

DFS180 - M7 - Public Message Interface

M7 Release 6.11.280

Date 31.08.2021
Author M7 Project Team
Reviewer M7 Project Manager

Deutsche Börse AG

Mailing address

Mergenthaleralle 61
65760 Eschborn

Web

www.deutsche-boerse.de

Chairman of the Supervisory Board

Martin Jetter

Executive Board

Theodor Weimer (Chief Executive Officer)
Christoph Böhm (Chief Information Officer / Chief Operating Officer)
Thomas Book (Trading & Clearing)
Heike Eckert (HR (Director of Labour Relations) & Compliance)
Stephan Leithner (Responsible for Pre- & Post-Trading)
Gregor Pottmeyer (Chief Financial Officer)

German stock corporation registered in

Frankfurt/Main

HRB No. 32232

Local court: Frankfurt/Main

Terminology

Term	Description
Client	Program or application acting as a client of the AMQP server.
Trader	Person that logs into the client to participate in the market.

1 Introduction

This document describes the Public Message Interface that the backend system provides to its third party clients.

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

There are two 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 protocol¹ on the application layer of the OSI model.

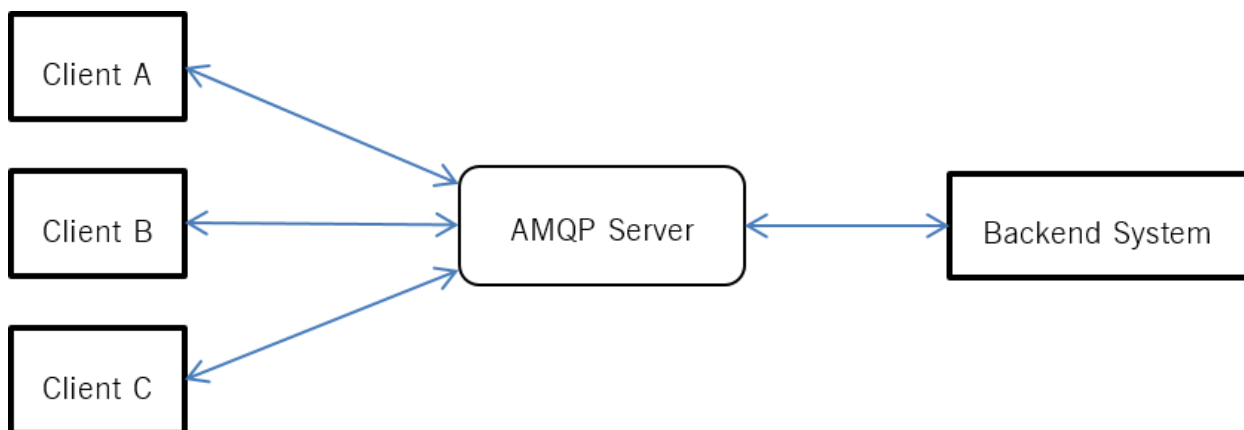
Payload of the messages sent over AMQP is formatted in XML. See www.w3.org/XML for information about XML.

For the Administrative Message Interface that the backend system provides to client applications that supports users with administrative privileges such as Market Operation, please refer to the *DFS180a*.

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



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.

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 that has to be used for further communications with the backend system
 - The message format of the XML Schema files. You will need the 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.



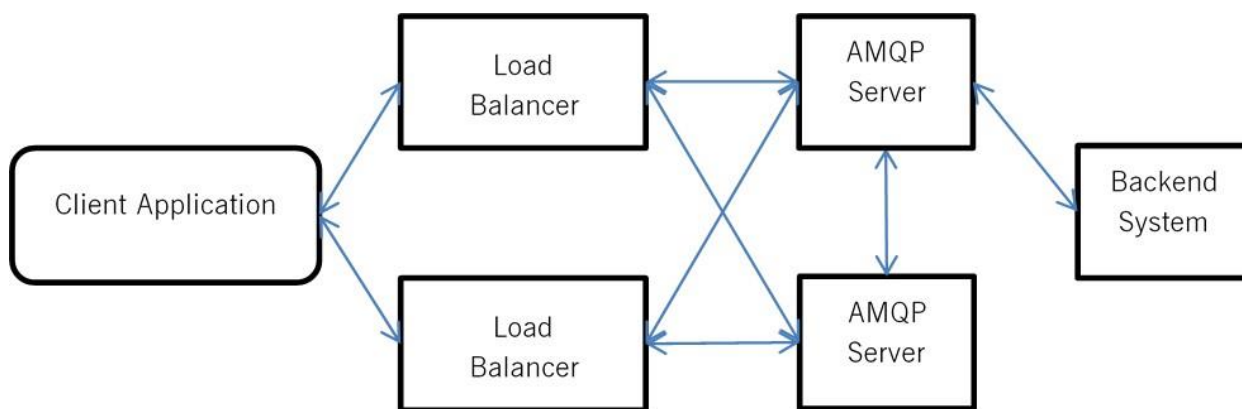
In the RabbitMQ Java client, AMQP-level heartbeats are enabled using method `ConnectionFactory.setRequestedHeartbeat(int)`, specifying the heartbeat interval length in seconds.

AMQP-level heartbeats

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 centers with different IP addresses and URLs. In the standard operating phase, both data centers are active and can be used to make a connection.



Basic M7 Architecture

All connection details (the port, username, password, client certificate) are the same for each data center 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 minimize the operational impact in case one data center 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 authorized 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

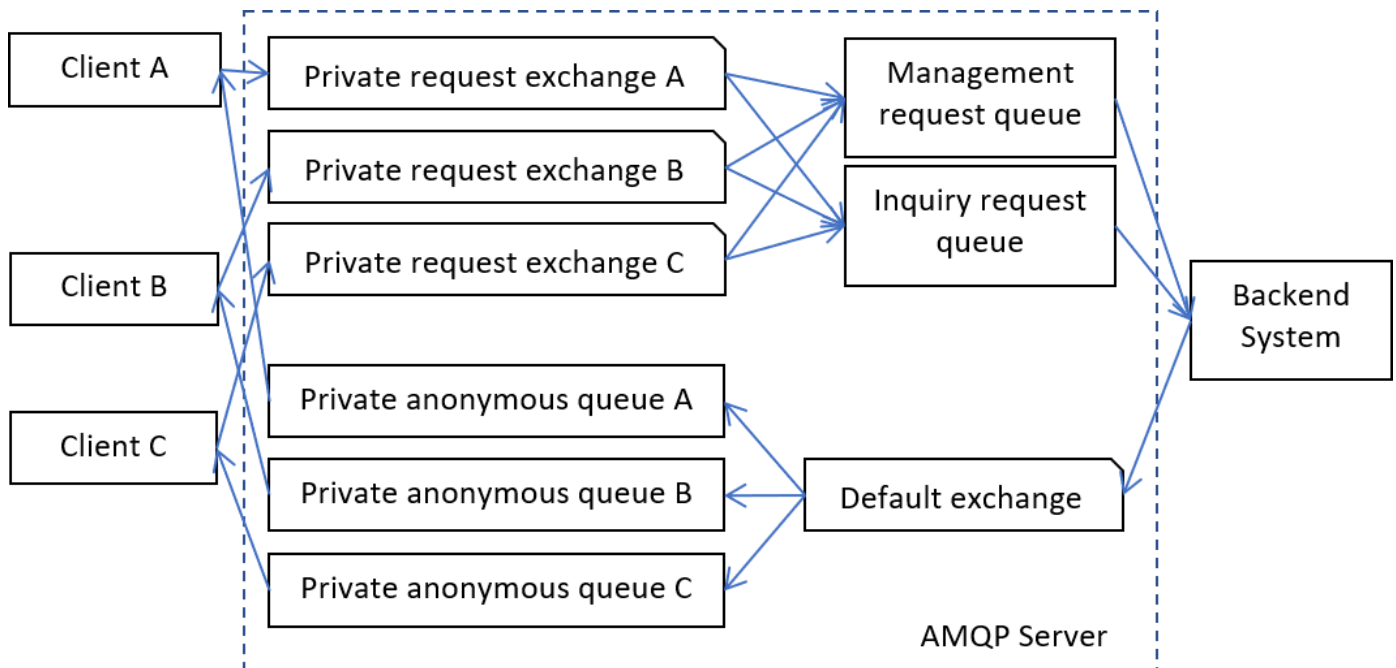
Traders that want to participate in the market send their requests to the backend system. Requests are sent to private trader-specific request exchanges. After processing a request, the backend system sends the response to a private response queue that belongs to the trader that has sent the request.

3.1.1 AMQP Configuration

In case a new trader has been setup by the granting authority, the backend system will synchronize 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 [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 initialization.

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 unroutable 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 maximum length of queue name is 127 characters and its validated² by 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 `OrdEntry` management request) and may therefore be seen by other clients. It is recommendable to keep the correlation-id unrecognizable.
- 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.

The response and broadcast messages sent by the M7 system carry the `timestamp` message property attribute, containing the time message had been sent.

For details about the request and response formats, see [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.

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

The error response is sent to `m7.private.responseQueue.<login-id>.<unique-id>` queue set in `reply-to` attribute. If `reply-to` is not set, error is sent to `m7.broadcastQueue.<login-id>` queue.

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 Planned Changes in the 6.12 Version for creation of the response queues

In version 6.12 the new pattern for the response queue will be `m7.private.responseQueue.<login-id>.queue<1-10>`

The maximum number of the queues will be 10 and client shall declare one private (exclusive) response queue with the following name on the AMQP server:

- `m7.private.responseQueue.<login-id>.queue1`
- `m7.private.responseQueue.<login-id>.queue2`
- `m7.private.responseQueue.<login-id>.queue3`
- `m7.private.responseQueue.<login-id>.queue4`
- `m7.private.responseQueue.<login-id>.queue5`
- `m7.private.responseQueue.<login-id>.queue6`
- `m7.private.responseQueue.<login-id>.queue7`
- `m7.private.responseQueue.<login-id>.queue8`
- `m7.private.responseQueue.<login-id>.queue9`
- `m7.private.responseQueue.<login-id>.queue10`

The following changes will be made in the chapter [How to Send Requests](#) and [Invalid and Unrouteable Requests](#). The rest of the text remains unchanged.

In the chapter [How to Send Requests](#) the text will change in the following sentences.

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

`m7.private.responseQueue.<login-id>.queue<1-10>`

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 queue creation and behavior is according to RabbitMq documentation.

The maximum length of queue name is 127 characters and its validated³ by regexp `^[a-zA-Z0-9-_.:]+$`

- Set the `reply-to` message attribute to the name of the private response queue (`m7.private.responseQueue.<login-id>.queue<1-10>`). See [AMQP Message Properties](#) for the formatting of the reply-to attribute.

In the chapter [Invalid and Unrouteable Requests](#) text will change in the following sentence.

The error response is sent to `m7.private.responseQueue.<login-id>.queue<1-10>` queue set in `reply-to` attribute. If `reply-to` is not set, error is sent to `m7.broadcastQueue.<login-id>.queue` .

3.1.5 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 class name (e.g. `ContractInfoRprt`). The class name for each message is shown in the Message properties summary table header.

3.1.6 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 client does not receive response in timely manner, it is encouraged to re-inquire for such message as described in [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.

3.1.7 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. 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 behavior. 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.

3.1.8 Unacknowledged Requests

A request may not be acknowledged because the connection used to send it is closed (see [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.

3.1.9 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.

3.1.10 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.

3.1.10.1 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).

3.1.10.2 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.

3.1.10.3 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 client and 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.)

3.1.11 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

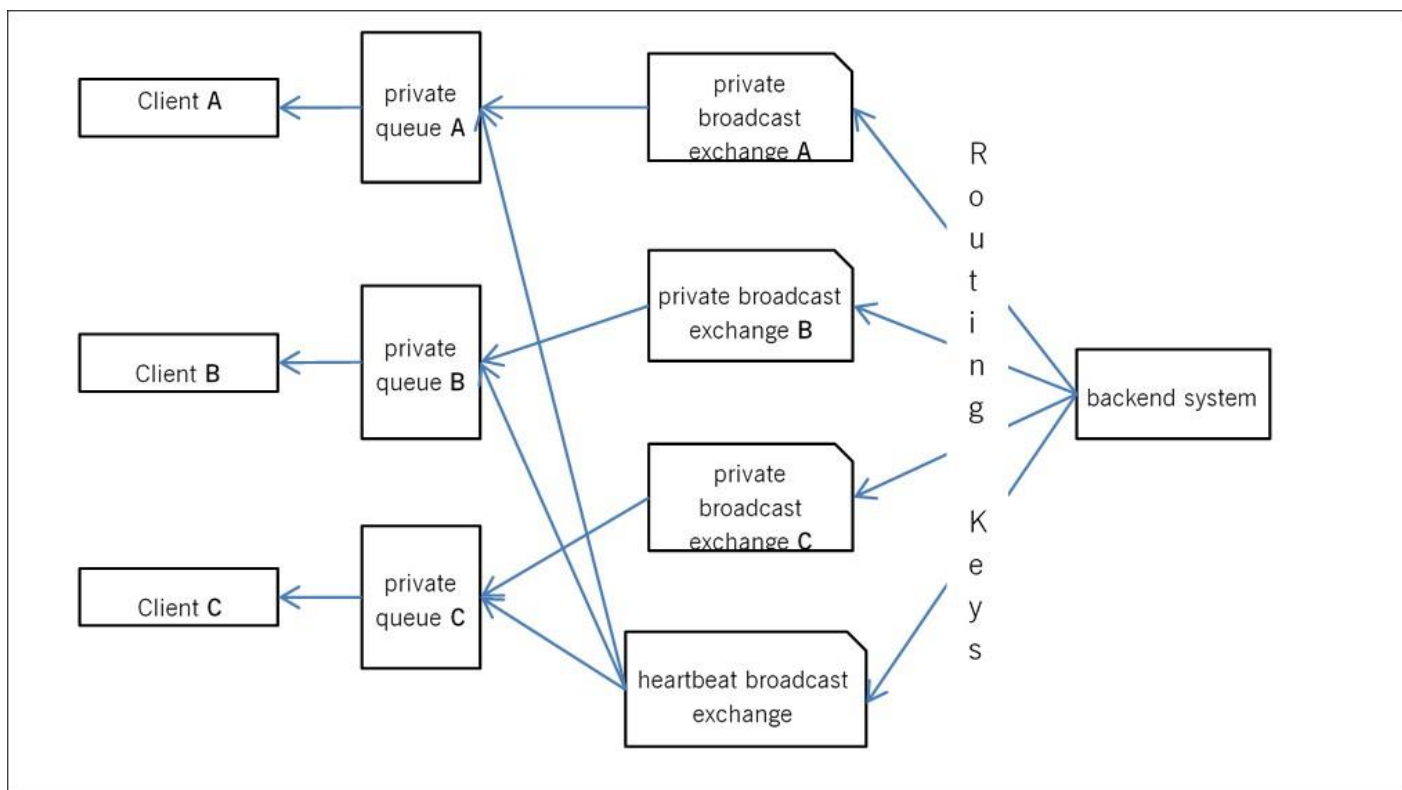
The backend system broadcasts three types of information:

- *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 sends 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 [M7 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 (heartbeat timeout defined when client creates connection) 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 in the AMQP Server. The private client's queue will be named

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

The backend system sends broadcast 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 `group-sequence` . x-m7-

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, while keeping in mind that a sequence reset as described above may occur.

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-synchronization. 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 is 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 Gap Detection and Acknowledgments

Gaps are an inherent part of the communication protocol. They can occur under some conditions and clients have to be able to recover from them by re-connection.

Gap occurrence conditions:

- slow consumer (broadcast queue has configured time to live before timeout)
- broker restart/fail-over (more about failover in *AIP130 - Application Failover*)
- connection loss
- client failures

Client libraries often contain a so-called auto-recovery mode which will not propagate connection errors to upper layers but will seamlessly reconnect to broker. Under those circumstances, gaps will occur because of connection loss and no error is reported. So clients should take this under consideration or have auto-recovery disabled.

3.2.6 Broadcast Distribution from M7 Back-End to AMQP Server

All broadcasts from M7 back-end to AMQP server may go through one or multiple separate connections. The distribution of broadcast messages into each connection is performed based on the routing key. These include for example:

- Several connections for broadcasts with the routing key containing the string `.prddlvr.` (i.e. Public Order Books Delta Report and Implied Order Report);
- A connection for member;
- A connection for balancing group;
- A connection for heartbeats. The heartbeats are sent through the dedicated connection in order to minimize the risk of slowing down their delivery.

For the ordering of the broadcast messages please refer to the chapter [Sequence Counting For Broadcast Messages](#).

Please note the information contained in this paragraph is for informative purposes only and may be subject to changes. This also includes 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.

3.2.7 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.

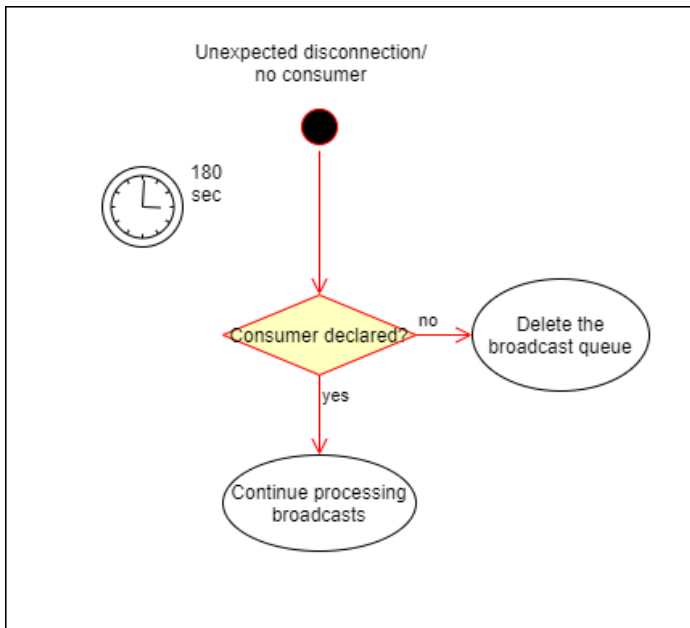
To stop receiving broadcasts, the client should remove the consumer from its private queue.

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.

3.2.7.1 Broadcast queue expiration time

Besides the time to live for messages waiting inside the broadcast queue, a time to live parameter for the existence of broadcast queues themselves is configured ("broadcast queue expiration time") to protect the AMQP server from memory issues.

If no consumer is declared for the queue in question, i.e. no client is consuming/using the queue, it gets deleted by the AMQP Server after the time to live expires.



The current broadcast queue expiration time is set to 180 seconds.

3.2.8 Re-login after unexpected AMQP disconnection

In case of an unexpected AMQP disconnection (e.g. due to network issues), the client is able to reconnect and start consuming from the existing broadcast queue before it reaches the end of the expiration time.

It is recommended to start consuming from the broadcast queue right after having logged in (i.e. right after the reception of the *UserRprt*).

This action should be taken before or in parallel with the action of sending inquiry requests and/or processing the corresponding responses to minimize the risk of losing the broadcasts present in the queue when it reaches the end of the expiration time.

3.2.9 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 [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.

3.2.9.1 Auto Acknowledgments

This is the preferred variant of acknowledgments because it provides higher throughput. Messages are reliably transferred from broker to client via TCP but client does not send explicit AMQP acknowledgments (this saves the line).

3.2.9.2 Manual Acknowledgments

In this mode an AMQP acknowledgment is sent after every message (or after every batch) by client. Broker tracks number of unacknowledged messages. When unacknowledged and new messages reach the configured threshold of messages then client is disconnected from broker because it threatens broker's stability.

An important point to note is that unacknowledged messages are sent after reconnect (when broadcast is not deleted yet). Those messages have the re-delivery flag set to true and are mixed together with new messages so the sequence numbers seems to not be continuous. Client has to deal with this and not treat it as gaps.

3.2.10 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 recognize 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.

3.2.11 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.

3.2.12 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 CA's Website⁴.

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 initialize 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.

Please refer to Java documentation for more information about the `keytool` utility.

4.2 Client Certificate

An organization 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.

The client certificate:

- complies to the X.509v3 specification,
- uses a key length of 2048 bits,
- subject contains a 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.

For further details regarding TLS security, please consult the TLS specification, AMQP specification and <https://www.rabbitmq.com/ssl.html>.

4.3 Authentication

4.3.1 General

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 the cipher 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.

4.3.2 Cipher Suites Supported by the M7 Trading application for API connection

As mentioned previously, after the TLS handshake the entire communication between the client and the AMQP server is encrypted.

