo	A	m		Long	The revision number of this account. This is always increased when a change is made.
state	A	m		Char(4)	The current state of the account. The following values are allowed: "**ACTI**": The account is active. It is possible to trade using this account. "**IACT**": The account is inactive. It is not possible to trade using this account. "**DELE**": The account is deleted. Trading using this account is not possible. "**SUSP**": The account is suspended. Trading using this account is not possible.
name	A	m		String(64)	The name of the account.
mbrId	A	m		String(5)	The unique identifier of the associated Member.
preArrangedAvlbl	A	m		Boolean	A flag that determines if this account can be used for entering pre-arranged orders.
prodName	CE	1..n		String	The list of assigned products.
dlvrAreaId	CE	o	0..n	String(16)	Assigned delivery areas
prodName	CE	o	0..n	String(255)	Assigned products

Table 67: The message layout of an Account Change Report.

6.6.10 All Users Request (AllUsersReq)

AllUsersReq	
Type:	Inquiry Request
Roles:	Trader, Market Operation, Broker, Market Maker, Sales
Routing Keys:	`m7.request.inquiry`
Request Limits:	14/70

This request retrieves a list of all of the users of the system. Traders are only allowed to retrieve the users of their own member.

XML Tag	Type	m/o	No.	Data Type	Short description
AllUsersReq	SE	m		Structure	
StandardHeader	*SE*	*m*		*Structure*	*Standard header of each message. Please see 5.1.3.*
mbrId	CE	c	0.. 1000	String(5)	The unique identifier for a member. This is mandatory for a trader. Market Operation users can use this attribute to filter all users of one member.

Table 68: The message layout of an All Users Request.

6.6.11 All Users Response (AllUsersResp)

AcctIdInfoResp	
Type:	Inquiry Response
Response to:	AllUsersReq (sent to the private response queue see 3.1
Broadcasted:	No
Broadcast Routing Keys:	
Roles:	Trader, Market Operation, Broker, Market Maker, Sales

This message is returned as a response to an All Users Request.

Note: loginId, orderRegActive, sendMail, mailAddress, sendSMS and mobileNumber attributes are sent only to users with the Market Operation role.

XML Tag	Type	m/o	No.	Data Type	Short description
AllUsersResp	SE	m	1	Structure	
StandardHeader	*SE*	*m*		*Structure*	*Standard header of each message. Please see 5.1.3.*
UsrList	SE	m	1	Structure	
Usr	SE	o	0..n	Structure	
usrId	A	m		Integer	The unique identifier of a user.
revisionNo	A	m		Long	The revision number of this user. This is always increases upon a change.
usrCode	A	m		String(6)	The trader's user code.
name	A	m		String(255)	The name of the user.
state	A	m		Char(4)	The current state of the user. The following values are allowed: "**ACTI**": The user is active. It is possible to trade using this user. "**DELE**": The user is deleted. Trading using this user is not possible. "**SUSP**": The user is suspended. Trading using this user is not possible.
mbrId	A	m		String(5)	The member id that the trader belongs to.
defaultAcctId	A	m		String(32)	The specified default account of the trader.
loginId	A	o		String	The login ID of the user.
orderReqActive	A	o		Boolean	States whether the order quote feature is enabled/disabled for this user.
sendMail	A	o		Boolean	Send a mail for order quote
mailAddress	A	o		String	The receiving mail address for order quotes.
sendSMS	A	o		Boolean	Send an SMS for order quotes.
mobileNumber	A	o		String	The receiving mobile number
AssgAcctId	SE	o	0..n	String(32)	Contains the accounts assigned to the user
UsrRole	SE	m	1..n	String(255)	Contains the user roles assigned to the user

Table 69: The message layout of an All Users Response.

6.6.12 Clearing Information request (ClgInfoReq)

ClgInfoReq	
Type:	Inquiry Request
Roles:	<ALL>
Routing Keys:	`m7.request.inquiry`
Request Limits:	1/10

This message is used to request the Clearing Information details. The response to this request is the Clearing information report.

XML Tag	Type	m/o	No.	Data Type	Short description
ClgInfoReq	SE	m		Structure	
StandardHeader	*SE*			*Structure*	*Standard header of each message. Please see 5.1.3.*

Table 70: The message layout of a Clearing information request.

6.6.13 Clearing Information report (ClgInfoRprt)

ClgInfoRprt	
Type:	Inquiry Response, Broadcast
Response to:	ClgInfoReq (sent to the private response queue see 3.1
Broadcasted:	Yes
Broadcast Routing Keys:	`<schema-version>.public`
Broadcast audience	`All`
Roles:	<ALL>

This response is used to return detailed Clearing information. It is returned as a response to the Clearing Info Request, and is also broadcast in the event that there is a change in the clearing information.

XML Tag	Type	m/o	No.	Data Type	Short description
ClgInfoRprt	SE	m		Structure	
StandardHeader	*SE*			*Structure*	*Standard header of each message. Please see 5.1.3.*
clgHseList	SE	o	1	Structure	
clgHse	SE	o	0..n	Structure	
clgHseCode	A	m		String(255)	The unique Code of the Clearing House.
clgHseName	A	m		String(255)	The string used to display the name of the Clearing House.
clgHseState	A	m		String(4)	The state of the Clearing house. Valid values are: **ACTI** – The clearing house is active **IACT** – The clearing house is inactive
ClgAcct	SE	o	0..n	Structure	Clearing Account element
clgAcctId	A	m		Integer	The Id of the Clearing Account
revisionNo	A	m		Long	The revision number of this Account. This is always increased upon a change.
name	A	m		String(64)	The name of the clearing account
state	A	m		String(4)	The state of the Clearing account. Valid values are: **ACTI** – The clearing account is active **IACT** – The clearing account is inactive
prodName	CE	m	0..n	String(255)	The list of products that are assigned to the CA

Table 71: The message layout of a Clearing information response.

6.6.14 Bespoke contract creation request (BespokeContractReq)

BespokeContractReq	
Type:	Inquiry Request
Roles:	<ALL>
Routing Keys:	`m7.request.management`
Request Limits:	1/10

This message is used to create a new bespoke contract on the backend. A ContractInfoRprt is returned by the M7 backend as a response if the creation of the bespoke contract was successful, otherwise an error (message ErrResp) is returned.

XML Tag	Type	m/o	No.	Data Type	Short description
BespokeContractReq	SE	m		Structure	
StandardHeader	*SE*			*Structure*	*Standard header of each message. Please see 5.1.3.*
name	A	m	1	String(128)	The contract name. This is used for display purposes.
strikePx	A	c	1	Long	In the event that the bespoke contract is an option, this field contains the strike price of the contract. The price format is described in 5.1.8.
CallPut	A	c	1	String(1)	In the event that the bespoke contract is an option, this field contains the contract type. The following values are available: • '**C**' : Call

					• **'P'** : Put
UndrlngContracts	SE	m	1	Structure	List of underlying contracts
contractId	CE	m	1..n	Long	The underlying Contract identification

Table 72: The message layout of a Bespoke contract creation request.

6.6.15 Risk set information request (RiskSetInfoReq)

RiskSetInfoReq	
Type:	Inquiry Request
Roles:	<ALL>
Routing Keys:	`m7.request.inquiry`
Request Limits:	14/70

This message serves to retrieve all available or specific set(s) of risk parameters.

XML Tag	Type	m/o	No.	Data Type	Short description
RiskSetInfoReq	SE	m		Structure	
StandardHeader	*SE*			*Structure*	*Standard header of each message. Please see 5.1.3.*
riskSetId	CE	o	0.100 0	Long	The ID of the risk set to be returned. When provided, this risk set is returned; otherwise all risk sets are returned.

Table 73: The message layout of a Risk set information request.

6.6.16 Risk Set information report (RiskSetInfoRprt)

RiskSetInfoRprt	
Type:	Inquiry Response, Broadcast
Response to:	RiskSetInfoReq (sent to the private response queue see 3.1
Broadcasted:	Yes
Broadcast Routing Keys:	`<schema-version>.public`
Broadcast audience	`All`
Roles:	<ALL>

The Risk Set information report is returned in response to a RiskSetInfoReq to a private response queue, and is broadcast publicly whenever a set of risk parameters are added, modified and/or deleted. The broadcast contains only risks sets for which something was modified.