The M7 Trading application currently supports the following cipher suites for the API connection:

- ECDHE-ECDSA-AES256-GCM-SHA384
- ECDHE-ECDSA-AES128-GCM-SHA256
- ECDHE-RSA-AES256-GCM-SHA384
- ECDHE-RSA-AES128-GCM-SHA256
- DHE-RSA-AES256-GCM-SHA384
- DHE-RSA-AES128-GCM-SHA256

Disclaimer: The above-mentioned list is based on the last penetration test performed by DBAG in November 2020 and may be subject to change. Any modifications to the list triggered by DBAG will be announced with sufficient lead time, whenever the situation allows it (an example of an exception would be the installation of a security patch).

Nevertheless, it is important to note that also upgrades of the OS and the SSL library may result in some (but not all) of the above listed ciphers not being supported for establishing the communication with the M7 Trading application via API. As DBAG cannot influence the project lifecycle of the third parties' software, it strongly recommends all AMQP clients to perform a simple

connectivity test to verify that the connection with the AMQP server can be established. Such test should be executed at least once for each major release (6.11, 6.12 etc.) during the release UAT phase, to prevent later connectivity issues in a Production environment.

4.4 Authorization

Authorization of client actions occurs on two levels.

4.4.1 Authorization 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 case of insufficient privileges, the attempt to access the resource is immediately (synchronously) rejected by AMQP.

4.4.2 Authorization 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): <ul style="list-style-type: none">- 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 Request-Response Communication)
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 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>i.e. 10 or 4.1.5</i>).

If an `app-version` is provided, M7 checks if it is higher or equal than the minimum version for the given `app-id`. If the `app-version` is lower, the connection is refused. The application also validates that the `app-version` is at most 32 characters long, otherwise, the connection is not allowed. 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 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

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 On-behalf Trading).

Table 1: 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 a 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 represent exactly the database values stored in the backend. The interpretation/display of these values depends on the product attributes *decShiftQty*, *minQty* and *qtyUnit* (see [Product Information Report](#)).

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 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 [Product Information Report](#)).

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-DDThh:mm:ss.sssZ (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

Symbol	Description	Example
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 [System Info Response](#)).

5.1.10 Date Values in Messages

Date values in messages (data type in the message layout tables equaling "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 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 [Standard Message Header](#)). 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 (OrdEntry)
- Order Modify Request (OrdModify)
- Order Request (OrdReq)
- Order Execution Report (OrdExeRprt)
- 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 (OrdLmtReq)

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 summarizes 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: <ul style="list-style-type: none"> - 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: A 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 Request Rate Limit): <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 [Forward compatibility](#).

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 LoginReq

- **Type:** Inquiry Request
- **Routing Keys:** `m7.request.inquiry`
- **Roles:** All
- **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 User Response (a successful login) or Error Response is returned.

Error Response might indicate that a trader with the same credentials is already logged into the backend system.

Login Request has to be sent at least 5 seconds after a successful Logout Request.

XML Tag		Type	m/o	No.	Data Type	Short description
LoginReq		SE	m	1	Structure	
	user	A	m		Char(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 wants to force a login even if a user with the same credentials is already logged in into the backend system.
	disconnectAction	A	m		Char(16)	An action that will be executed in case of an unexpected connection loss. Valid values: <ul style="list-style-type: none"> • 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	1	Structure	Standard header of each message.
	marketId	A	m		Char(255)	Market Identification Code (MIC) of the market to which the request is sent or from which the request originates.
	onBehalfUserId	A	o		Integer	UserID of the target user in case of On-behalf trading (see On-behalf Trading chapter in <i>DFS180</i>).
	clientData	SE	o	0..1	Structure	For order management transactions the client data fields in this section can be used by the client to store information or meta-data about a request. None of this information is used by the backend and it is returned to the client with the response.
	clientDataInt	A	o		Integer	Integer for client's usage.

XML Tag	Type	m/o	No.	Data Type	Short description
clientDataString	A	o		Char(255)	String for client's usage.
clientCorrelationId	A	o		Char(255)	Correlation Id for client's usage.

6.1.2 LogoutReq

- **Type:** Inquiry Request
- **Routing Keys:** `m7.request.inquiry`
- **Roles:** All
- **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	m	1	Structure	
sessionId	A	m		Long	Session id of the trader's session which is passed to the client upon login.
StandardHeader	SE	m	1	Structure	Standard header of each message.
marketId	A	m		Char(255)	Market Identification Code (MIC) of the market to which the request is sent or from which the request originates.
onBehalfUserId	A	o		Integer	UserID of the target user in case of On-behalf trading (see On-behalf Trading chapter in <i>DFS180</i>).
clientData	SE	o	0..1	Structure	For order management transactions the client data fields in this section can be used by the client to store information or meta-data about a request. None of this information is used by the backend and it is returned to the client with the response.
clientDataInt	A	o		Integer	Integer for client's usage.
clientDataString	A	o		Char(255)	String for client's usage.
clientCorrelationId	A	o		Char(255)	Correlation Id for client's usage.

6.1.3 LogoutRprt

- **Type:** Inquiry Response; Broadcast
- **Response to:** LogoutReq (sent to the private response queue)
- **Roles:** All
- **Broadcasted:** Yes
- **Broadcast Routing Keys:** `[schema-version].trdr.[user-id]`
- **Broadcast Audience:** Only the user with the usrId involved

The Logout Report is used:

- As a response to the Logout Request (to the private autogenerated response queue) in which the attribute usrId and

sessionId will be set to respective values.

- 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 in this case as well.
- As a broadcast in case of any other event leading to the logout of the user.

XML Tag		Type	m/o	No.	Data Type	Short description
LogoutRprt		SE	m	1	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.
	forced	A	o		Boolean	Indicates whether a user has been forced to log out. It is 'true' as a consequence of either a concurrent login with the same user credentials or a user password change where subsequently the user is forced to log out. It is 'false' as a response to a regular Logout Request.
	txt	A	o		Char(255)	A text field containing information about the reason of the logout. When M7 identifies that the AMQP connection for any reason is closed while the client has not sent a LogoutReq, it will broadcast a LogoutRprt with <code>txt=INACTIVITY</code> .
StandardHeader		SE	m	1	Structure	Standard header of each message.
	marketId	A	m		Char(255)	Market Identification Code (MIC) of the market to which the request is sent or from which the request originates.
	onBehalfUserId	A	o		Integer	UserID of the target user in case of On-behalf trading (see On-behalf Trading chapter in <i>DFS180</i>).
	clientData	SE	o	0..1	Structure	For order management transactions the client data fields in this section can be used by the client to store information or meta-data about a request. None of this information is used by the backend and it is returned to the client with the response.
	clientDataInt	A	o		Integer	Integer for client's usage.
	clientDataString	A	o		Char(255)	String for client's usage.
	clientCorrelationId	A	o		Char(255)	Correlation Id for client's usage.

6.1.4 SystemInfoReq

- Type:** Inquiry Request
- Routing Keys:** `m7.request.inquiry`
- Roles:** All
- Request Limits:** 14/70

The System Info Request is used to get general information about the backend system.

XML Tag			Type	m/o	No.	Data Type	Short description
SystemInfoReq			SE	m	1	Structure	
	StandardHeader		SE	m	1	Structure	Standard header of each message.
		marketId	A	m		Char(255)	Market Identification Code (MIC) of the market to which the request is sent or from which the request originates.
		onBehalfUserId	A	o		Integer	UserID of the target user in case of On-behalf trading (see On-behalf Trading chapter in <i>DFS180</i>).
		clientData	SE	o	0..1	Structure	For order management transactions the client data fields in this section can be used by the client to store information or meta-data about a request. None of this information is used by the backend and it is returned to the client with the response.
		clientDataInt	A	o		Integer	Integer for client's usage.
		clientDataString	A	o		Char(255)	String for client's usage.
		clientCorrelationId	A	o		Char(255)	Correlation Id for client's usage.

6.1.5 SystemInfoResp

- **Type:** Inquiry Response
- **Response to:** SystemInfoReq (sent to the private response queue)
- **Broadcasted:** No
- **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	m	1	Structure	
		backendVersion	A	m		Char(255)	The version number of the backend system.
		backendTimeZone	A	m		Char(255)	The time zone identifier of the time zone the backend system runs in.
		backendMarketTimeZone	A	m		Char(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 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		Char(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.

XML Tag		Type	m/o	No.	Data Type	Short description
	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		Char(∞)	<p>A list of the backend features available. If the feature is not in the list, it is disabled on backend side. Valid values:</p> <ul style="list-style-type: none"> • SETTLEMENT: The Update Settlement Information message is supported by the system 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. • QUOTE-ORDERS: Mass Quote requests are supported by the system. • QUOTE-REQUESTS: Order Quote requests are supported by the system. • OPEN-CLOSE-INDICATOR: The Open-close indicator is supported in the Order entry and the following messages. • TRADING-LIMIT: Cash limit requests are supported by the system.
	allowedClearingAcctTypes	A	m		Char(255)	Comma separated valid values for the <i>clearingAcctType</i> attribute in e.g. <i>OrdEntry</i> message. Example (spot markets): A, P
	StandardHeader	SE	m	1	Structure	Standard header of each message.
	marketId	A	m		Char(255)	Market Identification Code (MIC) of the market to which the request is sent or from which the request originates.
	onBehalfUserId	A	o		Integer	UserID of the target user in case of On-behalf trading (see On-behalf Trading chapter in <i>DFS180</i>).
	clientData	SE	o	0..1	Structure	For order management transactions the client data fields in this section can be used by the client to store information or meta-data about a request. None of this information is used by the backend and it is returned to the client with the response.
	clientDataInt	A	o		Integer	Integer for client's usage.
	clientDataString	A	o		Char(255)	String for client's usage.
	clientCorrelationId	A	o		Char(255)	Correlation Id for client's usage.
	RequestLimitList	SE	m	1	Structure	List of request limits
	RequestLimit	SE	o	0..n	Structure	Request limit
	message	A	m		Char(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)

6.1.6 AckResp

- **Type:** Management Response
- **Response to:** OrdEntry; OrderModify; ModifyAllOrders; TradeRecallReq: (sent to the private response queue see [Request-Response communication](#))
- **Roles:** Trader, Market Operation, Broker, Sales
- **Broadcasted:** No
- **Broadcast Routing Keys:** –

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 behavior that regards the order of sending. In theory, it is possible that, due to the above stated reasons, other types of messages are received earlier than the AckResp.

Note: Because of new architecture constraints, clOrdId has been removed from this message.

XML Tag		Type	m/o	No.	Data Type	Short description
AckResp		SE	m	1	Structure	
	StandardHeader	SE	m	1	Structure	Standard header of each message.
	marketId	A	m		Char(255)	Market Identification Code (MIC) of the market to which the request is sent or from which the request originates.
	onBehalfUserId	A	o		Integer	UserID of the target user in case of On-behalf trading (see On-behalf Trading chapter in <i>DFS180</i>).
	clientData	SE	o	0..1	Structure	For order management transactions the client data fields in this section can be used by the client to store information or meta-data about a request. None of this information is used by the backend and it is returned to the client with the response.
	clientDataInt	A	o		Integer	Integer for client's usage.
	clientDataString	A	o		Char(255)	String for client's usage.
	clientCorrelationId	A	o		Char(255)	Correlation Id for client's usage.

6.1.7 ErrResp

- **Type:** Inquiry Response; Management Response; Broadcast
- **Response to:** All (sent to the private response queue)
- **Roles:** All
- **Broadcasted:** Yes
- **Broadcast Routing Keys:** `[schema-version].trdr.[user-id]`
- **Broadcast Audience:** The connected users whose action caused the asynchronous error (i.e. OrdEntry whilst not having a sufficient cash limit).

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 text resource with the transferred variables. This way the localization of the error text can be completed on the client side (the server side provides no localization in the err texts).