XML Tag		Type	m/o	No.	Data Type	Short description
RiskSetInfoRprt		SE	m		Structure	
StandardHeader		*SE*			*Structure*	*Standard header of each message. Please see 5.1.3.*
RiskSetList		SE	o		Structure	
RiskSet		SE	o	1..n	Structure	
riskSetId		A	m		Long	The ID of the risk set
revisionNo		A	m		Long	Provides the current revision number for the current Risk Set entity when contained in a response. In a broadcast report which is sent as a result of a change, this is the revision number of the affected entity.
riskSetName		A	m		String(64)	The name of risk set
status		A	m		String	The type of action which was taken on the risk set. Allowed values: •"ACTI" - The risk set is active •"ADEL" - The risk set is deleted
CashLmtAParameters		SE	M		Structure	Contains a (price-dependent) cash limit parameters
posBuyOrdr		A	m		Double(2 decimals)	The value of a risk parameter for a buy order submitted with a positive price

	posSellOrdr		A	m		Double(2 decimals)	The value of a risk parameter for a sell order submitted with a positive price
	posBuyTrade		A	m		Double(2 decimals)	The value of a risk parameter for a buy side of the trade executed with a positive price
	posSellTrade		A	m		Double(2 decimals)	The value of a risk parameter for a sell side of the trade executed with a positive price
	negBuyOrdr		A	m		Double(2 decimals)	The value of a risk parameter for a buy order submitted with a negative price
	negSellOrdr		A	m		Double(2 decimals)	The value of a risk parameter for a sell order submitted with a negative price
	negBuyTrade		A	m		Double(2 decimals)	The value of a risk parameter for a buy side of the trade executed with a negative price
	negSellTrade		A	m		Double(2 decimals)	The value of a risk parameter for a sell side of the trade executed with a negative price
	CashLmtAlphaParameters		SE	M		Structure	Contains α (price-dependent) cash limit parameters
	buyOrdr		A	m		Double(2 decimals)	The value of a risk parameter for a buy order
	sellOrdr		A	m		Double(2 decimals)	The value of a risk parameter for a sell order
	buyTrade		A	m		Double(2 decimals)	The value of a risk parameter for a buy side of the trade
	sellTrade		A	m		Double(2 decimals)	The value of a risk parameter for a sell side of the trade

Table 74: The message layout of a Risk set information response.

7 Admin Requests and Responses

The requests and responses described in this chapter are used for users with administrative privileges such as Market Operation users to communicate with the backend.

7.1 Reference Data

7.1.1 All Members Request (AllMbrsReq)

AllMbrsReq	
Type:	Inquiry Request
Roles:	Market Operation, Broker, Sales
Routing Keys:	`m7.request.inquiry`
Request Limits:	14/70

A request to retrieve a list of all of the members in the system.

XML Tag	Type	m/o	No.	Data Type	Short description
AllMbrsReq	SE	m		Structure	
StandardHeader	*SE*			*Structure*	*Standard header of each message. Please see 5.1.3.*

Table 75: The message layout of an All Members Request.

7.1.2 All Members Response (AllMbrsResp)

AllMbrsResp	
Type:	Inquiry Response
Response to:	AllMbrsReq (sent to the private response queue)
Broadcasted:	No
Broadcast Routing Keys:	---
Roles:	Market Operation, Broker, Sales

This message is returned as a response to an All Members Request.

XML Tag		Type	m/o	No.	Data Type	Short description
AllMbrsResp		SE	m	1	Structure	
StandardHeader		*SE*			*Structure*	*Standard header of each message. Please see 5.1.3.*
MbrList		SE	o	0..1	Structure	
	Mbr	SE	o	0..n	Structure	
	mbrId	A	m		Char(5)	The unique identifier of the member.
	revisionNo	A	m		Long	The revision number of the member. This value always increases upon a change.
	name	A	m		String(64)	The name of the member.
	state	A	m		Char(4)	The current state of the member. The following values are allowed: "**ACTI**": The member is active. It is possible to trade using this member. "**DELE**": The member is deleted. Trading using this member is not possible. "**SUSP**": The member is suspended. Trading using this member is not possible.
	type	A	m		String(3)	Type of member **"REG"** – Regular member **"BRK"** – Broker member

	Address		SE	o		Structure	Address of the Member
		department	A	o		Char(64)	The department of the member
		street	A	o		Char(64)	The street of the member
		city	A	o		Char(64)	The city of the member
		post Code	A	o		Char(16)	The post code of the member
		country	A	o		Char(64)	The country of the member
	TrdContact		SE	o		Structure	Trading contact of the Member
		name1	A	o		Char(64)	The name of the first trading contact person
		phone1	A	o		Char(64)	The phone of the first trading contact person
		name2	A	o		Char(64)	The name of the second trading contact person
		phone2	A	o		Char(64)	The phone number of the second trading contact person
		fax	A	o		Char(64)	The fax number for trading contacts
	ClgContact		SE	o		Structure	Clearing contact of the Member
		name1	A	o		Char(64)	The name of the first clearing contact person
		phone1	A	o		Char(64)	The phone number of the first clearing contact person
		name2	A	o		Char(64)	The name of the second clearing contact person
		phone2	A	o		Char(64)	The phone number of the second clearing contact person
		fax	A	o		Char(64)	The fax number of the clearing contacts

Table 76: The message layout of an All Members Response.

8 Message Overview & Access Rights

The following matrix provides an overview of all messages, and lists the access rights for each different user role.

Message	Trader	Broker	Market Maker	Data Vendor
General Requests and Responses				
Login Request (LoginReq)	X	X	X	X
Logout Request (LogoutReq)	X	X	X	X
Logout Report (LogoutRprt)	X	X	X	X
System Info Request (SystemInfoReq)	X	X	X	X
System Info Response (SystemInfoResp)	X	X	X	X
Acknowledgement Response (AckResp)	X	X		
Error Response (ErrResp)	X	X	X	X
Change password Request (ChgPwdReq)	X	X	X	X
Order Entry and Maintenance				
Order Entry (OrdrEntry)	X	X	X	
Order Modify (OrdrModify)	X	X	X	
Order Request (OrdrReq)	X	X	X	
Order Execution Report (OrdrExeRprt)	X	X	X	
Pre-Arranged Order Processing (PreArrangedOrdrProcess)	X	X	X	
Modify All Orders (ModifyAllOrdrs)	X	X	X	
Order Limit Request (OrdrLmtReq)	X	X	X	
Order Limit Report (OrdrLmtRprt)	X	X	X	
Trade Maintenance				
Trade Recall Request (TradeRecallReq)	X	X	X	
Prearranged Trade Entry (PrearrangedTradeEntry)		X		
Market Information				
Public Order Books Request (PblcOrdrBooksReq)	X	X	X	X
Public Order Books Response (PblcOrdrBooksResp)	X	X	X	X

Public Order Books Delta Report (PblcOrdrBooksDeltaRprt)	X	X	X	X
Cash Limit Request (CashLmtReq)	X	X	X	X
Cash Limit Report (CashLmtRprt)	X	X	X	X
Cash Limit Delta Report (CashLmtDeltaRprt)	X	X	X	X
Commodity Limit Request (CommodityLmtReq)	X	X	X	
Commodity Limit Report (CommodityLmtRprt)	X	X	X	
Message Request (MsgReq)	X	X	X	X
Message Report (MsgRprt)	X	X	X	X
Trade Capture Request (TradeCaptureReq)	X	X	X	
Trade Capture Report (TradeCaptureRprt)	X	X	X	
Public Trade Confirmation Request (PblcTradeConfReq)	X	X		X
Public Trade Confirmation Report (PblcTradeConfRprt)	X	X		X
Contract Information Request (ContractInfoReq)	X	X	X	X
Contract Information Report (ContractInfoRprt)	X	X	X	X
Product Information Request (ProdInfoReq)	X	X	X	X
Product Information Report (ProdInfoRprt)	X	X	X	X
Market State Request (MktStateReq)	X	X	X	X
Market State Report (MktStateRprt)	X	X	X	X
Hub-to-Hub ATC Matrix Request (HubToHubReq)[8]	X		X	X
Hub-to-Hub ATC Matrix Response (HubToHubResp)[9]	X		X	X
Hub-to-Hub Notification (HubToHubNtf)[10]	X		X	X

[8] The user must have the additional right ‘Capacity info’.

[9] The user must have the additional right ‘Capacity info’.

[10] The user must have the additional right ‘Capacity info’.