Example of ErrResp

```
<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	m	1	Structure	
	StandardHeader		SE	m	1	Structure	Standard header of each message.
		marketId	A	m		Char(255)	Market Identification Code (MIC) of the market to which the request is sent or from which the request originates.
		onBehalfUserId	A	o		Integer	UserID of the target user in case of On-behalf trading (see On-behalf Trading chapter in <i>DFS180</i>).
		clientData	SE	o	0..1	Structure	For order management transactions the client data fields in this section can be used by the client to store information or meta-data about a request. None of this information is used by the backend and it is returned to the client with the response.
		clientDataInt	A	o		Integer	Integer for client's usage.
		clientDataString	A	o		Char(255)	String for client's usage.
		clientCorrelationId	A	o		Char(255)	Correlation Id for client's usage.
	Error		SE	m	1..n	Structure	The single error element
		err	A	m		Char(255)	The error message for this error. Always in the English language with variables already replaced by their values.

XML Tag					Type	m/o	No.	Data Type	Short description
				errCode	A	m		Integer	The error code of the error. In case an error message does not have a specific error code, the value of 0 will be used.
				clOrdId	A	o		Char(40)	The client order id
				VarList	SE	o	0..1	Structure	A list of variables used in the err text field
				Var	SE	o	0..n	Structure	Structure containing information on variables
				id	A	m		Integer	In Error Response and DeleteQuotesResp, 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		Char(255)	Value of the variable (e.g. Acct1)

6.1.8 ChgPwdReq

- **Type:** Management Request
- **Routing Keys:** `m7.request.management`
- **Roles:** All
- **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.

AckResp (confirmation that ChgPwdReq was received) is sent, and 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 and second AckResp (confirmation that password was changed) are sent. *Note that shutdown signal might be sent before or after second AckResp.* The user then can re-login 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.

New password shall be different than 6 previously used passwords (this is configured in LDAP settings). 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 expire after a certain length of time ⁵ 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 ⁶ 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}`.

XML Tag		Type	m/o	No.	Data Type	Short description
ChgPwdReq		SE	m	1	Structure	
	currentPwd	A	m		Char(64)	The current password
	newPwd	A	m		Char(64)	A new password
StandardHeader		SE	m	1	Structure	Standard header of each message.
	marketId	A	m		Char(255)	Market Identification Code (MIC) of the market to which the request is sent or from which the request originates.
	onBehalfUserId	A	o		Integer	UserID of the target user in case of On-behalf trading (see On-behalf Trading chapter in <i>DFS180</i>).
	clientData	SE	o	0..1	Structure	For order management transactions the client data fields in this section can be used by the client to store information or meta-data about a request. None of this information is used by the backend and it is returned to the client with the response.
	clientDataInt	A	o		Integer	Integer for client's usage.
	clientDataString	A	o		Char(255)	String for client's usage.
	clientCorrelationId	A	o		Char(255)	Correlation Id for client's usage.

6.2 Order Entry and Maintenance

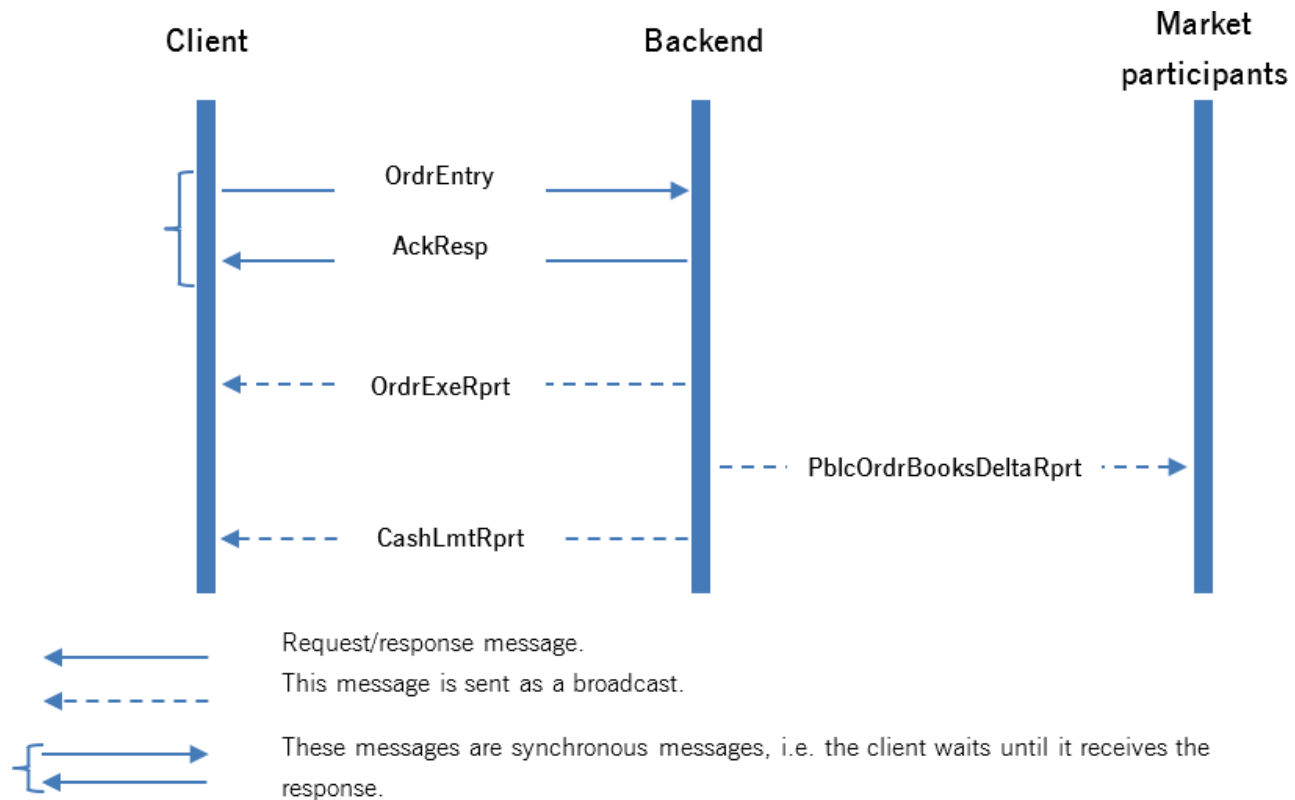
The messages described in this section are used to enter and maintain orders. These messages can be used only by trading participants and are not available for Market Operation users.

6.2.1 OrdEntry

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

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.

The message flow is shown in the below figure [7](#):



Note: 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.

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).

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

XML Tag		Type	m/o	No.	Data Type	Short description
	listExecInst	A	o		Char(5)	<p>Defines the execution instruction for the whole list of orders:</p> <p>NONE: All orders are treated independently. This value is also to be used if the OrdEntry message contains only one order.</p> <p>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.</p> <ul style="list-style-type: none"> • 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. Please check its availability with the System Info Request and the Product Information Request. • IMPL: Implied order. This value is obsolete and may be removed in the next versions.
StandardHeader		SE	m	1	Structure	Standard header of each message.
	marketId	A	m		Char(255)	Market Identification Code (MIC) of the market to which the request is sent or from which the request originates.
	onBehalfUserId	A	o		Integer	UserId of the target user in case of On-behalf trading (see On-behalf Trading chapter in <i>DFS180</i>).
clientData		SE	o	0..1	Structure	For order management transactions the client data fields in this section can be used by the client to store information or meta-data about a request. None of this information is used by the backend and it is returned to the client with the response.
	clientDataInt	A	o		Integer	Integer for client's usage.
	clientDataString	A	o		Char(255)	String for client's usage.
	clientCorrelationId	A	o		Char(255)	Correlation Id for client's usage.
OrdList		SE	m	1	Structure	List of all orders contained in the basket.
	Ord	SE	m	1..n	Structure	
	clearingAcctType	A	m		Char(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).
	acctId	A	m		Char(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	o		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	o		Char(255)	Product identifier. This is mandatory in case that the contract Id is omitted.

XML Tag			Type	m/o	No.	Data Type	Short description
		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.
		stopPx	A	o		Long	Stop price for stop limit orders. Mandatory if the type= S .
		ppd	A	o		Long	<p>Peak price delta for Iceberg orders.</p> <p>The ppd of buy orders must be smaller than or equal to zero.</p> <p>The ppd of sell orders must be greater than or equal to zero.</p> <p>If it is omitted the system will assume a value of 0,00.</p>
		qty	A	m		Integer	Contains the total quantity of the order. In case of an Iceberg order this field corresponds to the hidden quantity + display quantity.
		displayQty	A	o		Integer	Used to define the display quantity of an Iceberg Order. This field is only required when the type= I .
		ordrExeRestriction	A	o		Char(3)	<p>Execution restriction of the order. Valid values:</p> <ul style="list-style-type: none"> 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): Deprecated. Orders with this restriction will be rejected. This value will be removed in the next release. <p>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.</p> <p>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.</p>
		txt	A	o		Char(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	o		Char(16)	<p>Defines the delivery area of the order.</p> <p>This is mandatory for exchanges with a multiple delivery area setup.</p>
		clOrdId	A	o		Char(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	o		Boolean	This flag indicates if the entered order is a pre-arranged order or not.
		preArrangedAcct	A	o		Char(32)	This is required in case of a pre-arranged order. It contains the account of the counterpart.

XML Tag			Type	m/o	No.	Data Type	Short description
		type	A	m		Char(1)	Order type: <ul style="list-style-type: none"> 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 and 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	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. For "GFS" orders, the field is populated with date and time of the end of trading phase, or with the date and time of the contract expiry point, depending on which of the two dates comes earlier.
		validityRes	A	o		Char(3)	Validity restriction of the order. If this field is omitted, the order will be treated as a <i>Good for Session</i> order. Valid values: <ul style="list-style-type: none"> GFS (Good for trading session): The order rests in the order book until it is either executed, removed by the user or until start of next non-trading (closed) phase of the underlying contract. GTD (Good till date): The order rests in the order book until the date specified in the vldtyDate field. NON (No validity restriction): Mandatory for orders with execution restriction codes FOK or IOC.
		state	A	o		Char(4)	The desired state of the order. <ul style="list-style-type: none"> 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	o		Char(1)	Mandatory for Futures product and Cross product spreads. For other Commodities is not present. Valid values: <ul style="list-style-type: none"> O: Open position indicator C: Close position indicator
		exGTD	A	o		DateTime	This attribute is deprecated and should not be used. Its value is ignored and will be deleted in the M7 6.12 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. (not relevant for SEMOpX) Default value: false.

XML Tag				Type	m/o	No.	Data Type	Short description
			ClgHse	SE	o	0..n	Structure	<p>List of the Clearing House elements.</p> <p>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.</p> <p>This is mandatory if the clearing house functionality is set-up on the exchange. See SystemInfoResp.</p>
			clgAcctId	A	m		Integer	Clearing Account Id.
			clgHseCode	A	m		Char(255)	Clearing House Code.

6.2.2 OrdModify

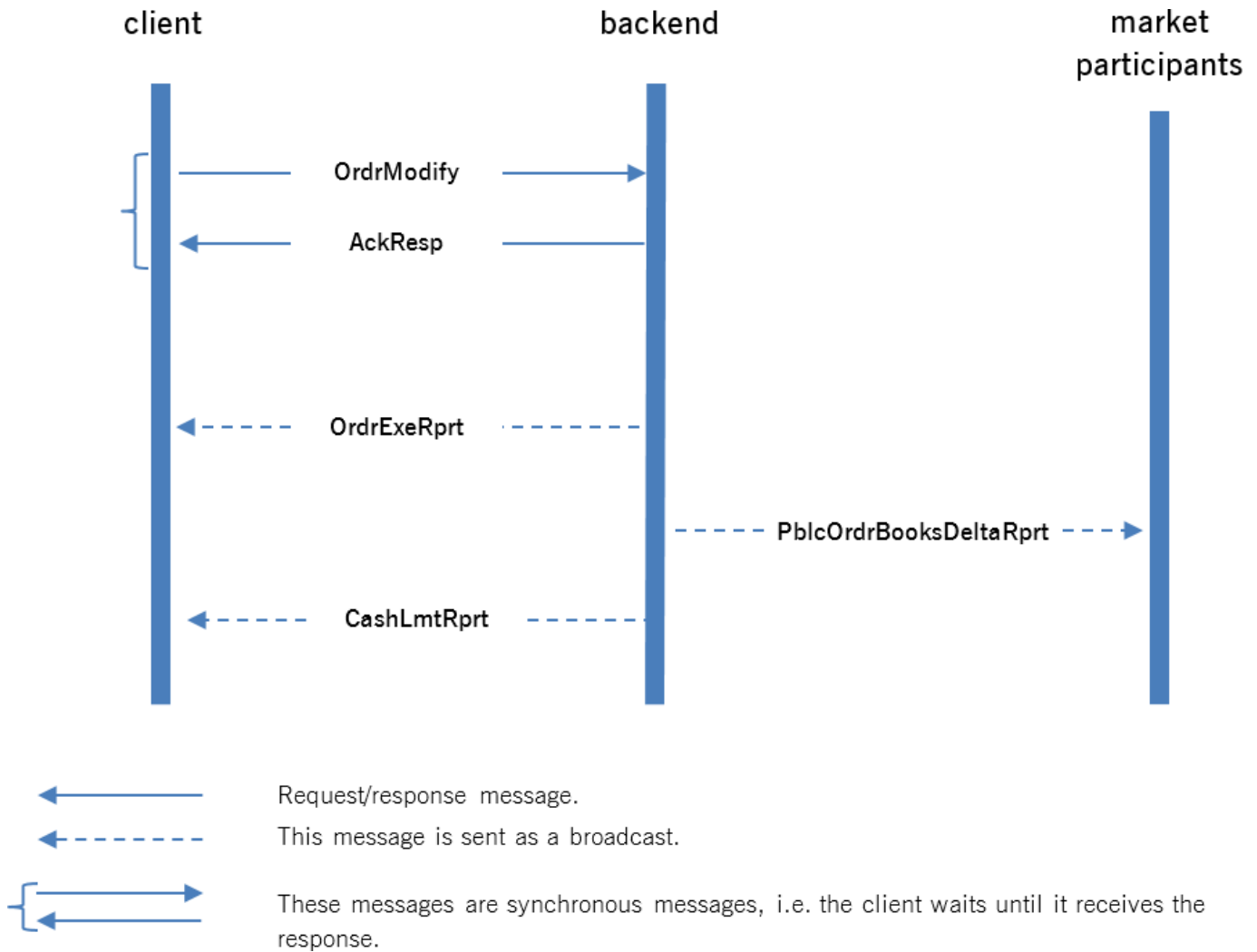
- **Type:** Management Request
- **Routing Keys:** `m7.request.management`
- **Roles:** Trader, Market Operation

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 is removed by the system and a new order with a new order id is entered instead. As a result, after an order modification is made which has an impact on the execution priority, an OrdExeRprt message containing two records is sent:

- One record details the deletion of the order that was present in the system before the modification.
- Another record details the addition of a new order that was just created in the system to reflect the modification.

Due to performance optimization, the order of the above two records in the OrdExeRprt is not given (the deletion of the old order can be before or after the addition of the new one).

The message flow is shown in the below figure:



XML Tag		Type	m/o	No.	Data Type	Short description
OrdModify		SE	m	1	Structure	
	ordrModType	A	m		Char(4)	Types of order modification. Offers the possibility to activate, deactivate, modify or delete all orders contained in the basket. <ul style="list-style-type: none"> • 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.
StandardHeader		SE	m	1	Structure	Standard header of each message.
	marketId	A	m		Char(255)	Market Identification Code (MIC) of the market to which the request is sent or from which the request originates.

XML Tag			Type	m/o	No.	Data Type	Short description
		onBehalfUserId	A	o		Integer	UserID of the target user in case of On-behalf trading (see On-behalf Trading chapter in <i>DFS180</i>).
		clientData	SE	o	0..1	Structure	For order management transactions the client data fields in this section can be used by the client to store information or meta-data about a request. None of this information is used by the backend and it is returned to the client with the response.
		clientDataInt	A	o		Integer	Integer for client's usage.
		clientDataString	A	o		Char(255)	String for client's usage.
		clientCorrelationId	A	o		Char(255)	Correlation Id for client's usage.
		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		Char(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.
		stopPx	A	o		Long	Stop price for stop limit orders. Mandatory if the type= S .
		ppd	A	o		Long	Peak price delta for Iceberg orders. <ul style="list-style-type: none"> The ppd of buy orders must be smaller 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	o		Integer	Used to define the display quantity of an Iceberg Order.
		ordrExeRestriction	A	o		Char(3)	Execution restriction of the order. Valid values: <ul style="list-style-type: none"> 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): Deprecated. Orders with this restriction will be rejected. This value will be removed in the next release. 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.

XML Tag			Type	m/o	No.	Data Type	Short description
		txt	A	o		Char(250)	Text entered by the client. This text will not be modified by the backend.
		type	A	m		Char(1)	Order type: <ul style="list-style-type: none"> • 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 and lift matcher • Q: Quote order • W: Indicative quote order
		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 <i>Good for Session</i> order. Valid values: <ul style="list-style-type: none"> • GFS (Good for trading session): The order rests in the order book until it is either executed, removed by the user or until start of next non-trading (closed) phase of the underlying contract. • GTD (Good till date): The order rests in the order book until the date specified in the vldtyDate field. • NON (No validity restriction): Mandatory for orders with execution restriction codes 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	o		Char(1)	This is mandatory for Futures and Cross product spreads. For other Commodities it is not used. Valid values: <ul style="list-style-type: none"> • O: Open position indicator • C: Close position indicator
		exGTD	A	o		DateTime	This attribute is deprecated and should not be used. Its value is ignored and will be deleted in the M7 6.12 version.
		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. (not relevant for SEMOpX)

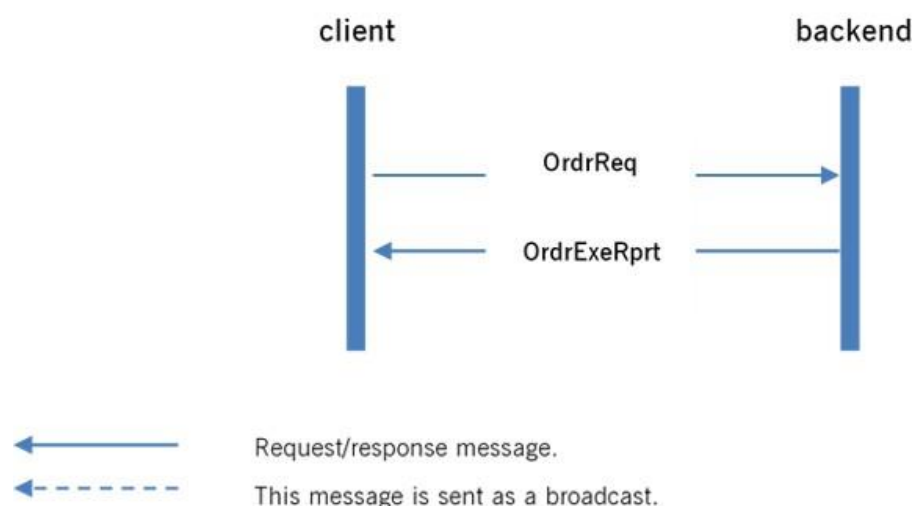
XML Tag				Type	m/o	No.	Data Type	Short description
			ClgHse	SE	o	0..n	Structure	<p>List of the Clearing House elements.</p> <p>The priority order will be the same as the order of the Clearing House in the xml message. The Clearing House specified first will have the top priority, and the Clearing House specified at the end of the file will have the least priority.</p> <p>This is mandatory if the clearing house functionality is set-up on the exchange. See SystemInfoResp.</p>
			clgAcctId	A	m		Integer	Clearing Account Id.
			clgHseCode	A	m		Char(255)	Clearing House Code.

6.2.3 OrdReq

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

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

The message flow is shown in the below figure:



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

XML Tag		Type	m/o	No.	Data Type	Short description
	acctId	A	o		Char(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
StandardHeader		SE	m	1	Structure	Standard header of each message.
	marketId	A	m		Char(255)	Market Identification Code (MIC) of the market to which the request is sent or from which the request originates.
	onBehalfUserId	A	o		Integer	UserID of the target user in case of On-behalf trading (see On-behalf Trading chapter in <i>DFS180</i>).
	clientData	SE	o	0..1	Structure	For order management transactions the client data fields in this section can be used by the client to store information or meta-data about a request. None of this information is used by the backend and it is returned to the client with the response.
	clientDataInt	A	o		Integer	Integer for client's usage.
	clientDataString	A	o		Char(255)	String for client's usage.
	clientCorrelationId	A	o		Char(255)	Correlation Id for client's usage.
	prodName	CE	o	0..1000	Char(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.

6.2.4 OrdExeRprt

- **Type:** Management Response; Broadcast
- **Response to:** OrdEntry; OrdModify; OrdReq; ModifyAllOrdrs; (sent to the private response queue)
- **Broadcast:** Yes
- **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.

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
OrdExeRprt		SE	m	1	Structure	
	listExecInst	A	o		Char(5)	Defines the execution instruction for a whole list of orders.
	listId	A	o		Long	ID for the basket determined by the backend.
StandardHeader		SE	m	1	Structure	Standard header of each message.
	marketId	A	m		Char(255)	Market Identification Code (MIC) of the market to which the request is sent or from which the request originates.
	onBehalfUserId	A	o		Integer	UserID of the target user in case of On-behalf trading (see On-behalf Trading chapter in <i>DFS180</i>).
	clientData	SE	o	0..1	Structure	For order management transactions the client data fields in this section can be used by the client to store information or meta-data about a request. None of this information is used by the backend and it is returned to the client with the response.
	clientDataInt	A	o		Integer	Integer for client's usage.
	clientDataString	A	o		Char(255)	String for client's usage.
	clientCorrelationId	A	o		Char(255)	Correlation Id for client's usage.
OrdList		SE	o	0..1	Structure	
	Ord	SE	o	0..n	Structure	
	ordId	A	m		Long	Order Id as returned by the backend system.
	initialOrdId	A	m		Long	In the event of an order modification, this value contains the Id of the first order in the modification chain.
	parentOrdId	A	o		Long	In the event of an order modification, this field contains the Id of the modified order.
	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).
	acctId	A	m		Char(32)	Account for which the order was entered.
	contractId	A	m		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 event of an order in a user-defined contract.
	side	A	m		Char(4)	Defines on which side of the market the order is entered. Valid values: <ul style="list-style-type: none"> • BUY: Buy order. • SELL: Sell order.
	px	A	m		Long	Limit price of the order.
	stopPx	A	o		Long	Stop price for stop limit orders.
	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.

XML Tag			Type	m/o	No.	Data Type	Short description
		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: <ul style="list-style-type: none"> • NON: No restriction. This is the default. • FOK (Fill or Kill): The order is immediately fully executed or deleted. • IOC (Immediate and cancel): The order is executed immediately to its maximum extend. In case of a partial execution, the remaining volume is removed from the order book. • 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): Deprecated. Orders with this restriction will be rejected. This value will be removed in the next release.
		txt	A	o		Char(250)	Text entered by the client. This text will not be modified by the backend.
		dlvryAreaId	A	m		Char(16)	Defines the delivery area of the order.
		clOrdId	A	o		Char(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		Char(32)	The account of the counterparty in case of a pre-arranged order.
		type	A	m		Char(1)	Order type: <ul style="list-style-type: none"> • 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 and lift matcher • Q: Quote order • W: Indicative quote order

XML Tag			Type	m/o	No.	Data Type	Short description
		state	A	m		Char(4)	<p>The current state of the order in the system. Valid values:</p> <ul style="list-style-type: none"> • HIBE: The order is entered into the backend system but is not exposed to the market. • ACTI: The order is entered and is immediately exposed to the market for execution. • IACT: The order is deleted. •
		aggressorIndicator	A	o		Char(1)	<p>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.</p> <ul style="list-style-type: none"> • Y - Trade aggressor • N - Trade originator •
		usrCode	A	m		Char(6)	<p>The user code of the user performing the last successful action on the order.</p>
		revisionNo	A	m		Long	<p>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).</p>
		timestmp	A	m		DateTime	<p>The timestamp of the order entry as determined by the backend. This timestamp determines the execution priority in case of identical limit prices.</p>
		validityDate	A	o		DateTime	<p>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.</p>
		validityRes	A	o		Char(3)	<p>Validity restriction of the order. If this field is omitted, the order will be treated as a <i>Good for Session</i> order. Valid values:</p> <ul style="list-style-type: none"> • GFS (Good for trading session): The order rests in the order book until it is either executed, removed by the user or until start of next non-trading (closed) phase of the underlying contract. • GTD (Good till date): The order rests in the order book until the date specified in the vldtyDate field. • NON (No validity restriction): Mandatory for orders with the execution restriction <i>FOK</i> or <i>IOC</i>.

XML Tag			Type	m/o	No.	Data Type	Short description