Hub-to-Hub Heartbeat (HubToHubHeartbeat)[11]	X		X	X
Hub-to-Hub Area Info Request (H2HAreaInfoReq)	X		X	X
Hub-to-Hub Area Info Response (H2HAreaInfoResp)	X		X	X
Settlement Process Information Request (StlmntProcessInfoReq)	(X)[12]	(X)[2]	X	
Settlement Process Information Report (StlmntProcessInfoRprt)	(X)[12]	(X)[12]	X	
Reference Price Update (RefPxUpd)				X
Reference Price Request (RefPxReq)				X
Reference Price Report (RefPxRprt)	X	X	X	X
Implied Order Request (ImplOrdrReq)	X	X	X	
Implied Order Report (ImplOrdrRprt	X	X	X	
Order Quotes				
Order Quote Request (OrdrQuoteReq)	X	X	X	
Order Quote Report (OrdrQuoteRprt)	X	X	X	
Order Quote Setup Request (OrdrQuoteSetupReq)	X	X	X	
Delete Quotes Request (DeleteQuotesReq)			X	
Delete Quotes Report (DeleteQuotesRprt)			X	
Quote request (QuoteReq)	X	X	X	
Quote response (QuoteResp)			X	
Reference Data				
User Report (UserRprt)	X	X	X	X
Account Change Report (AcctChangeRprt)	X	X	X	X
Member Change Report (MbrChangeRprt)	X	X	X	X
Delivery Area Information Request (DlvryAreaInfoReq)	X	X	X	X

[11] The user must have the additional right ‘Capacity info’.

[12] Access is configurable on the backend side for the ‘Trader’ role.

Delivery Area Information Report (DlvryAreaInfoRprt)	X	X	X	X
Market Area Information Request (MktAreaInfoReq)	X	X	X	X
Market Area Information Report (MktAreaInfoRprt)	X	X	X	X
Account Information Request (AcctInfoReq)	X	X	X	X
Account Information Report (AcctInfoRprt)	X	X	X	X
All Users Request (AllUsersReq)	X	X	X	
All Users Response (AllUsersResp)	X	X	X	
Clearing Information request (ClgInfoReq)	X	X	X	X
Clearing Information report (ClgInfoRprt)	X	X	X	X
Bespoke contract creation request (BespokeContractReq)		X		
Risk Set Information Request (RiskSetInfoReq)	X	X	X	X
Risk Set Information Report (RiskSetInfoRprt)	X	X	X	X
All Members Request (AllMbrsReq)		X		
All Members Response (AllMbrsResp)		X		

9 Forward compatibility

9.1 Forward compatibility - <any> element and <anyAttribute>

The M7 6.0.10 release and API bring new elements for ensuring forward compatibility.

Messages now can contain the “<any>” element and the “<anyAttribute>” attribute. This allows for new elements and attributes to be added in the 6.X API without changing the respective XSDs and connectors, resulting in higher **minor** version messages being compatible with a lower **minor** API version.

The “processContents” attribute of the <any> and <anyAttribute> element is either set to “lax” or “skip” as there will be no new namespace/additional XSDs added to the 6.1 API. However in further minor releases they may be added.

NOTE: Please do not use any additional elements when sending **requests**, unless specifically documented otherwise.

For Java (JAXB) codes the following approach is tested and supported:

Accommodating the use of <any> and <anyAttribute> in JAXB is documented at https://jaxb.java.net Define the Base class; this class would be implemented by all the classes that need to be Extensible:

```
import java.util.List;
import java.util.Map;
import javax.xml.bind.annotation.XmlAnyAttribute;
import javax.xml.bind.annotation.XmlAnyElement;
import javax.xml.bind.annotation.XmlType;
import javax.xml.namespace.QName;
import org.w3c.dom.Element;


public class BaseSchemaExtensible
{
    @XmlAnyElement(lax=true)
    private List<Element> otherElements; // this will have <any> tag data

    @XmlAnyAttribute
    private Map<QName,Object> otherAttributes; // this will have
<anyattribute> tag data
}



@XmlRootElement
class Person extends BaseSchemaExtensible {

  public String getName();
  public void setName(String);

}
```

9.2 Forward compatibility – adding messages to XSD

From M7 Trading XSD schema version 6.5 on, there is no need to raise the major version of XSD when adding new messages. If new XML messages are added, only the **minor** version of XSD will be raised. Clients that do not want to use new messages can simply ignore them, but it is necessary to update the XSD schema.