		action	A	m		Char(255)	<p>Lists the action code. Valid values:</p> <ul style="list-style-type: none"> • 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. • SERR: The order validation failed on SOB side, or the request sent to SOB timed out. This is only valid for remote orders. • SNAV: Order state is unknown due to SOB unavailability. This is only valid for remote orders. • QADD: Quote was added • QFEX: Quote was fully executed • QPEX: Quote was partially executed
		lastUpdateUsrInfo	A	m		Char(255)	<p>Information (concatenation of accountId and usrCode) about the user who last updated the order, either on their own or on behalf.</p> <p>If the <i>action</i> 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 (TTTTTTTTTTTTSYSTEM).</p>
		openCloseInd	A	o		Char(1)	<p>Mandatory for Futures and Cross product spreads. For other Commodities the value is not used. Valid values:</p> <ul style="list-style-type: none"> • O: Open position indicator • C: Close position indicator
		brokerUsrId	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.
		remoteOrdId	A	o		Long	The Order Id as returned by the remote backend system (not relevant for SEMOpX)
		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 <i>text</i> field).

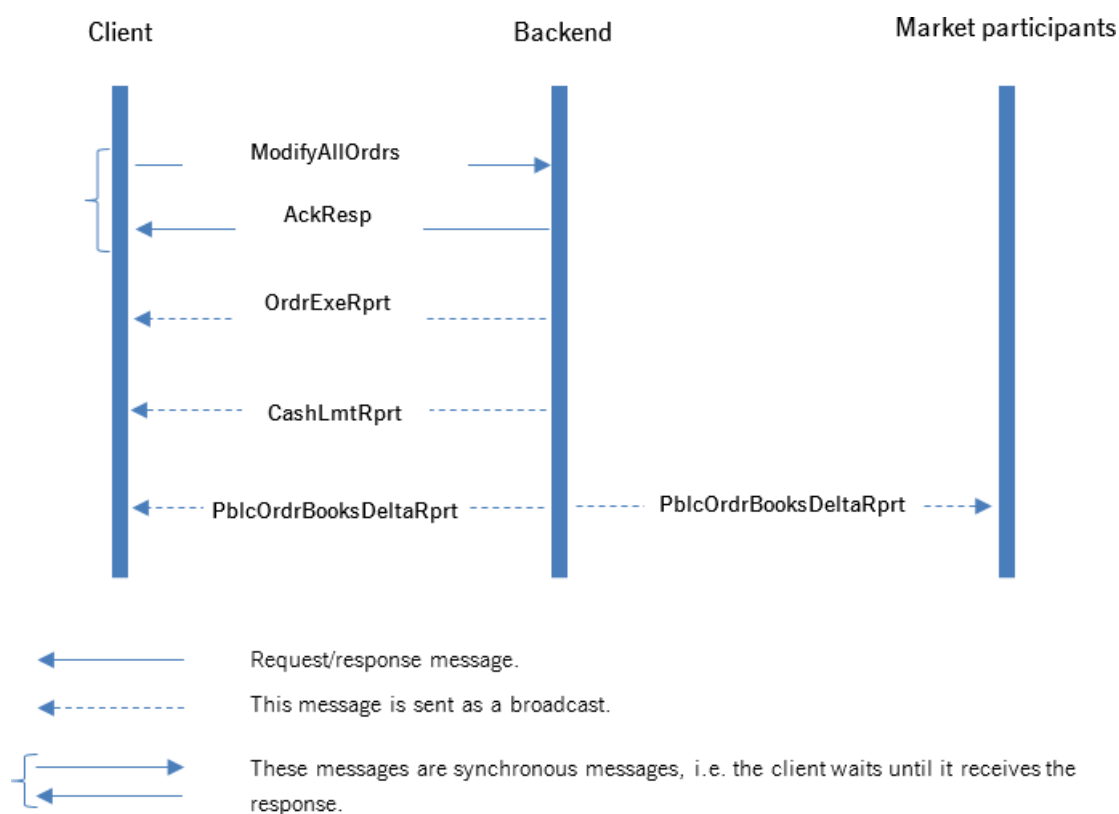
XML Tag				Type	m/o	No.	Data Type	Short description
			remoteLastUpdateTm	A	o		DateTime	The timestamp of the last modification of the order object in the remote backend system.
			preAotId	A	o		Long	Local order ID of the remote order that this order originated from during AOT. The field is populated 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 (not relevant for SEMOpX). Default value: false.
			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. This is mandatory if the clearing house functionality is set-up on the exchange. See SystemInfoResp.
			clgAcctId	A	m		Integer	Clearing Account Id.
			clgHseCode	A	m		Char(255)	Clearing House Code.

6.2.5 ModifyAllOrdrs

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

Modify All Orders message is used to activate, deactivate or delete all orders belonging to an account or a trader or deactivate all orders belonging to a member. Only one of the attributes mbrId, usrId or acctId must 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:



The message flow for a Modify All Orders message

XML Tag	Type	m/o	No.	Data Type	Short description
ModifyAllOrdrs	SE	m	1	Structure	
mbrld	A	o		Char(5)	Unique identifier of a member. <i>Only one of attributes mbrld, usrld or acctld must be provided.</i>
usrld	A	o		Integer	Unique identifier of a user <i>Only one of attributes mbrld, usrld or acctld must be provided.</i>
acctld	A	o		Char(32)	Unique identifier of an account. <i>Only one of attributes mbrld, usrld or acctld must be provided.</i>
dlvryAreald	A	o		Char(16)	Orders for the given usrld and list of delivery areas in dlvryAreald will be Deactivated or Deleted. This element can only be supplied when usrld is provided in the message. If left out, all delivery areas assigned to usrld are affected.

XML Tag	Type	m/o	No.	Data Type	Short description
ordrModType	A	m		Char(4)	Types for modification of multiple orders, in comparison with single modification the type MODI is not allowed here. The following values are allowed: ACTI : Activate all orders. Already active orders are ignored. Not available for modifications on member level (mbrld provided). 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. <ul style="list-style-type: none"> DELE: Deletes all orders. Not available for modifications on member level (mbrld provided).
inclPreArranged	A	m		Boolean	Specifies if pre-arranged orders should be modified or not.
StandardHeader	SE	m	1	Structure	Standard header of each message.
marketId	A	m		Char(255)	Market Identification Code (MIC) of the market to which the request is sent or from which the request originates.
onBehalfUserId	A	o		Integer	UserID of the target user in case of On-behalf trading (see On-behalf Trading chapter in <i>DFS180</i>).
clientData	SE	o	0..1	Structure	For order management transactions the client data fields in this section can be used by the client to store information or meta-data about a request. None of this information is used by the backend and it is returned to the client with the response.
clientDataInt	A	o		Integer	Integer for client's usage.
clientDataString	A	o		Char(255)	String for client's usage.
clientCorrelationId	A	o		Char(255)	Correlation Id for client's usage.
prodName	CE	o	0..1000	Char(255)	Only orders for the given products will be modified. This element can only be supplied when an account is provided.

6.2.6 OrdrLmtReq

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

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	

XML Tag		Type	m/o	No.	Data Type	Short description
	mbrId	A	o		Char(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
StandardHeader		SE	m	1	Structure	Standard header of each message.
	marketId	A	m		Char(255)	Market Identification Code (MIC) of the market to which the request is sent or from which the request originates.
	onBehalfUserId	A	o		Integer	UserID of the target user in case of On-behalf trading (see On-behalf Trading chapter in <i>DFS180</i>).
clientData		SE	o	0..1	Structure	For order management transactions the client data fields in this section can be used by the client to store information or meta-data about a request. None of this information is used by the backend and it is returned to the client with the response.
	clientDataInt	A	o		Integer	Integer for client's usage.
	clientDataString	A	o		Char(255)	String for client's usage.
	clientCorrelationId	A	o		Char(255)	Correlation Id for client's usage.
	ProdName	CE	o	0..50	Char(255)	ProductName

6.2.7 OrdLmtRprt

- **Type:** Inquiry Response; Broadcast
- **Response to:** OrdLmtReq (sent to the private response queue)
- **Broadcast:** Yes
- **Routing Keys:** `[schema-version].mbr.[mbrId]`
- **Broadcast audience:** All traders from particular member, Admins
- **Roles:** Trader, Market Operation, Broker, Market Maker

The OrdLmtResp is returned as a response to the OrdLmtReq 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 the member concerned.

XML Tag		Type	m/o	No.	Data Type	Short description
OrdLmtRprt		SE	m	1	Structure	
StandardHeader		SE	m	1	Structure	Standard header of each message.
	marketId	A	m		Char(255)	Market Identification Code (MIC) of the market to which the request is sent or from which the request originates.
	onBehalfUserId	A	o		Integer	UserID of the target user in case of On-behalf trading (see On-behalf Trading chapter in <i>DFS180</i>).

XML Tag				Type	m/o	No.	Data Type	Short description
			clientData	SE	o	0..1	Structure	For order management transactions the client data fields in this section can be used by the client to store information or meta-data about a request. None of this information is used by the backend and it is returned to the client with the response.
			clientDataInt	A	o		Integer	Integer for client's usage.
			clientDataString	A	o		Char(255)	String for client's usage.
			clientCorrelationId	A	o		Char(255)	Correlation Id for client's usage.
			MbrList	SE	m	1	Structure	List of the members
			Mbr	SE	o	0..n	Structure	Member element
			mbrId	A	m		Char(5)	Member Id.
			OrdLmtList	SE	m	1	Structure	List of order limits
			OrdLmt	SE	o	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		Char(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.

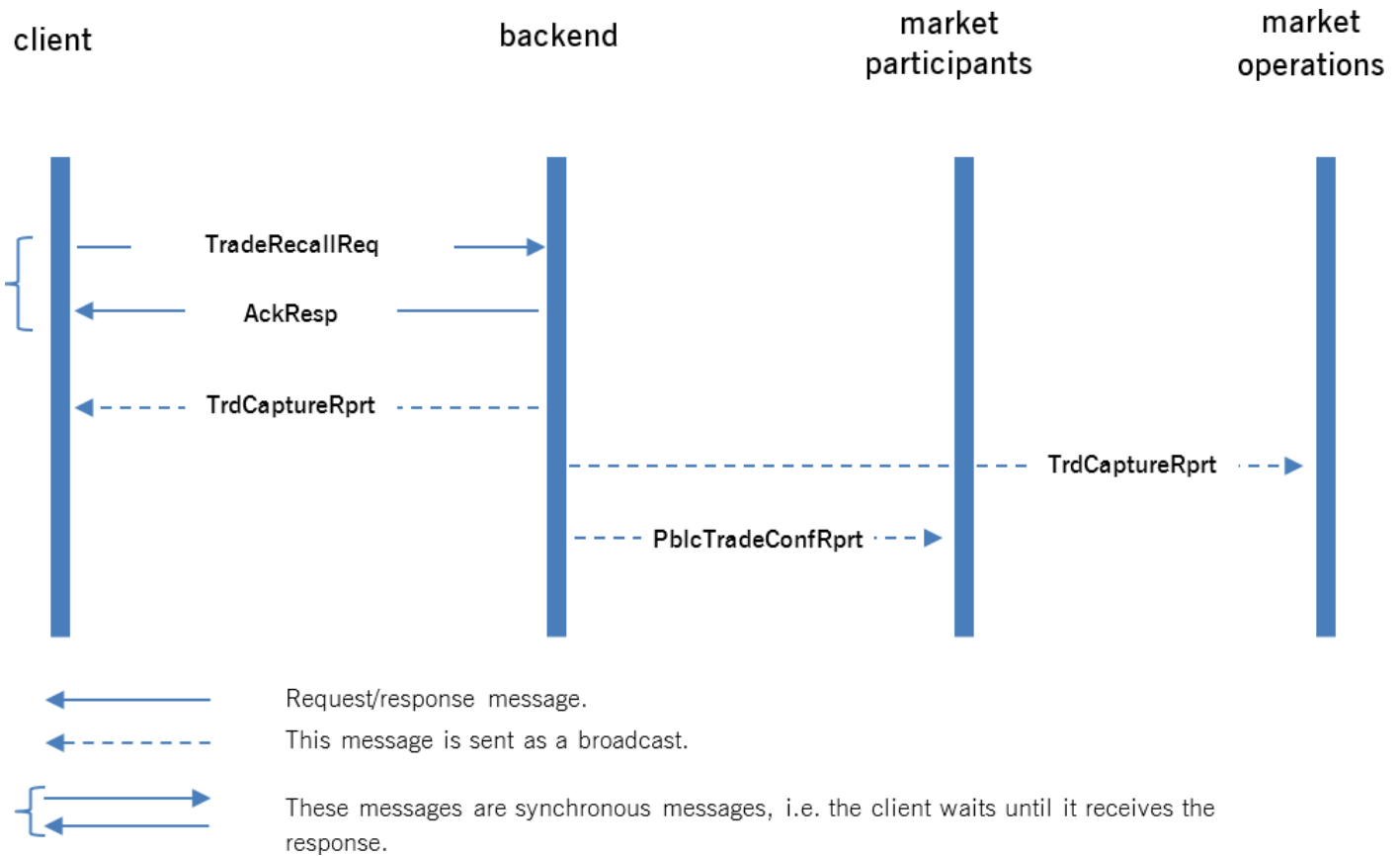
6.3 Trade Maintenance

6.3.1 TradeRecallReq

- **Type:** Management Request
- **Routing Keys:** `m7.request.management`
- **Roles:** Trader, Market operation

This message is used to request a recall of a mistrade. Market Operation will be informed about the trade recall request after successful submission.

The message flow is shown in the below figure:



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

Note: if the trade for which the 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	1	Structure	
	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.
StandardHeader		SE	m	1	Structure	Standard header of each message.
	marketId	A	m		Char(255)	Market Identification Code (MIC) of the market to which the request is sent or from which the request originates.
	onBehalfUserId	A	o		Integer	UserId of the target user in case of On-behalf trading (see On-behalf Trading chapter in <i>DFS180</i>).
clientData		SE	o	0..1	Structure	For order management transactions the client data fields in this section can be used by the client to store information or meta-data about a request. None of this information is used by the backend and it is returned to the client with the response.
	clientDataInt	A	o		Integer	Integer for client's usage.
	clientDataString	A	o		Char(255)	String for client's usage.

XML Tag	Type	m/o	No.	Data Type	Short description
clientCorrelationId	A	o		Char(255)	Correlation Id for client's usage.

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 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.

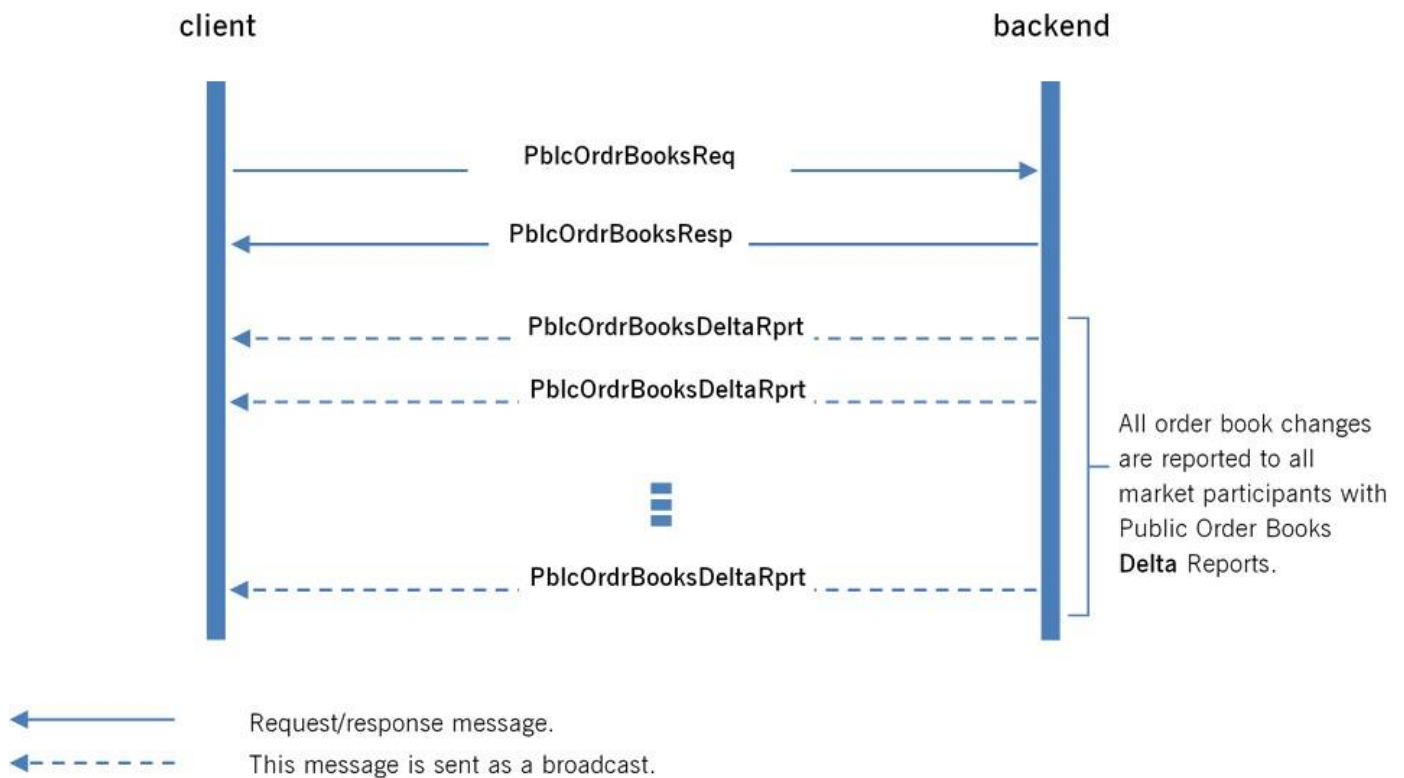
PblcOrdrBooksDeltaRprt messages with an order book revision number smaller than the ones received in the PblcOrdrBooksResp must be ignored.

6.4.2 PblcOrdrBooksReq

- **Type:** Inquiry Request
- **Routing Keys:** `m7.request.inquiry`
- **Roles:** [All]
- **Request Limits:** 14/70

The Public Order Books Request is used to retrieve the local view of the public order books. 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. Note that the purpose of the request is to get an 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 view on the public order books 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.

The message flow is shown in the below figure:



XML Tag	Type	m/o	No.	Data Type	Short description
PblcOrdRBooksReq	SE	m	1	Structure	
contractType	A	o		Char(3)	Defines which kind of contracts should be retrieved. Possible values are: <ul style="list-style-type: none"> 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)	Open/Close indicator value. Valid values: <ul style="list-style-type: none"> O: Open position indicator C: Close position indicator
StandardHeader	SE	m	1	Structure	Standard header of each message.
marketId	A	m		Char(255)	Market Identification Code (MIC) of the market to which the request is sent or from which the request originates.
onBehalfUserId	A	o		Integer	UserID of the target user in case of On-behalf trading (see On-behalf Trading chapter in <i>DFS180</i>).
clientData	SE	o	0..1	Structure	For order management transactions the client data fields in this section can be used by the client to store information or meta-data about a request. None of this information is used by the backend and it is returned to the client with the response.
clientDataInt	A	o		Integer	Integer for client's usage.

XML Tag			Type	m/o	No.	Data Type	Short description
		clientDataString	A	o		Char(255)	String for client's usage.
		clientCorrelationId	A	o		Char(255)	Correlation Id for client's usage.
	prodName		CE	o	0..1000	Char(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.
	contractId		CE	o	0..1	Long	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.
	dlvryAreald		CE	o	0..1000	Char(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.

6.4.3 PbIcOrdRBooksResp

- **Type:** Inquiry Response
- **Response to:** PbIcOrdRBooksReq (sent to private response queue).
- **Broadcast:** No
- **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.

Subsequent changes to the public order books as a result of an order entry, modification, deletion or execution 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.

XML Tag			Type	m/o	No.	Data Type	Short description
PbIcOrdRBooksResp			SE	m	1	Structure	
	StandardHeader		SE	m	1	Structure	Standard header of each message.
	marketId		A	m		Char(255)	Market Identification Code (MIC) of the market to which the request is sent or from which the request originates.
	onBehalfUserId		A	o		Integer	UserID of the target user in case of On-behalf trading (see On-behalf Trading chapter in <i>DFS180</i>).
	clientData		SE	o	0..1	Structure	For order management transactions the client data fields in this section can be used by the client to store information or meta-data about a request. None of this information is used by the backend and it is returned to the client with the response.
		clientDataInt	A	o		Integer	Integer for client's usage.

XML Tag				Type	m/o	No.	Data Type	Short description
			clientDataString	A	o		Char(255)	String for client's usage.
			clientCorrelationId	A	o		Char(255)	Correlation Id for client's usage.
			OrdrbookList	SE	o	0..1	Structure	
			OrdrBook	SE	o	0..n	Structure	
			contractId	A	m		Long	The contract Id of the underlying contract.
			dlvryAreaId	A	m		Char(16)	Delivery Area to which the attached order books refer to.
			lastPx	A	o		Long	Last traded price
			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: <ul style="list-style-type: none"> -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. Starts from 1 (first empty order book after contract generation). 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 beginning.
			highPx	A	o		Long	The highest traded price since the start of the trading period.
			lowPx	A	o		Long	The lowest traded price since the start of the trading period.
			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.
			SellOrdrList	SE	o	0..1	Structure	
			OrdrBookEntry	SE	o	0..n	Structure	
			ordrId	A	m		Long	The Order Id as determined by the backend system. <ul style="list-style-type: none">

XML Tag					Type	m/o	No.	Data Type	Short description
				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.
				ordrEntryTime	A	m		DateTime	The timestamp of the order.
				ordrExeRestriction	A	o		Char(3)	Execution restriction of the order. This attribute is set only in the case of AON orders (value = AON). For more information and possible values see the Order Entry message.
				ordrType	A	o		Char(1)	The following values can be used: <ul style="list-style-type: none"> ● 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: <ul style="list-style-type: none"> ● O: Open position indicator ● C: Close position indicator
				ClgHse	SE	o	0..n	Structure	
				clgHseCode	A	m		Char(255)	Clearing House Code.
				BuyOrdrList	SE	o	0..1	Structure	
				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: <ul style="list-style-type: none"> ● 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.
				ordrEntryTime	A	m		DateTime	The timestamp of the order.
				ordrExeRestriction	A	o		Char(3)	Execution restriction of the order. This attribute is set only in the case of AON orders (value = AON). For more information and possible values see the Order Entry message.

XML Tag					Type	m/o	No.	Data Type	Short description
				ordrType	A	o		Char(1)	The following values can be used: <ul style="list-style-type: none"> • 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: <ul style="list-style-type: none"> • O: Open position indicator • C: Close position indicator
				ClgHse	SE	o	0..n	Structure	
				clgHseCode	A	m		Char(255)	Clearing House Code.

6.4.4 PbIcOrdrBooksDeltaRprt

- **Type**: Broadcast
- **Response to**: n/a
- **Broadcast**: Yes
- **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.

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.⁸

The behavior 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 contract expires, Public Order Books Delta Reports for these delivery areas with a quantity of 0 will not be re-distributed.

The message layout is shared with the Public Order Books Response.

XML Tag				Type	m/o	No.	Data Type	Short description
PbIcOrdrBooksDeltaRprt				SE	m	1	Structure	
	StandardHeader			SE	m	1	Structure	Standard header of each message.
		marketId		A	m		Char(255)	Market Identification Code (MIC) of the market to which the request is sent or from which the request originates.
		onBehalfUserId		A	o		Integer	UserID of the target user in case of On-behalf trading (see On-behalf Trading chapter in <i>DFS180</i>).

XML Tag			Type	m/o	No.	Data Type	Short description
		clientData	SE	o	0..1	Structure	For order management transactions the client data fields in this section can be used by the client to store information or meta-data about a request. None of this information is used by the backend and it is returned to the client with the response.
		clientDataInt	A	o		Integer	Integer for client's usage.
		clientDataString	A	o		Char(255)	String for client's usage.
		clientCorrelationId	A	o		Char(255)	Correlation Id for client's usage.
		OrdrbookList	SE	o	0..1	Structure	
		OrdrBook	SE	o	0..n	Structure	
		contractId	A	m		Long	The contract Id of the underlying contract.
		dlvryAreaId	A	m		Char(16)	Delivery Area to which the attached order books refer to.
		lastPx	A	o		Long	Last traded price
		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: <ul style="list-style-type: none"> • -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. Starts from 1 (first empty order book after contract generation). 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 beginning.
		highPx	A	o		Long	The highest traded price since the start of the trading period.
		lowPx	A	o		Long	The lowest traded price since the start of the trading period.
		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.
		SellOrdrList	SE	o	0..1	Structure	
		OrdrBookEntry	SE	o	0..n	Structure	

XML Tag					Type	m/o	No.	Data Type	Short description
				ordrId	A	m		Long	The Order Id as determined by the backend system.
				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.
				ordrEntryTime	A	m		DateTime	The timestamp of the order.
				ordrExeRestriction	A	o		Char(3)	Execution restriction of the order. This attribute is set only in the case of AON orders (value = AON). For more information and possible values see the Order Entry message.
				ordrType	A	o		Char(1)	The following values can be used: <ul style="list-style-type: none"> • 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: <ul style="list-style-type: none"> • O: Open position indicator • C: Close position indicator
				ClgHse	SE	o	0..n	Structure	
				clgHseCode	A	m		Char(255)	Clearing House Code.
				BuyOrdrList	SE	o	0..1	Structure	
				OrdrBookEntry	SE	o	0..n	Structure	
				ordrId	A	m		Long	The Order Id as determined by the backend system. <ul style="list-style-type: none"> • •
				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.
				ordrEntryTime	A	m		DateTime	The timestamp of the order.

XML Tag					Type	m/o	No.	Data Type	Short description
				ordrExeRestriction	A	o		Char(3)	Execution restriction of the order. This attribute is set only in the case of AON orders (value = AON). For more information and possible values see the Order Entry message.
				ordrType	A	o		Char(1)	The following values can be used: <ul style="list-style-type: none"> 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: <ul style="list-style-type: none"> O: Open position indicator C: Close position indicator
				ClgHse	SE	o	0..n	Structure	
				clgHseCode	A	m		Char(255)	Clearing House Code.

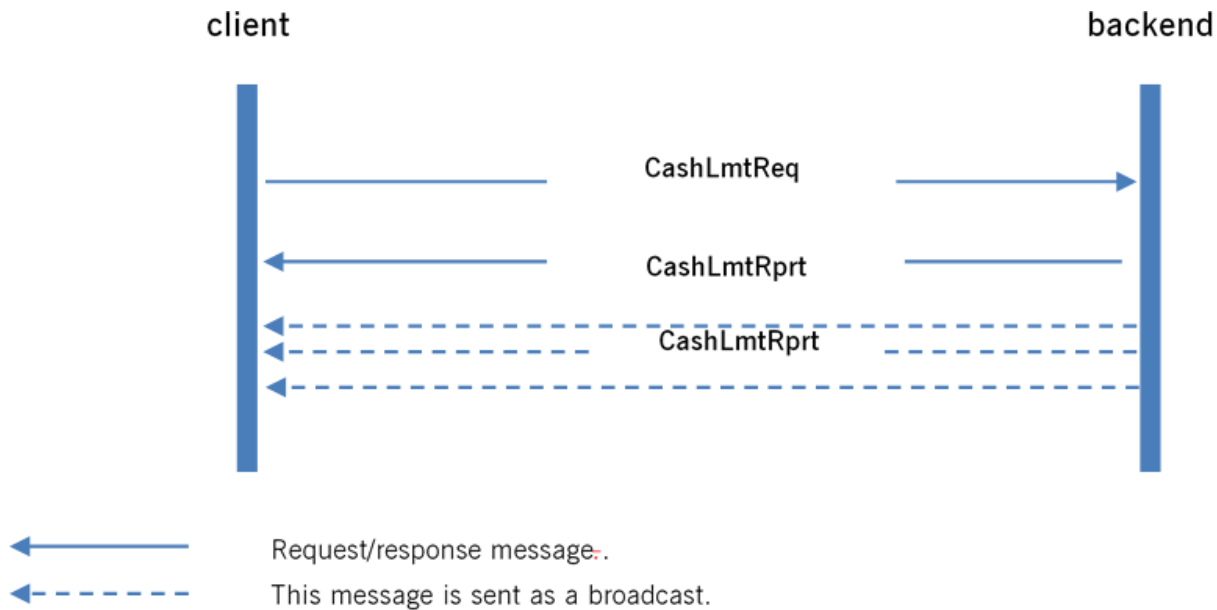
6.4.5 CashLmtReq

- **Type:** Inquiry Request
- **Routing Keys:** `m7.request.inquiry`
- **Roles:** Trader, Market Operation, Broker, Market Maker, Clearing user
- **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 message flow during a user session in respect of the Cash Limit Request and Cash Limit Report messages.



XML Tag	Type	m/o	No.	Data Type	Short description
CashLmtReq	SE	m	1	Structure	
mbrld	A	m		Char(5)	Member Id for which the trading limit is requested.
StandardHeader	SE	m	1	Structure	Standard header of each message.
marketId	A	m		Char(255)	Market Identification Code (MIC) of the market to which the request is sent or from which the request originates.
onBehalfUserId	A	o		Integer	UserID of the target user in case of On-behalf trading (see On-behalf Trading chapter in <i>DFS180</i>).
clientData	SE	o	0..1	Structure	For order management transactions the client data fields in this section can be used by the client to store information or meta-data about a request. None of this information is used by the backend and it is returned to the client with the response.
clientDataInt	A	o		Integer	Integer for client's usage.
clientDataString	A	o		Char(255)	String for client's usage.
clientCorrelationId	A	o		Char(255)	Correlation Id for client's usage.

6.4.6 CashLmtRprt

- **Type:** Inquiry Response; Broadcast
- **Response to:** CashLmtReq (sent to the private response queue see [Request-Response communication](#))
- **Roles:** Trader, Market Operation, Broker, Market Maker, Clearing user
- **Broadcasted:** Trader, Market Operation, Broker, Market Maker, Clearing user
- **Broadcast Routing Keys:** [schema-version].mbr.[mbrId]
- **Broadcast Audience:** All users from a particular member, Admins, Brokers with assigned members, Clearing users

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 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. 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	1	Structure	
	mbrld	A	m		Char(5)	Member Id
StandardHeader		SE	m	1	Structure	Standard header of each message.
	marketId	A	m		Char(255)	Market Identification Code (MIC) of the market to which the request is sent or from which the request originates.
	onBehalfUserId	A	o		Integer	UserID of the target user in case of On-behalf trading (see On-behalf Trading chapter in <i>DFS180</i>).
	clientData	SE	o	0..1	Structure	For order management transactions the client data fields in this section can be used by the client to store information or meta-data about a request. None of this information is used by the backend and it is returned to the client with the response.
	clientDataInt	A	o		Integer	Integer for client's usage.
	clientDataString	A	o		Char(255)	String for client's usage.
	clientCorrelationId	A	o		Char(255)	Correlation Id for client's usage.
CurrentCashLmtList		SE	m	1	Structure	
	CurrentCashLmt	SE	o	0..50	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		Char(3)	The currency of the cash limit (EUR/GBP). The default value is EUR.
	decShift	A	o		Integer	The decimal shift used to determine the cash limit in the currency of the limit. However, the value will always be 2. It means that the value in the cash limit corresponds to the sub unit of the limit currency (currency/100). The attribute is deprecated and might be removed in one of the future versions.
CashLmtList		SE	m	1	Structure	
	CashLmt	SE	o	0..50	Structure	
	revisionNo	A	m		Long	This value is increased every time the current cash limit is changed.
	cashLmt	A	m		Long	The cash limit that is available for the member.

XML Tag			Type	m/o	No.	Data Type	Short description
		decShift	A	o		Integer	The decimal shift used to determine the cash limit in the currency of the limit. However, the value will always be 2. It means that the value in the cash limit corresponds to the sub unit of the limit currency (currency/100). The attribute is deprecated and might be removed in one of the future versions.
		currency	A	o		Char(3)	Currency of the cash limit (EUR/GBP). Default value is EUR.
		lmtId	A	m		Long	The (internal) limit ID, the primary key.
		state	A	m		Char(4)	The status of the limit entry. Valid values: <ul style="list-style-type: none">• ACTI : The limit is active
		startDate	A	o		DateTime	The first date when the limit is active.
		endDate	A	o		DateTime	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.

6.4.7 CashLmtDeltaRprt

- **Type:** Broadcast
- **Response to:** –
- **Roles:** Trader, Market Operation, Broker, Market Maker, Clearing user
- **Broadcast Routing Keys:** [schema-version].mbr.[mbrId]
- **Broadcast Audience:** All users from a particular member, Admins, Brokers with assigned members, Clearing users

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	1	Structure	
	revisionNo	A	m		Long	This value is increased every time the current cash limit is changed.
	cashLmt	A	m		Long	The cash limit that is available for the member.
	decShift	A	o		Integer	The decimal shift used to determine the cash limit in the currency of the limit. However, the value will always be 2. It means that the value in the cash limit corresponds to the sub unit of the limit currency (currency/100). The attribute is deprecated and might be removed in one of the future versions.
	currency	A	o		Char(3)	Currency of the cash limit (EUR/GBP). Default value is EUR.
	mbrId	A	m		Char(5)	Member ID for whom the limit is set.
StandardHeader		SE	m	1	Structure	Standard header of each message.

XML Tag		Type	m/o	No.	Data Type	Short description
	marketId	A	m		Char(255)	Market Identification Code (MIC) of the market to which the request is sent or from which the request originates.
	onBehalfUserId	A	o		Integer	UserID of the target user in case of On-behalf trading (see On-behalf Trading chapter in <i>DFS180</i>).
	clientData	SE	o	0..1	Structure	For order management transactions the client data fields in this section can be used by the client to store information or meta-data about a request. None of this information is used by the backend and it is returned to the client with the response.
	clientDataInt	A	o		Integer	Integer for client's usage.
	clientDataString	A	o		Char(255)	String for client's usage.
	clientCorrelationId	A	o		Char(255)	Correlation Id for client's usage.

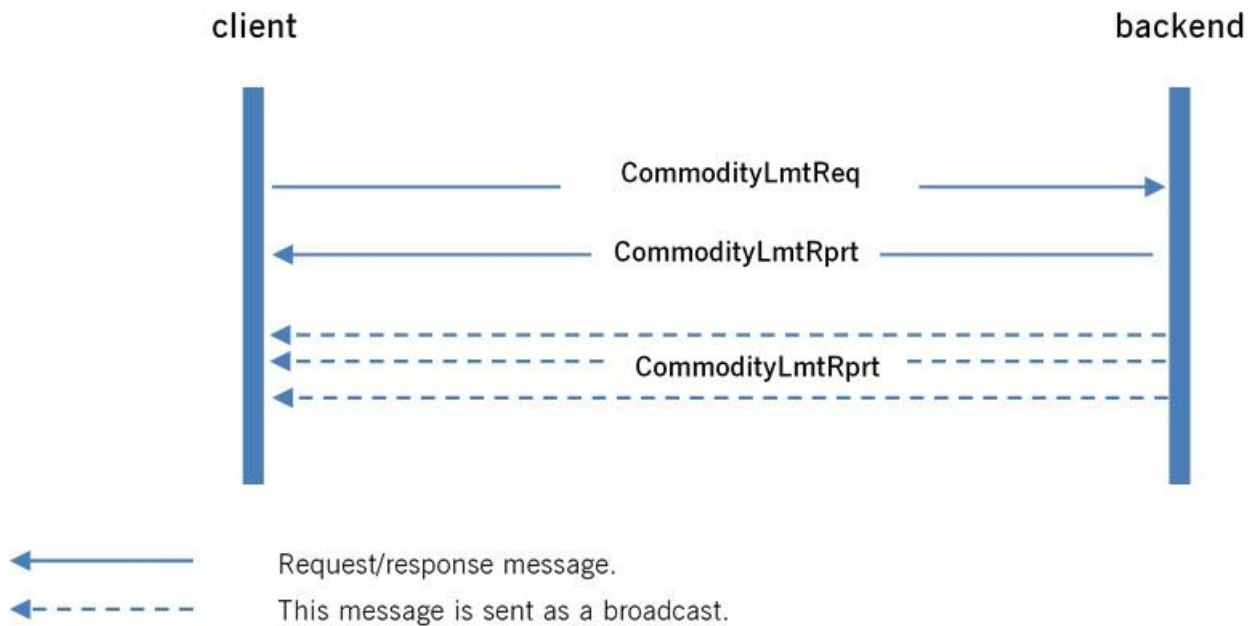
6.4.8 CommodityLmtReq

- **Type:** Inquiry Request
- **Routing Keys:** `m7.request.inquiry`
- **Roles:** Trader, Market Operation, Broker, Market Maker
- **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. Please note that commodity limits are not relevant for intraday trading. 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 only required parameter of the request is Member Id.

The typical message flow is shown in the below figure:



XML Tag	Type	m/o	No.	Data Type	Short description
CommodityLmtReq	SE	m	1	Structure	
mbrld	A	m		Char(5)	Member Id for which the commodity limit is requested.
StandardHeader	SE	m	1	Structure	Standard header of each message.
marketId	A	m		Char(255)	Market Identification Code (MIC) of the market to which the request is sent or from which the request originates.
onBehalfUserId	A	o		Integer	UserID of the target user in case of On-behalf trading (see On-behalf Trading chapter in <i>DFS180</i>).
clientData	SE	o	0..1	Structure	For order management transactions the client data fields in this section can be used by the client to store information or meta-data about a request. None of this information is used by the backend and it is returned to the client with the response.
clientDataInt	A	o		Integer	Integer for client's usage.
clientDataString	A	o		Char(255)	String for client's usage.
clientCorrelationId	A	o		Char(255)	Correlation Id for client's usage.

6.4.9 CommodityLmtRprt

- **Type:** Inquiry Response; Broadcast
- **Response to:** CommodityLmtReq (sent to the private response queue)
- **Roles:** Trader, Market Operation, Broker, Market Maker
- **Broadcasted:** Yes
- **Broadcast Routing Keys:** [schema-version].mbr.[mbr.Id]
- **Broadcast Audience:** All users from a particular member, Admins and Brokers with assigned members

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).

When the Commodity Limit Report is sent as a response it is sent to the private response queue of the requesting user. 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	1	Structure	Standard header of each message.
		marketId	A	m		Char(255)	Market Identification Code (MIC) of the market to which the request is sent or from which the request originates.
		onBehalfUserId	A	o		Integer	UserID of the target user in case of On-behalf trading (see On-behalf Trading chapter in <i>DFS180</i>).
	clientData		SE	o	0..1	Structure	For order management transactions the client data fields in this section can be used by the client to store information or meta-data about a request. None of this information is used by the backend and it is returned to the client with the response.
		clientDataInt	A	o		Integer	Integer for client's usage.
		clientDataString	A	o		Char(255)	String for client's usage.
		clientCorrelationId	A	o		Char(255)	Correlation Id for client's usage.
	CommodityLmtList		SE	o	0..1	Structure	
	CommodityLmt		SE	o	0..n	Structure	
		commodityLmt	A	m		Integer	The commodity limit available for the member in the specified contract after the change is applied. This value will be used as the default commodity limit for the member as from the application of the change.
		mbrId	A	m		Char(5)	The member Id. Unique identifier of the member.
		contractId	A	m		Long	The contract Id of the contract to which the commodity limit applies.
		revisionNo	A	m		Long	This value is increased every time the commodity limit is changed and a new Commodity Limit Response is issued.

6.4.10 MsgReq

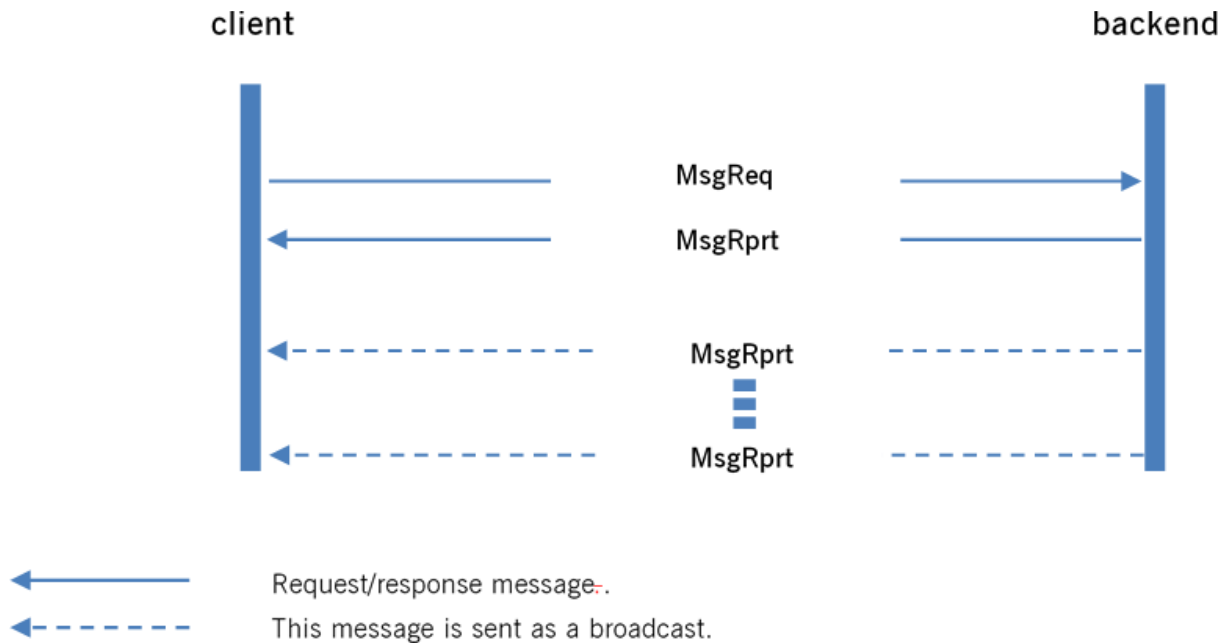
- **Type:** Inquiry Request
- **Routing Keys:** `m7.request.inquiry`
- **Roles:** All
- **Request Limits:** 14/70

This inquiry message is used to retrieve the public and/or private messages issued by the back-end 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 back-end.

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.

The message flow is shown in the below figure:



XML Tag	Type	m/o	No.	Data Type	Short description
MsgReq	SE	m	1	Structure	
acctId	A	o		Char(32)	The account Id (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.
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.
endDate	A	m		DateTime	The timestamp defining the point in time that the messages should be retrieved.
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: <ul style="list-style-type: none"> ALL: Return all messages. PUBLIC: Return only public messages. PRIVATE: Return only private messages.
StandardHeader	SE	m	1	Structure	Standard header of each message.
marketId	A	m		Char(255)	Market Identification Code (MIC) of the market to which the request is sent or from which the request originates.
onBehalfUserId	A	o		Integer	UserId of the target user in case of On-behalf trading (see On-behalf Trading chapter in <i>DFS180</i>).
clientData	SE	o	0..1	Structure	For order management transactions the client data fields in this section can be used by the client to store information or meta-data about a request. None of this information is used by the backend and it is returned to the client with the response.

XML Tag	Type	m/o	No.	Data Type	Short description
clientDataInt	A	o		Integer	Integer for client's usage.
clientDataString	A	o		Char(255)	String for client's usage.
clientCorrelationId	A	o		Char(255)	Correlation Id for client's usage.

6.4.11 MsgRprt

- **Type:** Inquiry Response; Broadcast
- **Response to:** MsgReq (sent to private response queue)
- **Roles:** All
- **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:** Depends on the particular message type

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.

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		Boolean	If set to true, the MsgList has been truncated to the maximum number of messages (specified in the configuration).
StandardHeader	SE	m	1	Structure	Standard header of each message.
marketId	A	m		Char(255)	Market Identification Code (MIC) of the market to which the request is sent or from which the request originates.
onBehalfUserId	A	o		Integer	UserID of the target user in case of On-behalf trading (see On-behalf Trading chapter in <i>DFS180</i>).
clientData	SE	o	0..1	Structure	For order management transactions the client data fields in this section can be used by the client to store information or meta-data about a request. None of this information is used by the backend and it is returned to the client with the response.
clientDataInt	A	o		Integer	Integer for client's usage.
clientDataString	A	o		Char(255)	String for client's usage.
clientCorrelationId	A	o		Char(255)	Correlation Id for client's usage.

XML Tag			Type	m/o	No.	Data Type	Short description
MsgList			SE	o	0..1	Structure	
Msg			SE	o	0..n	Structure	
		msgld	A	m		Long	The message Id assigned by the back-end system.
		type	A	m		Char(7)	Defines the message type. Valid values: <ul style="list-style-type: none"> 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		Char(255)	Underlying product
		acctld	A	o		Char(32)	Unique identifier of an account
		timestmp	A	m		DateTime	The timestamp of the message as assigned by the back-end system.
		svrty	A	m		Char(3)	Severity of the message: <ul style="list-style-type: none"> URG: Urgent message. ERR: Error. HIG: High priority message. MED: Medium priority message. LOW: Low priority message.
		txt	A	m		Char(300)	Message text for status messages.
		sellDlrvyAreald	A	o		Char(16)	In case of an order execution, this field contains the delivery area of the sell side.
		buyDlrvyAreald	A	o		Char(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	0..1	Structure	List of variables used in message txt attribute.
		Var	SE	o	0..n	Structure	Structure containing information on variables
		id	A	m		Integer	In Error Response and DeleteQuotesResp, 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		Char(255)	Value of the variable (e.g. Acct1)

6.4.12 TradeCaptureReq

- **Type**: Inquiry Request
- **Routing Keys**: `m7.request.inquiry`
- **Roles**: Trader, Market Operation, Broker, Market Maker, Sales
- **Request Limits**: 56/280⁹

This message is used to retrieve trades created during the last x days, where x corresponds to the value of the attribute `contractStoreTimeInDays` in the `SystemInfoResp`:

- For *Traders*, a set of valid accounts must be specified and all private trades are returned. If no account is specified, empty `TradeCaptureRprt` is returned. In case the request is sent with an invalid or non-existent account ID, an error response 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.

XML Tag		Type	m/o	No.	Data Type	Short description
TradeCaptureReq		SE	m	1	Structure	
	startDate	A	m		DateTime	The start of the period for which the trades are retrieved. This value must fulfil the following conditions: endDate-startDate <= the number of hours configured for the client. The default configuration is 7 hours.
	endDate	A	o		DateTime	The end of the period for which the trades are retrieved. The following condition must be fulfilled: endDate-startDate <= the number of hours configured for the client. The default configuration is 7 hours.
	dateMeaning	A	o		Char(11)	This defines the meaning of the startDate and endDate parameters. Valid values: <ul style="list-style-type: none"> • 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
StandardHeader		SE	m	1	Structure	Standard header of each message.
	marketId	A	m		Char(255)	Market Identification Code (MIC) of the market to which the request is sent or from which the request originates.
	onBehalfUserId	A	o		Integer	UserID of the target user in case of On-behalf trading (see On-behalf Trading chapter in <i>DFS180</i>).
clientData		SE	o	0..1	Structure	For order management transactions the client data fields in this section can be used by the client to store information or meta-data about a request. None of this information is used by the backend and it is returned to the client with the response.
	clientDataInt	A	o		Integer	Integer for client's usage.
	clientDataString	A	o		Char(255)	String for client's usage.
	clientCorrelationId	A	o		Char(255)	Correlation Id for client's usage.
	acctId	CE	o	0..n	Char(32)	The selected account. For Market Operation and Sales users this attribute is optional and ignored by the backend.

6.4.13 TradeCaptureRprt

- **Type:** Management Response, Broadcast
- **Response to:** TradeCaptureReq (sent to private response queue)
- **Broadcasted:** Yes
- **Routing Keys:** `[schema-version].hlftd.[acctId]` , `[schema-version].trd.[mktAreaId]`
- **Broadcast audience:** **Half trade:** Trader (owner of the order) and other traders from his Balancing groups, Broker with assignment to 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].hlftd.[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].trd.[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).

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	1	Structure	Standard header of each message.
	marketId		A	m		Char(255)	Market Identification Code (MIC) of the market to which the request is sent or from which the request originates.
	onBehalfUserId		A	o		Integer	UserID of the target user in case of On-behalf trading (see On-behalf Trading chapter in <i>DFS180</i>).
	clientData		SE	o	0..1	Structure	For order management transactions the client data fields in this section can be used by the client to store information or meta-data about a request. None of this information is used by the backend and it is returned to the client with the response.

XML Tag			Type	m/o	No.	Data Type	Short description
		clientDataInt	A	o		Integer	Integer for client's usage.
		clientDataString	A	o		Char(255)	String for client's usage.
		clientCorrelationId	A	o		Char(255)	Correlation Id for client's usage.
		TradeList	SE	o	0..1	Structure	
		Trade	SE	o	0..n	Structure	
		tradeId	A	m		Long	The trade ID of the trade.
		state	A	m		Char(4)	The current state of the trade. Valid values: <ul style="list-style-type: none"> CNCL: The trade was cancelled by market operations. RGRA: The requested recall was granted by market operations. ACTI: The trade is active (this is the default value). RREQ: The recall of this trade was requested. Not applicable to TradeStlmntRprt. RREJ: The requested Recall was rejected by market operations. Not applicable to TradeStlmntRprt.
		contractId	A	m		Long	The Id of the underlying contract.
		qty	A	m		Integer	The executed quantity.
		px	A	m		Long	The execution price in Eurocents (1 Euro = 100).
		execTime	A	m		DateTime	The execution date as assigned by the backend.
		revisionNo	A	m		Long	The revision number of the trade. With every change of the trade the revision number is increased by one.
		preArranged	A	m		Boolean	This defines if the trade was pre-arranged. Valid values: <p>false: Not pre-arranged trade</p> <p>true: Pre-arranged trade</p>
		prearrangeType	A	o		Char(3)	The type of pre-arranged trade. <p>OTC: over the counter trade</p> <p>OPT: on exchange prearranged trade</p> <p>PNC: private and confidential trade</p>
		recallReqTime	A	o		DateTime	The date and time of a recall request.
		recallGrantedTime	A	o		DateTime	The date and time when market operations granted the recall. This is also used when a trade has been cancelled.

XML Tag				Type	m/o	No.	Data Type	Short description
			recallRejectedTime	A	o		DateTime	The 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.
			recallRequestor	A	o		Char(5)	The Member Id of the party who initiated the trade recall. The information is provided to the balancing group of the recall requesting party in case the recall was requested by a member known to M7.
			contractPhase	A	m		Char(4)	Specifies the contract phase: <ul style="list-style-type: none"> • CONT: Continuous Trading • BALA: Balancing Phase • AUCT: Auction Phase • CLSD: Closed Phase. Trading is not possible during this phase. • SDAT: Same Delivery Area Trading Phase
			clgHseCode	A	o		Char(255)	The clearing House code of the Trade
			parentTradeId	A	o		Long	This is filled in the event of cross-contract trades it points to the parent trade id. Note: this parent trade is not sent to the client.
			decomposed	A	o		Boolean	This indicates whether the trade was decomposed to trades in underlying or parent products.
			remoteTradeId	A	o		Long	The remote Trade Id as defined by remote backend system (ie. 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	0..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).
			dlvryAreald	A	o		Char(16)	The delivery Area to which the attached order books refer to.
			acctId	A	o		Char(32)	The account for which the order is entered.
			ordrId	A	o		Long	The Order Id of the order.
			txt	A	o		Char(250)	The text of the order.
			aggressorIndicator	A	o		Char(1)	Indicates whether the executed order was a trade aggressor or trade originator. Valid values: <ul style="list-style-type: none"> • Y: Trade aggressor • N: Trade originator • U: Unknown, for executed orders of remote products and data before migration. Default value.
			usrCode	A	o		Char(6)	The User Code of the user who entered the order.

XML Tag				Type	m/o	No.	Data Type	Short description
			clOrdId	A	o		Char(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		Char(5)	The MemberID of the member who entered the order.
			openCloseInd	A	o		Char(1)	This is mandatory for Futures and Cross product spreads. For other Commodities it is not used. Valid values: <ul style="list-style-type: none"> • O: Open position indicator • C: Close position indicator
			brokerUserId	A	o		Integer	UserID of the broker user
			clgAcctId	A	o		Integer	Clearing account Id
			remoteOrdId	A	o		Long	Order Id as returned by remote backend system (not relevant for SEMOpX)
			Sell	SE	o	0..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).
			dlvryAreald	A	o		Char(16)	The delivery Area to which the attached order books refer to.
			acctId	A	o		Char(32)	The account for which the order is entered.
			ordId	A	o		Long	The Order Id of the order.
			txt	A	o		Char(250)	The text of the order.
			aggressorIndicator	A	o		Char(1)	Indicates whether the executed order was a trade aggressor or trade originator. Valid values: <ul style="list-style-type: none"> • Y: Trade aggressor • N: Trade originator • U: Unknown, for executed orders of remote products and data before migration. Default value.
			usrCode	A	o		Char(6)	The User Code of the user who entered the order.
			clOrdId	A	o		Char(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		Char(5)	The MemberID of the member who entered the order.
			openCloseInd	A	o		Char(1)	This is mandatory for Futures and Cross product spreads. For other Commodities it is not used. Valid values: <ul style="list-style-type: none"> • O: Open position indicator • C: Close position indicator
			brokerUserId	A	o		Integer	UserID of the broker user
			clgAcctId	A	o		Integer	Clearing account Id
			remoteOrdId	A	o		Long	Order Id as returned by remote backend system (not relevant for SEMOpX)

6.4.14 PblcTradeConfReq

- **Type:** Inquiry Request
- **Routing Keys:** `m7.request.inquiry`
- **Roles:** Trader, Broker, Data Vendor
- **Request Limits:** 56/280¹⁰

This message is used to retrieve a list of public trades executed during a specified period. This period can only be *x* days in the past, where *x* corresponds to the value of the attribute `contractStoreTimeInDays` in the `SystemInfoResp`.

XML Tag		Type	m/o	No.	Data Type	Short description
PblcTradeConfReq		SE	m	1	Structure	
	startDate	A	m		DateTime	The start of the period for which the trades are retrieved. This value must fulfil the following conditions: (endDate – startDate) <= the number of hours configured for the client. The default configuration is 7 hours.
	endDate	A	m		DateTime	The end of the period for which the trades are retrieved. The following condition must be fulfilled: (endDate – startDate) <= the number of hours configured for the client. The default configuration is 7 hours.
StandardHeader		SE	m	1	Structure	Standard header of each message.
	marketId	A	m		Char(255)	Market Identification Code (MIC) of the market to which the request is sent or from which the request originates.
	onBehalfUserId	A	o		Integer	UserID of the target user in case of On-behalf trading (see On-behalf Trading chapter in <i>DFS180</i>).
clientData		SE	o	0..1	Structure	For order management transactions the client data fields in this section can be used by the client to store information or meta-data about a request. None of this information is used by the backend and it is returned to the client with the response.
	clientDataInt	A	o		Integer	Integer for client's usage.
	clientDataString	A	o		Char(255)	String for client's usage.
	clientCorrelationId	A	o		Char(255)	Correlation Id for client's usage.
	prodName	CE	o	0..1000	Char(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. If a non-existent prodName is entered, an error response is returned.

6.4.15 PblcTradeConfRprt

- **Type:** Inquiry Response, Broadcast
- **Response to:** PblcTradeConfReq (sent to private response queue)
- **Broadcasted:** Yes
- **Routing Keys:** `[schema-version].pblc.trd.[prodName]`
- **Broadcast Audience:** All traders, brokers and data vendors with 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 the exchange is on the 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.

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
PbblTradeConfRprt			SE	m	1	Structure	
	StandardHeader		SE	m	1	Structure	Standard header of each message.
	marketId		A	m		Char(255)	Market Identification Code (MIC) of the market to which the request is sent or from which the request originates.
	onBehalfUserId		A	o		Integer	UserID of the target user in case of On-behalf trading (see On-behalf Trading chapter in <i>DFS180</i>).
	clientData		SE	o	0..1	Structure	For order management transactions the client data fields in this section can be used by the client to store information or meta-data about a request. None of this information is used by the backend and it is returned to the client with the response.
		clientDataInt	A	o		Integer	Integer for client's usage.
		clientDataString	A	o		Char(255)	String for client's usage.
		clientCorrelationId	A	o		Char(255)	Correlation Id for client's usage.
	TradeList		SE	o	0..1	Structure	
	PbblTradeConf		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: <ul style="list-style-type: none"> • CNCL: The trade 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 recall of this trade was requested. • ACTI: The trade 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).
		contractId	A	m		Long	The contract Id of the traded contract.
		qty	A	m		Integer	The traded quantity.

XML Tag			Type	m/o	No.	Data Type	Short description
		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.
		buyDivryAreald	A	o		Char(16)	The delivery area of the buy side.
		sellDivryAreald	A	o		Char(16)	The delivery area of the sell side.
		clgHseCode	A	o		Char(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.

6.4.16 ContractInfoReq

- **Type:** Inquiry Request
- **Routing Keys:** `m7.request.inquiry`
- **Roles:** All
- **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	m	1	Structure	
		startDate	A	o		DateTime	Start date for requested contract mandatory if contractId is not specified.
		endDate	A	o		DateTime	End date for requested contract mandatory if contractId is not specified.
StandardHeader			SE	m	1	Structure	Standard header of each message.
		marketId	A	m		Char(255)	Market Identification Code (MIC) of the market to which the request is sent or from which the request originates.
		onBehalfUserId	A	o		Integer	UserID of the target user in case of On-behalf trading (see On-behalf Trading chapter in <i>DFS180</i>).
clientData			SE	o	0..1	Structure	For order management transactions the client data fields in this section can be used by the client to store information or meta-data about a request. None of this information is used by the backend and it is returned to the client with the response.
		clientDataInt	A	o		Integer	Integer for client's usage.

XML Tag			Type	m/o	No.	Data Type	Short description
		clientDataString	A	o		Char(255)	String for client's usage.
		clientCorrelationId	A	o		Char(255)	Correlation Id for client's usage.
	prodName		CE	o	0..1000	Char(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.

6.4.17 ContractInfoRprt

- **Type:** Inquiry Response; Broadcast
- **Response to:** ContractInfoReq (sent to the private response queue)
- **Roles:** All
- **Broadcasted:** Yes
- **Broadcast Routing Keys:** `[schema-version].prd.[prodName]`
- **Broadcast Audience:** All users with an assignment to a particular product.

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

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	1	Structure	Standard header of each message.
	marketId		A	m		Char(255)	Market Identification Code (MIC) of the market to which the request is sent or from which the request originates.
	onBehalfUserId		A	o		Integer	UserID of the target user in case of On-behalf trading (see On-behalf Trading chapter in <i>DFS180</i>).
	clientData		SE	o	0..1	Structure	For order management transactions the client data fields in this section can be used by the client to store information or meta-data about a request. None of this information is used by the backend and it is returned to the client with the response.
	clientDataInt		A	o		Integer	Integer for client's usage.
	clientDataString		A	o		Char(255)	String for client's usage.
	clientCorrelationId		A	o		Char(255)	Correlation Id for client's usage.
	ContractList		SE	o	0..1	Structure	
	Contract		SE	o	0..n	Structure	

XML Tag			Type	m/o	No.	Data Type	Short description
		contractId	A	m		Long	The contract Id as defined by the backend system.
		prod	A	m		Char(255)	The underlying product.
		name	A	m		Char(255)	The contract name. This is used for display purposes.
		longName	A	m		Char(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)	<p>The current state of the contract / contract-delivery area combination. Valid values:</p> <ul style="list-style-type: none"> • ACTI: Active: the contract is active and available for trading. • STBY: Stand by: The contract is waiting on external event to become available for trading. For a contract-delivery area combination, the triggering events are: <ul style="list-style-type: none"> ◦ 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 <i>DFS160a</i>. • HIBE: Hibernate: The contract was manually deactivated by market operations. • IACT: Inactive: The contract is inactive and not available for trading.
		tradingPhase	A	m		Char(4)	<p>Specifies the contract phase:</p> <ul style="list-style-type: none"> • CONT: Continuous Trading • BALA: Balancing Phase • AUCT: Auction Phase • CLSD: Closed Phase. Trading is not possible during this 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		DateTime	The parameter gives the contract activation point (e.g. delivery start).
		expPoint	A	o		DateTime	The parameter gives the contract expiry point (e.g. delivery end).
		strikePx	A	o		Long	In case that the bespoke contract is an option, this field contains the strike price of the contract.

XML Tag				Type	m/o	No.	Data Type	Short description
			CallPut	A	o		Char(1)	In the event that the bespoke contract is an option, this field contains a contract type. Valid values: <ul style="list-style-type: none"> • C: Call • P: Put
			optionExpPoint	A	o		DateTime	In case that the bespoke contract is option, this parameter gives an option expiration point.
			bespoke	A	o		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 (not relevant for SEMOpX)
			DivvyAreaState	SE	o	0..n	Structure	The type defines the state of a contract for a specific delivery area.
			divvyAreald	A	m		Char(16)	Delivery or Market Area Id. In the power market this corresponds to the EIC code. This value must be unique.
			state	A	o		Char(4)	The current state of the contract / contract-delivery area combination. Valid values: <ul style="list-style-type: none"> • ACTI: Active: the contract is active and available for trading. • STBY: Stand by: The contract is waiting on external event to become available for trading. For a contract-delivery area combination, the triggering events are: <ul style="list-style-type: none"> • The end of a trading phase of a global product (i.e. ContractInfoRpt 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 <i>DFS160a</i>. • HIBE: Hibernate: The contract was manually deactivated by market operations. • IACT: Inactive: The contract is inactive and not available for trading.
			tradingPhase	A	o		Char(4)	Specifies the contract phase: <ul style="list-style-type: none"> • CONT: Continuous Trading • BALA: Balancing Phase • AUCT: Auction Phase • CLSD: Closed Phase. Trading is not possible during this phase. • SDAT: Same Delivery Area Trading Phase
			tradingPhaseStart	A	m		DateTime	The start date and time of the current/next trading phase in the delivery area.

XML Tag				Type	m/o	No.	Data Type	Short description
			tradingPhaseEnd	A	o		DateTime	The end date and time of the current/next trading phase in the delivery area.
			UndrIngContracts	SE	o	0..1	Structure	In case of bespoke contracts, this structure is used to define the underlying contracts of the bespoke product.
			contractId	CE	m	1..n	Long	The contract Id of the underlying contract leg as defined by the backend system.
			SpreadContracts	SE	o	0..1	Structure	
			leg1ContractId	CE	m	1	Long	The contract Id of the first underlying contract for Auto Combination contracts
			leg2ContractId	CE	m	1	Long	The contract Id of the second underlying contract for Auto Combination contracts

6.4.18 ProdInfoReq

- **Type:** Inquiry Request
- **Routing Keys:** `m7.request.inquiry`
- **Roles:** [All]
- **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	1	Structure	Standard header of each message.
			marketId	A	m		Char(255)	Market Identification Code (MIC) of the market to which the request is sent or from which the request originates.
			onBehalfUserId	A	o		Integer	UserID of the target user in case of On-behalf trading (see On-behalf Trading chapter in <i>DFS180</i>).
			clientData	SE	o	0..1	Structure	For order management transactions the client data fields in this section can be used by the client to store information or meta-data about a request. None of this information is used by the backend and it is returned to the client with the response.
			clientDataInt	A	o		Integer	Integer for client's usage.
			clientDataString	A	o		Char(255)	String for client's usage.
			clientCorrelationId	A	o		Char(255)	Correlation Id for client's usage.
			prodName	CE	o	0..1000	Char(255)	Product name.

6.4.19 ProdInfoRprt

- **Type:** Inquiry Response
- **Response to:** ProdInfoReq (sent to the private response queue)

- **Broadcasted:** Yes
- **Routing Keys:** [schema-version].prd.[prodName]
- **Broadcast Audience:** All users with an assignment to 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 [AcctInfoRprt](#)).

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 [AcctInfoRprt](#)).

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

Key	Value	Prod Type	Value
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	all	Defines the treatment of contracts during the 25 hour DST switch.
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.
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.
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.

Key	Value	Prod Type	Value
intraProductSpreads	true	FUT, CRO	The product supports intra-product spreads.
intraProductSpreadContractCount		FUT, CRO	Defines how many contracts are used for intra-product spreads.
productsWithinDelivery	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).
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	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	true	all	The product group name
aotEnabled	true	Linked only	An automated order transfer for the linked product is enabled.

Prod types:

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

XML Tag	Type	m/o	No.	Data Type	Short description
ProdInfoRprt	SE	m	1	Structure	
StandardHeader	SE	m	1	Structure	Standard header of each message.
marketId	A	m		Char(255)	Market Identification Code (MIC) of the market to which the request is sent or from which the request originates.
onBehalfUserId	A	o		Integer	UserID of the target user in case of On-behalf trading (see On-behalf Trading chapter in <i>DFS180</i>).
clientData	SE	o	0..1	Structure	For order management transactions the client data fields in this section can be used by the client to store information or meta-data about a request. None of this information is used by the backend and it is returned to the client with the response.
clientDataInt	A	o		Integer	Integer for client's usage.
clientDataString	A	o		Char(255)	String for client's usage.
clientCorrelationId	A	o		Char(255)	Correlation Id for client's usage.
ProdList	SE	o	0..1	Structure	

XML Tag			Type	m/o	No.	Data Type	Short description
	Prod		SE	o	0..n	Structure	
	prodType		A	m		Char(3)	Indicates the type of the product. Valid values: <ul style="list-style-type: none"> • ENG: Energy products • COM: Commodities • FUT: Futures • CRO: Cross Product Spreads • UDD: User defined delivery
	prodName		A	m		Char(255)	The unique identifier name of the product.
	dspIName		A	m		Char(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.
	base		A	m		Boolean	Defines if the product is a base product.
	baseReference		A	o		Char(255)	If the product is not a base product, this attribute may contain the reference to the base product.
	groupName		A	o		Char(255)	A group name of the product as set up in the system
	cashLmtEnabled		A	o		Boolean	Indicates whether the cash limits are enabled for the product.
	riskSetId		A	o		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. Valid values: An existing set of parameters.
	currency		A	m		Char(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		Char(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.

XML Tag			Type	m/o	No.	Data Type	Short description
		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 in <i>DFS120</i>).
		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.
		exchangeld	A	m		Char(255)	The exchange Id of the exchange where the orders for this product are matched.
		assetType	A	o		Char(3)	This field is used to define the type of the underlying asset. Valid values: <ul style="list-style-type: none"> • COM: Commodity • STO: Stock • IDX: Index
		assetName	A	o		Char(64)	A Text input field used to define the asset name.
		assetExchangeld	A	o		Char(4)	A text input field used to define the exchange on which the underlying asset is traded.
		executionRestriction	A	m		Char(3)	Defines whether orders referencing the contracts of this product can be partially matched or if only with the full quantity. <ul style="list-style-type: none"> • NON: No restriction (partial matching allowed) • AON: All or nothing
		contractsGenerationNumber	A	m		Integer	Determines 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		Char(3)	The weekday at which the contract is activated. Valid values: <ul style="list-style-type: none"> • MON • TUE • WED • THU • FRI • SAT • SUN • [none]
		contActTime	A	o		DateTime	The time at which the contract is activated.
		contExpiryBusinessDay	A	o		Boolean	Defines if the contract expiry day should be on a business day or on any calendar day.

XML Tag				Type	m/o	No.	Data Type	Short description
			contExpiryDay	A	o		Char(3)	The week day at which the contract expires. Valid values: <ul style="list-style-type: none"> • MON • TUE • WED • THU • FRI • SAT • SUN • [none]
			contExpiryTime	A	o		DateTime	The time at which the contract expires.
			state	A	m		Char(4)	The current state of the product. Valid values: <ul style="list-style-type: none"> • HIBE: Hibernate: The product is hibernated and is not available for trading. • IACT: The product is inactive and not available for trading. • ACTI: The product is active and available for trading.
			timeZone	A	o		Char(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: <ul style="list-style-type: none"> • CET • Europe/London
			masterProd	A	o		Char(255)	Reference to the corresponding master product.
			linkedProd	A	o		Char(255)	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		Char(255)	Exchange specific product attribute name.
			cfgVal	A	m		Char(255)	Exchange specific product attribute value.
			DlvryAreaAssignment	SE	o	0..n	Structure	List of delivery areas assigned to the product.
			dlvryAreald	A	m		Char(16)	Delivery Area Id.
			riskSetId	A	o		Long	Id of the risk set used for cash limit calculation for this product or for this delivery area. The attribute is mandatory when cashLmtEnabled is true; otherwise it is not present. The default selection is the product default risk set from Attributes panel. Valid value: An existing set of parameters.
			schedule	SE	o	0..1	Structure	Trading schedule used for the product or delivery area.
			scheduleId	A	m		Long	Unique identifier for the schedule.
			scheduleName	A	m		Char(255)	Display name for the schedule.

XML Tag					Type	m/o	No.	Data Type	Short description
				tradingDays	A	m		Char(∞)	List of trading days. Valid values: <ul style="list-style-type: none"> • monday • tuesday • wednesday • thursday • friday • saturday • sunday
				tradeOnHolidays	A	m		Boolean	Indicates whether the contracts are tradeable on holidays.
				tradingPhases	SE	m	1..n	Structure	
				phaseName	A	o		Char(255)	Display name for the phase.
				phaseType	A	m		Char(4)	Phase of the contract. Valid values: <ul style="list-style-type: none"> • CONT: Continuous Trading • BALA: Balancing Phase • AUCT: Auction Phase • CLSD: Closed Phase. Trading is not possible during this phase. • SDAT: Same Delivery Area Trading Phase
				referencePoint	A	m		Char(24)	Reference point. Valid values: <ul style="list-style-type: none"> • ABSOLUTE_START • ABSOLUTE_END • ABSOLUTE_YEAR_START • ABSOLUTE_YEAR_END • ABSOLUTE_QUARTER_START • ABSOLUTE_QUARTER_END • RELATIVE_START • RELATIVE_END • FULL_HOUR_RELATIVE_START • FULL_HOUR_RELATIVE_END • FIXED
				offset	A	o		Duration	Defines the offset for the phase (if referencePoint != FIXED).
				fixedTime	A	o		DateTime	Allows to specify a fixed type (if referencePoint == FIXED).
				ContractNamePattern	CE	o	0..1	Char(255)	Regular expression for the contract name.

6.4.20 MktStateReq

- **Type:** Inquiry Request
- **Routing Keys:** `m7.request.inquiry`
- **Roles:** All
- **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	m	1	Structure	Standard header of each message.
	marketId		A	m		Char(255)	Market Identification Code (MIC) of the market to which the request is sent or from which the request originates.
	onBehalfUserId		A	o		Integer	UserID of the target user in case of On-behalf trading (see On-behalf Trading chapter in <i>DFS180</i>).
	clientData		SE	o	0..1	Structure	For order management transactions the client data fields in this section can be used by the client to store information or meta-data about a request. None of this information is used by the backend and it is returned to the client with the response.
		clientDataInt	A	o		Integer	Integer for client's usage.
		clientDataString	A	o		Char(255)	String for client's usage.
		clientCorrelationId	A	o		Char(255)	Correlation Id for client's usage.

6.4.21 MktStateRprt

- **Type:** Inquiry Response; Broadcast
- **Response to:** MktStateReq (sent to the private response queue)
- **Roles:** All
- **Broadcasted:** Yes
- **Broadcast Routing Keys:** `[schema-version].public`
- **Broadcast Audience:** 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.

XML Tag			Type	m/o	No.	Data Type	Short description
MktStateRprt			SE	m	1	Structure	
	revisionNo		A	m		Long	The revision number of the market. With every change of the market state this value is increased by one.
	state		A	m		Char(4)	Contains the current market state. Valid values: <ul style="list-style-type: none"> • HIBE: Hibernated; no trading is possible and order books are empty. • ACTI: Market is active and trading is possible.
	StandardHeader		SE	m	1	Structure	Standard header of each message.
	marketId		A	m		Char(255)	Market Identification Code (MIC) of the market to which the request is sent or from which the request originates.
	onBehalfUserId		A	o		Integer	UserID of the target user in case of On-behalf trading (see On-behalf Trading chapter in <i>DFS180</i>).

XML Tag			Type	m/o	No.	Data Type	Short description
		clientData	SE	o	0..1	Structure	For order management transactions the client data fields in this section can be used by the client to store information or meta-data about a request. None of this information is used by the backend and it is returned to the client with the response.
		clientDataInt	A	o		Integer	Integer for client's usage.
		clientDataString	A	o		Char(255)	String for client's usage.
		clientCorrelationId	A	o		Char(255)	Correlation Id for client's usage.

6.4.22 StlmntProcessInfoReq

- **Type:** Inquiry Request
- **Routing Keys:** `m7.request.inquiry`
- **Roles:** Market Operation, Sales. Access for Traders is configurable.
- **Request Limits:** 56/280¹¹

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.

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	
	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 <= the number of hours configured for the client. The default configuration is 7 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 <= the number of hours configured for the client. The default configuration is 7 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		Char(32)	This specifies the clientAccountId to which the request is applicable.
StandardHeader		SE	m	1	Structure	Standard header of each message.
	marketId	A	m		Char(255)	Market Identification Code (MIC) of the market to which the request is sent or from which the request originates.
	onBehalfUserId	A	o		Integer	UserId of the target user in case of On-behalf trading (see On-behalf Trading chapter in <i>DFS180</i>).
clientData		SE	o	0..1	Structure	For order management transactions the client data fields in this section can be used by the client to store information or meta-data about a request. None of this information is used by the backend and it is returned to the client with the response.
	clientDataInt	A	o		Integer	Integer for client's usage.
	clientDataString	A	o		Char(255)	String for client's usage.
	clientCorrelationId	A	o		Char(255)	Correlation Id for client's usage.

6.4.23 StlmntProcessInfoRprt

- **Type:** Inquiry Response, Broadcast
- **Response to:** StlmntProcessInfoReq (sent to private response queue)
- **Broadcasted:** Yes
- **Routing Keys:** [schema-version].settlement , [schema-version].bg.[acctId]
- **Broadcast audience:** Admins, Sales
Configurable (with routing key [schema-version].bg.[acctId]):
 - Owner of half of the trade and all users from his balancing group.
 - Broker with assignment to 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).

XML Tag		Type	m/o	No.	Data Type	Short description
StlmntProcessInfoRprt		SE	m	1	Structure	
	StandardHeader	SE	m	1	Structure	Standard header of each message.
	marketId	A	m		Char(255)	Market Identification Code (MIC) of the market to which the request is sent or from which the request originates.
	onBehalfUserId	A	o		Integer	UserID of the target user in case of On-behalf trading (see On-behalf Trading chapter in <i>DFS180</i>).
	clientData	SE	o	0..1	Structure	For order management transactions the client data fields in this section can be used by the client to store information or meta-data about a request. None of this information is used by the backend and it is returned to the client with the response.
	clientDataInt	A	o		Integer	Integer for client's usage.
	clientDataString	A	o		Char(255)	String for client's usage.
	clientCorrelationId	A	o		Char(255)	Correlation Id for client's usage.
	Trade	SE	o	0..n	Structure	
	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: <ul style="list-style-type: none"> • INIT: The initial status of a trade before being processed. • 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		Char(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)

6.4.24 RefPxReq

- **Type:** Inquiry Request
- **Routing Keys:** `m7.request.inquiry`
- **Roles:** All
- **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	
	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		Long	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: <ul style="list-style-type: none"> • 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.
StandardHeader		SE	m	1	Structure	Standard header of each message.
	marketId	A	m		Char(255)	Market Identification Code (MIC) of the market to which the request is sent or from which the request originates.
	onBehalfUserId	A	o		Integer	UserID of the target user in case of On-behalf trading (see On-behalf Trading chapter in <i>DFS180</i>).
	clientData	SE	o	0..1	Structure	For order management transactions the client data fields in this section can be used by the client to store information or meta-data about a request. None of this information is used by the backend and it is returned to the client with the response.
	clientDataInt	A	o		Integer	Integer for client's usage.
	clientDataString	A	o		Char(255)	String for client's usage.
	clientCorrelationId	A	o		Char(255)	Correlation Id for client's usage.

6.4.25 RefPxRprt

- **Type**: Inquiry Response, Broadcast
- **Response to**: Reference price request (sent to private response queue)
- **Broadcasted**: Yes

- **Routing Keys:** `[schema-version].prd.[prodName]`
- **Broadcast audience:** All users with assignment to 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	1	Structure	Standard header of each message.
		marketId	A	m		Char(255)	Market Identification Code (MIC) of the market to which the request is sent or from which the request originates.
		onBehalfUserId	A	o		Integer	UserID of the target user in case of On-behalf trading (see On-behalf Trading chapter in <i>DFS180</i>).
		clientData	SE	o	0..1	Structure	For order management transactions the client data fields in this section can be used by the client to store information or meta-data about a request. None of this information is used by the backend and it is returned to the client with the response.
		clientDataInt	A	o		Integer	Integer for client's usage.
		clientDataString	A	o		Char(255)	String for client's usage.
		clientCorrelationId	A	o		Char(255)	Correlation Id for client's usage.
	RefPxList		SE	o	0..1	Structure	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: <ul style="list-style-type: none"> • C: Closing price (the default value) • O: Opening price
		refPx	A	m		Long	The reference price of the contract
		date	A	m		DateTime	The date to which the reference price refers.

6.4.26 ImplOrdReq

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

Note this message is obsolete and may be removed in the next versions.

Implied Order Request is used to inquire for calculated implied orders. If allowed by exchange, the response is ImplOrdResp. Otherwise, ErrResp is returned.

XML Tag			Type	m/o	No.	Data Type	Short description
ImplOrdReq			SE	m	1	Structure	
	StandardHeader		SE	m	1	Structure	Standard header of each message.
	marketId		A	m		Char(255)	Market Identification Code (MIC) of the market to which the request is sent or from which the request originates.
	onBehalfUserId		A	o		Integer	UserID of the target user in case of On-behalf trading (see On-behalf Trading chapter in <i>DFS180</i>).
	clientData		SE	o	0..1	Structure	For order management transactions the client data fields in this section can be used by the client to store information or meta-data about a request. None of this information is used by the backend and it is returned to the client with the response.
		clientDataInt	A	o		Integer	Integer for client's usage.
		clientDataString	A	o		Char(255)	String for client's usage.
		clientCorrelationId	A	o		Char(255)	Correlation Id for client's usage.
	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.

6.4.27 ImplOrdRprt

- **Type:** Inquiry Response; Broadcast
- **Response to:** ImplOrdReq (sent to private response queue)
- **Roles:** Trader, Broker, Market Maker, Market Operator
- **Broadcasted:** Yes
- **Broadcast Routing Keys:** `[schema-version].prddlvr.[prodName].[dlvryAreaId]`
- **Broadcast Audience:** All users with assigned product and delivery area

Note this message is obsolete and may be removed in the next versions.

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 OrdEntry message with listExecInst set to IMPL for the opposite side.

For details about implied orders, please consult *DFS140*.

XML Tag			Type	m/o	No.	Data Type	Short description
ImplOrdRprt			SE	m	1	Structure	
	StandardHeader		SE	m	1	Structure	Standard header of each message.
	marketId		A	m		Char(255)	Market Identification Code (MIC) of the market to which the request is sent or from which the request originates.
	onBehalfUserId		A	o		Integer	UserID of the target user in case of On-behalf trading (see On-behalf Trading chapter in <i>DFS180</i>).

XML Tag				Type	m/o	No.	Data Type	Short description
			clientData	SE	o	0..1	Structure	For order management transactions the client data fields in this section can be used by the client to store information or meta-data about a request. None of this information is used by the backend and it is returned to the client with the response.
			clientDataInt	A	o		Integer	Integer for client's usage.
			clientDataString	A	o		Char(255)	String for client's usage.
			clientCorrelationId	A	o		Char(255)	Correlation Id for client's usage.
			Orders	SE	o	0..1	Structure	List of implied orders
			orders	SE	o	0..n	Structure	Implied order element
			impliedOrderId	A	m		Char(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 on which side of the market the order is entered or displayed. Valid values: <ul style="list-style-type: none"> BUY: Buy order. SELL: Sell order.
			qty	A	m		Integer	Contains the total quantity of the order
			price	A	m		Long	The limit price of the implied order.
			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	m		Long	The Id of the contract
			price	A	m		Long	The price of the order

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.

6.5.1 OrdQuoteReq

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

This message is used to submit an order quote request.

XML Tag		Type	m/o	No.	Data Type	Short description
OrdQuoteReq		SE	m	1	Structure	
	ordrId	A	m		Long	Unique identifier for an Order.
StandardHeader		SE	m	1	Structure	Standard header of each message.
	marketId	A	m		Char(255)	Market Identification Code (MIC) of the market to which the request is sent or from which the request originates.
	onBehalfUserId	A	o		Integer	UserID of the target user in case of On-behalf trading (see On-behalf Trading chapter in <i>DFS180</i>).
	clientData	SE	o	0..1	Structure	For order management transactions the client data fields in this section can be used by the client to store information or meta-data about a request. None of this information is used by the backend and it is returned to the client with the response.
	clientDataInt	A	o		Integer	Integer for client's usage.
	clientDataString	A	o		Char(255)	String for client's usage.
	clientCorrelationId	A	o		Char(255)	Correlation Id for client's usage.

6.5.2 OrdQuoteRprt

- **Type:** Broadcast
- **Broadcasted:** Yes
- **Routing Keys:** [schema-version].[prodName]
- **Broadcast audience:** All users with assigned product.
- **Roles:** [All]

A broadcast message about an order quote in the market.

XML Tag		Type	m/o	No.	Data Type	Short description
OrdQuoteRprt		SE	m	1	Structure	
	ordrId	A	m		Long	The unique identifier for the order.
	px	A	m		Long	The limit price of the order.
	contractId	A	m		Long	The underlying contract ID of the order.
	qty	A	m		Integer	The quantity of the order.
	side	A	m		Char(4)	Defines on which side of the market the order is entered or displayed. Valid values: <ul style="list-style-type: none"> • BUY: Buy order. • SELL: Sell order.
	dlvryAreaId	A	m		Char(16)	Delivery or Market Area Id. In the power market this corresponds to the EIC code. This value must be unique.
	timestamp	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.

XML Tag		Type	m/o	No.	Data Type	Short description
StandardHeader		SE	m	1	Structure	Standard header of each message.
	marketId	A	m		Char(255)	Market Identification Code (MIC) of the market to which the request is sent or from which the request originates.
	onBehalfUserId	A	o		Integer	UserID of the target user in case of On-behalf trading (see On-behalf Trading chapter in <i>DFS180</i>).
	clientData	SE	o	0..1	Structure	For order management transactions the client data fields in this section can be used by the client to store information or meta-data about a request. None of this information is used by the backend and it is returned to the client with the response.
	clientDataInt	A	o		Integer	Integer for client's usage.
	clientDataString	A	o		Char(255)	String for client's usage.
	clientCorrelationId	A	o		Char(255)	Correlation Id for client's usage.

6.5.3 OrdQuoteSetupReq

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

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

XML Tag		Type	m/o	No.	Data Type	Short description
OrdQuoteSetupReq		SE	m	1	Structure	
	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	o		Char(255)	The receiving mail address.
	mobileNumber	A	o		Char(255)	The receiving mobile number.
StandardHeader		SE	m	1	Structure	Standard header of each message.
	marketId	A	m		Char(255)	Market Identification Code (MIC) of the market to which the request is sent or from which the request originates.
	onBehalfUserId	A	o		Integer	UserID of the target user in case of On-behalf trading (see On-behalf Trading chapter in <i>DFS180</i>).
	clientData	SE	o	0..1	Structure	For order management transactions the client data fields in this section can be used by the client to store information or meta-data about a request. None of this information is used by the backend and it is returned to the client with the response.
	clientDataInt	A	o		Integer	Integer for client's usage.
	clientDataString	A	o		Char(255)	String for client's usage.

XML Tag	Type	m/o	No.	Data Type	Short description
clientCorrelationId	A	o		Char(255)	Correlation Id for client's usage.

6.5.4 DeleteQuotesReq

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

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	
acctId	A	o		Char(32)	Unique identifier of an account
StandardHeader	SE	m	1	Structure	Standard header of each message.
marketId	A	m		Char(255)	Market Identification Code (MIC) of the market to which the request is sent or from which the request originates.
onBehalfUserId	A	o		Integer	UserID of the target user in case of On-behalf trading (see On-behalf Trading chapter in <i>DFS180</i>).
clientData	SE	o	0..1	Structure	For order management transactions the client data fields in this section can be used by the client to store information or meta-data about a request. None of this information is used by the backend and it is returned to the client with the response.
clientDataInt	A	o		Integer	Integer for client's usage.
clientDataString	A	o		Char(255)	String for client's usage.
clientCorrelationId	A	o		Char(255)	Correlation Id for client's usage.
ProdList	SE	m	1	Structure	List of products for which the quote deletion is requested.
Prod	SE	m	1..n	Structure	
prodName	A	m		Char(255)	The product identifier
action	CE	m	1	Char(∞)	Valid values: <ul style="list-style-type: none"> • 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	m	1	Structure	List of contracts. Mandatory if action ALL is not present.
Contract	SE	m	1..n	Structure	Contract element

XML Tag					Type	m/o	No.	Data Type	Short description
				action	A	m		Char(4)	Allows to select if one or both sides of a quote should be deleted. BUY: Delete only the buy quote side. SELL: Delete only the sell quote side. BOTH: Delete both of the quote sides
				contractId	A	m		Long	The contract identifier
			clOrdId		CE	o	0..n	Char(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.

6.5.5 DeleteQuotesResp

- **Type:** Management Response; Broadcast
- **Response to:** DeleteAllQuotesReq (sent to the private response queue)
- **Roles:** Market maker, (Market Operation trading on behalf)
- **Broadcasted:** 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.

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	
			state	A	m		Char(3)	Valid values: <ul style="list-style-type: none"> • SUC: completed without errors • REJ: the request was rejected • ERR: there were errors during processing of the request
	StandardHeader			SE	m	1	Structure	Standard header of each message.
			marketId	A	m		Char(255)	Market Identification Code (MIC) of the market to which the request is sent or from which the request originates.
			onBehalfUserId	A	o		Integer	UserID of the target user in case of On-behalf trading (see On-behalf Trading chapter in <i>DFS180</i>).
			clientData	SE	o	0..1	Structure	For order management transactions the client data fields in this section can be used by the client to store information or meta-data about a request. None of this information is used by the backend and it is returned to the client with the response.
			clientDataInt	A	o		Integer	Integer for client's usage.
			clientDataString	A	o		Char(255)	String for client's usage.

XML Tag				Type	m/o	No.	Data Type	Short description
			clientCorrelationId	A	o		Char(255)	Correlation Id for client's usage.
ErrorList				SE	m	1	Structure	The list of Clearing houses for the quotes
			Error	SE	o	0..n	Structure	The grouping error element for each contract
			contractId	A	o		Long	The contract identifier
			Error	SE	m	1	Structure	The single error element
			err	A	m		Char(255)	The error message for this error. Always in the English language with variables already replaced by their values.
			errCode	A	m		Integer	The error code of the error. In case an error message does not have a specific error code, the value of 0 will be used.
			clOrdId	A	o		Char(40)	The client order id
			VarList	SE	o	0..1	Structure	A list of variables used in the err text field
			Var	SE	o	0..n	Structure	Structure containing information on variables
			id	A	m		Integer	In Error Response and DeleteQuotesResp, 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		Char(255)	Value of the variable (e.g. Acct1)

6.5.6 QuoteReq

- **Type:** Inquiry Request
- **Routing Keys:** `m7.request.inquiry`
- **Roles:** Trader, Market operator trading on behalf
- **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	1	Structure	Standard header of each message.
			marketId	A	m		Char(255)	Market Identification Code (MIC) of the market to which the request is sent or from which the request originates.
			onBehalfUserId	A	o		Integer	UserID of the target user in case of On-behalf trading (see On-behalf Trading chapter in <i>DFS180</i>).

XML Tag		Type	m/o	No.	Data Type	Short description
	clientData	SE	o	0..1	Structure	For order management transactions the client data fields in this section can be used by the client to store information or meta-data about a request. None of this information is used by the backend and it is returned to the client with the response.
	clientDataInt	A	o		Integer	Integer for client's usage.
	clientDataString	A	o		Char(255)	String for client's usage.
	clientCorrelationId	A	o		Char(255)	Correlation Id for client's usage.
	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.
	sellQty	A	o		Integer	The quantity of the bid side.
	buyPx	A	o		Long	The ask price of the quote.
	buyQty	A	o		Integer	The quantity of the ask side.

6.5.7 QuoteResp

- **Type:** Inquiry Response, Broadcast
- **Broadcast:** Yes
- **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	1	Structure	Standard header of each message.
	marketId	A	m		Char(255)	Market Identification Code (MIC) of the market to which the request is sent or from which the request originates.
	onBehalfUserId	A	o		Integer	UserID of the target user in case of On-behalf trading (see On-behalf Trading chapter in <i>DFS180</i>).
	clientData	SE	o	0..1	Structure	For order management transactions the client data fields in this section can be used by the client to store information or meta-data about a request. None of this information is used by the backend and it is returned to the client with the response.
	clientDataInt	A	o		Integer	Integer for client's usage.
	clientDataString	A	o		Char(255)	String for client's usage.

XML Tag			Type	m/o	No.	Data Type	Short description
		clientCorrelationId	A	o		Char(255)	Correlation Id for client's usage.
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.
		sellQty	A	o		Integer	The quantity of the bid side.
		buyPx	A	o		Long	The ask price of the quote.
		buyQty	A	o		Integer	The quantity of the ask side.

6.6 Reference Data

6.6.1 UserRprt

- **Type:** Management Response, Broadcast
- **Response to:** LoginReq
- **Broadcasted:** Yes
- **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	1	Structure	
	StandardHeader		SE	m	1	Structure	Standard header of each message.
		marketId	A	m		Char(255)	Market Identification Code (MIC) of the market to which the request is sent or from which the request originates.
		onBehalfUserId	A	o		Integer	UserID of the target user in case of On-behalf trading (see On-behalf Trading chapter in <i>DFS180</i>).
	clientData		SE	o	0..1	Structure	For order management transactions the client data fields in this section can be used by the client to store information or meta-data about a request. None of this information is used by the backend and it is returned to the client with the response.
		clientDataInt	A	o		Integer	Integer for client's usage.
		clientDataString	A	o		Char(255)	String for client's usage.
		clientCorrelationId	A	o		Char(255)	Correlation Id for client's usage.
Usr			SE	m	1	Structure	
		sessionId	A	m		Long	The current session id of the user given after the login to the system.

XML Tag		Type	m/o	No.	Data Type	Short description
	connectionLossMsg	A	o		Char(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
	mbrName	A	m		Char(255)	The name of member that the user belongs to
	usrId	A	m		Integer	The unique identifier of a user.
	revisionNo	A	m		Long	Revision number of this User. Always increasing upon a change. In a request, this is the revision number before the change. The backend will verify if this revision number matches the current revision number. If not, ErrResp will be returned.
	name	A	m		Char(255)	The name of the user.
	usrCode	A	m		Char(6)	The user's user code. The maximum length is 6 characters.
	mbrId	A	m		Char(5)	The member Id that the user belongs to.
	defaultAcctId	A	m		Char(32)	The specified default account of the user.
	state	A	m		Char(4)	The current state of the User. Valid values: <ul style="list-style-type: none"> • 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.
	AssgAcctId	CE	o	0..n	Char(32)	Contains the accounts assigned to the user
	UsrRole	CE	m	1..n	Char(255)	Contains the user roles assigned to the user

6.6.2 MbrChangeRprt

- **Type**: Broadcast
- **Response to**: –
- **Roles**: All
- **Broadcasted**: Yes
- **Broadcast Routing Keys**: [schema-version].public
- **Broadcast Audience**: 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
MbrChangeRprt		SE	m	1	Structure	
	mbrId	A	m		Char(5)	The ID of the Member.
	name	A	m		Char(255)	The member's name

XML Tag		Type	m/o	No.	Data Type	Short description
	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. Valid values: <ul style="list-style-type: none"> • 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.
StandardHeader		SE	m	1	Structure	Standard header of each message.
	marketId	A	m		Char(255)	Market Identification Code (MIC) of the market to which the request is sent or from which the request originates.
	onBehalfUserId	A	o		Integer	UserID of the target user in case of On-behalf trading (see On-behalf Trading chapter in <i>DFS180</i>).
clientData		SE	o	0..1	Structure	For order management transactions the client data fields in this section can be used by the client to store information or meta-data about a request. None of this information is used by the backend and it is returned to the client with the response.
	clientDataInt	A	o		Integer	Integer for client's usage.
	clientDataString	A	o		Char(255)	String for client's usage.
	clientCorrelationId	A	o		Char(255)	Correlation Id for client's usage.

6.6.3 DivryAreaInfoReq

- **Type:** Inquiry Request
- **Routing Keys:** `m7.request.inquiry`
- **Roles:** All
- **Request Limits:** 14/70

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
DivryAreaInfoReq		SE	m	1	Structure	
	acctId	A	o		Char(32)	Unique identifier of an account
StandardHeader		SE	m	1	Structure	Standard header of each message.
	marketId	A	m		Char(255)	Market Identification Code (MIC) of the market to which the request is sent or from which the request originates.

XML Tag			Type	m/o	No.	Data Type	Short description
		onBehalfUserId	A	o		Integer	UserID of the target user in case of On-behalf trading (see On-behalf Trading chapter in <i>DFS180</i>).
		clientData	SE	o	0..1	Structure	For order management transactions the client data fields in this section can be used by the client to store information or meta-data about a request. None of this information is used by the backend and it is returned to the client with the response.
		clientDataInt	A	o		Integer	Integer for client's usage.
		clientDataString	A	o		Char(255)	String for client's usage.
		clientCorrelationId	A	o		Char(255)	Correlation Id for client's usage.
		prodName	CE	o	0..1000	Char(255)	List of products

6.6.4 DlvryAreaInfoRprt

- **Type:** Inquiry Response; Broadcast
- **Response to:** DlvryAreaInfoReq (sent to the private response queue)
- **Roles:** All
- **Broadcasted:** Yes
- **Broadcast Routing Keys:** `[schema-version].dlvr.[dlvryAreaId],[schema-version].public`
- **Broadcast Audience:** 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	1	Structure	Standard header of each message.
		marketId	A	m		Char(255)	Market Identification Code (MIC) of the market to which the request is sent or from which the request originates.
		onBehalfUserId	A	o		Integer	UserID of the target user in case of On-behalf trading (see On-behalf Trading chapter in <i>DFS180</i>).
		clientData	SE	o	0..1	Structure	For order management transactions the client data fields in this section can be used by the client to store information or meta-data about a request. None of this information is used by the backend and it is returned to the client with the response.
		clientDataInt	A	o		Integer	Integer for client's usage.
		clientDataString	A	o		Char(255)	String for client's usage.
		clientCorrelationId	A	o		Char(255)	Correlation Id for client's usage.
		DlvryAreaList	SE	o	0..1	Structure	
		DlvryArea	SE	o	0..n	Structure	

XML Tag			Type	m/o	No.	Data Type	Short description
		revisionNo	A	m		Long	Revision number. With every change of the delivery area this value is increased by one. For a request, it should be the revision number before the change. The back-end will verify if this revision number matches the current revision number. If not, ErrResp will be returned.
		remoteState	A	o		Char(4)	The current Remote Status of the delivery area. The following values are allowed: <ul style="list-style-type: none"> • 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.
		name	A	m		Char(255)	The name of the delivery area usually used for display purposes.
		longName	A	m		Char(255)	The long name of the delivery area.
		mktAreald	A	m		Char(16)	The EIC id of the market area to which the delivery area belongs to.
		dlvryAreald	A	m		Char(16)	The Delivery Area Id. In the power market this corresponds to the EIC code. Either way, this value must be unique.
		state	A	m		Char(4)	The current Local Status of the delivery area. Valid values: <ul style="list-style-type: none"> • ACTI: The delivery area is active. It is possible to trade in that area. • SUSP: The delivery area is deactivated (hibernated). Trading in that delivery area is not possible. • DELE: The delivery area was deleted. Trading is not possible.
		prodName	CE	o	0..n	Char(255)	List of assigned products. In case of a state change for a delivery area, this list is not provided. In ModifyDlvryAreaReq, this list will replace the list of assigned products.

6.6.5 MktAreaInfoReq

- **Type**: Inquiry Request
- **Routing Keys**: `m7.request.inquiry`
- **Roles**: All
- **Request Limits**: 14/70

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	1	Structure	
	acctId	A	o		Char(32)	ID of the account. If not specified, all Market areas of all delivery areas assigned to requesting user are returned.
StandardHeader		SE	m	1	Structure	Standard header of each message.

XML Tag		Type	m/o	No.	Data Type	Short description
	marketId	A	m		Char(255)	Market Identification Code (MIC) of the market to which the request is sent or from which the request originates.
	onBehalfUserId	A	o		Integer	UserID of the target user in case of On-behalf trading (see On-behalf Trading chapter in <i>DFS180</i>).
	clientData	SE	o	0..1	Structure	For order management transactions the client data fields in this section can be used by the client to store information or meta-data about a request. None of this information is used by the backend and it is returned to the client with the response.
	clientDataInt	A	o		Integer	Integer for client's usage.
	clientDataString	A	o		Char(255)	String for client's usage.
	clientCorrelationId	A	o		Char(255)	Correlation Id for client's usage.
	prodName	CE	o	0..1000	Char(255)	List of products

6.6.6 MktAreaInfoRprt

- **Type:** Inquiry Response; Broadcast
- **Response to:** MktAreaInfoReq (sent to the private response queue)
- **Roles:** All
- **Broadcasted:** Yes
- **Broadcast Routing Keys:** `[schema-version].mkt.[marketAreaId],[schema-version].public`
- **Broadcast Audience:** All

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

XML Tag		Type	m/o	No.	Data Type	Short description
MktAreaInfoRprt		SE	m	1	Structure	
	StandardHeader	SE	m	1	Structure	Standard header of each message.
	marketId	A	m		Char(255)	Market Identification Code (MIC) of the market to which the request is sent or from which the request originates.
	onBehalfUserId	A	o		Integer	UserID of the target user in case of On-behalf trading (see On-behalf Trading chapter in <i>DFS180</i>).
	clientData	SE	o	0..1	Structure	For order management transactions the client data fields in this section can be used by the client to store information or meta-data about a request. None of this information is used by the backend and it is returned to the client with the response.
	clientDataInt	A	o		Integer	Integer for client's usage.
	clientDataString	A	o		Char(255)	String for client's usage.
	clientCorrelationId	A	o		Char(255)	Correlation Id for client's usage.
	MktAreaList	SE	o	0..1	Structure	
	MktArea	SE	o	0..n	Structure	

XML Tag			Type	m/o	No.	Data Type	Short description
		revisionNo	A	m		Long	<p>In a request, the revision number of the information on the Market Area before the change. The backend will verify if this revision number matches the current revision number. If it does not, an ErrResp will be returned.</p> <p>In a report, the revision number of the information on the Market Area after the change. With every Market Area change this value is increased by one.</p>
		mktAreald	A	m		Char(16)	The market Area Id. In the power market, this corresponds to the EIC code of the market area.
		name	A	m		Char(255)	The name of the market area, that is usually used for display purposes.
		longName	A	m		Char(255)	Usually the long name of the market area.
		state	A	m		Char(4)	<p>In a CreateMktAreaReq, it is the desired state of the market area.</p> <p>In a ModifyMktAreaReq, it is the current state of the market area.</p> <p>Valid values:</p> <ul style="list-style-type: none"> • IACT: The market area is inactive and thus not tradable. • ACTI: The market area is active. It is possible to trade in that area. • SUSP: The market area is deactivated (hibernated). Trading in this market area is not possible. This state is not available for Create Market Area message. • DELE: The market area was deleted. Trading in this market area is not possible. This state is not available for Create Market Area Request and Modify Market Area Request messages.

6.6.7 AcctInfoReq

- **Type:** Inquiry Request
- **Routing Keys:** `m7.request.inquiry`
- **Roles:** All
- **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	1	Structure	
	StandardHeader		SE	m	1	Structure	Standard header of each message.
		marketId	A	m		Char(255)	Market Identification Code (MIC) of the market to which the request is sent or from which the request originates.
		onBehalfUserId	A	o		Integer	UserID of the target user in case of On-behalf trading (see On-behalf Trading chapter in <i>DFS180</i>).

XML Tag			Type	m/o	No.	Data Type	Short description
		clientData	SE	o	0..1	Structure	For order management transactions the client data fields in this section can be used by the client to store information or meta-data about a request. None of this information is used by the backend and it is returned to the client with the response.
		clientDataInt	A	o		Integer	Integer for client's usage.
		clientDataString	A	o		Char(255)	String for client's usage.
		clientCorrelationId	A	o		Char(255)	Correlation Id for client's usage.
		acctId	CE	o	0..1000	Char(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.

6.6.8 AcctInfoRprt

- **Type:** Inquiry Response
- **Response to:** AcctInfoReq (sent to private response queue)
- **Roles:** All
- **Broadcasted:** Yes
- **Broadcast Routing Keys:** [schema-version].[acctId],[schema-version].public
- **Broadcast Audience:** 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	1	Structure	
		StandardHeader	SE	m	1	Structure	Standard header of each message.
		marketId	A	m		Char(255)	Market Identification Code (MIC) of the market to which the request is sent or from which the request originates.
		onBehalfUserId	A	o		Integer	UserID of the target user in case of On-behalf trading (see On-behalf Trading chapter in <i>DFS180</i>).
		clientData	SE	o	0..1	Structure	For order management transactions the client data fields in this section can be used by the client to store information or meta-data about a request. None of this information is used by the backend and it is returned to the client with the response.
		clientDataInt	A	o		Integer	Integer for client's usage.
		clientDataString	A	o		Char(255)	String for client's usage.
		clientCorrelationId	A	o		Char(255)	Correlation Id for client's usage.
		AcctList	SE	o	0..1	Structure	List of accounts
		Acct	SE	o	0..n	Structure	Account element

XML Tag			Type	m/o	No.	Data Type	Short description
		revisionNo	A	m		Long	Revision number of this Account. Always increasing upon a change. In a request, it is the revision before a change. Backend will verify this value against the current revision number of the account and return ErrResp if incorrect.
		acctId	A	m		Char(32)	The unique identifier of the account
		name	A	m		Char(64)	The name of the account
		mbrId	A	m		Char(5)	The unique identifier of the associated member
		preArrangedAvbl	A	o		Boolean	A flag which determines if this account can be used for entering pre-arranged (OTC) orders.
		state	A	m		Char(4)	The current state of the account. The following values are allowed: <ul style="list-style-type: none"> • 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. This state is not available for CreateAcctsReq and ModifyAcctsReq. • DELE: The account is deleted. Trading using this account is not possible. • SUSP: The account is suspended. Trading using this account is not possible.
		dlvryAreaId	CE	o	0..n	Char(16)	Assigned delivery areas
		prodName	CE	o	0..n	Char(255)	Assigned products
		assignedAcctIds	CE	o	0..n	Char(32)	List of assigned accounts for broker
		clgAcctId	CE	o	0..n	Integer	Assigned clearing accounts

6.6.9 AcctChangeRprt

- **Type**: Broadcast
- **Response to**: –
- **Roles**: All
- **Broadcasted**: Yes
- **Broadcast Routing Keys**: `[schema-version].public`

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	1	Structure	
	StandardHeader		SE	m	1	Structure	Standard header of each message.
		marketId	A	m		Char(255)	Market Identification Code (MIC) of the market to which the request is sent or from which the request originates.

XML Tag			Type	m/o	No.	Data Type	Short description
		onBehalfUserId	A	o		Integer	UserID of the target user in case of On-behalf trading (see On-behalf Trading chapter in <i>DFS180</i>).
		clientData	SE	o	0..1	Structure	For order management transactions the client data fields in this section can be used by the client to store information or meta-data about a request. None of this information is used by the backend and it is returned to the client with the response.
		clientDataInt	A	o		Integer	Integer for client's usage.
		clientDataString	A	o		Char(255)	String for client's usage.
		clientCorrelationId	A	o		Char(255)	Correlation Id for client's usage.
		AcctList	SE	m	1	Structure	List of accounts
		Acct	SE	m	1..n	Structure	Account element
		revisionNo	A	m		Long	Revision number of this Account. Always increasing upon a change.
		acctId	A	m		Char(32)	The unique identifier of the account
		name	A	m		Char(64)	The name of the account
		mbrId	A	m		Char(5)	The unique identifier of the associated member
		preArrangedAvlbl	A	o		Boolean	A flag which determines if this account can be used for entering pre-arranged (OTC) orders.
		state	A	m		Char(4)	The current state of the account. The following values are allowed: <ul style="list-style-type: none"> • 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. This state is not available for CreateAcctsReq and ModifyAcctsReq. • DELE: The account is deleted. Trading using this account is not possible. • SUSP: The account is suspended. Trading using this account is not possible.
		dlvryAreald	CE	o	0..n	Char(16)	Assigned delivery areas
		prodName	CE	o	0..n	Char(255)	Assigned products

6.6.10 AllUsersReq

- **Type:** Inquiry Request
- **Routing Keys:** `m7.request.inquiry`
- **Roles:** Trader, Market Operation, Broker, Market Maker
- **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	1	Structure	Request to retrieve all users of the system. Traders are only allowed to retrieve the users of their own member.
	StandardHeader	SE	m	1	Structure	Standard header of each message.
	marketId	A	m		Char(255)	Market Identification Code (MIC) of the market to which the request is sent or from which the request originates.
	onBehalfUserId	A	o		Integer	UserID of the target user in case of On-behalf trading (see On-behalf Trading chapter in <i>DFS180</i>).
	clientData	SE	o	0..1	Structure	For order management transactions the client data fields in this section can be used by the client to store information or meta-data about a request. None of this information is used by the backend and it is returned to the client with the response.
	clientDataInt	A	o		Integer	Integer for client's usage.
	clientDataString	A	o		Char(255)	String for client's usage.
	clientCorrelationId	A	o		Char(255)	Correlation Id for client's usage.
	mbId	CE	o	0..1000	Char(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.

6.6.11 AllUsersResp

- **Type:** Inquiry Response
- **Response to:** AllUsersReq (sent to private response queue)
- **Roles:** Trader, Market Operation, Broker, Market Maker, Sales
- **Broadcasted:** No
- **Broadcast Routing Keys:** –

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	1	Structure	Standard header of each message.
	marketId	A	m		Char(255)	Market Identification Code (MIC) of the market to which the request is sent or from which the request originates.
	onBehalfUserId	A	o		Integer	UserID of the target user in case of On-behalf trading (see On-behalf Trading chapter in <i>DFS180</i>).
	clientData	SE	o	0..1	Structure	For order management transactions the client data fields in this section can be used by the client to store information or meta-data about a request. None of this information is used by the backend and it is returned to the client with the response.

XML Tag			Type	m/o	No.	Data Type	Short description
		clientDataInt	A	o		Integer	Integer for client's usage.
		clientDataString	A	o		Char(255)	String for client's usage.
		clientCorrelationId	A	o		Char(255)	Correlation Id for client's usage.
UsrList			SE	m	1	Structure	
	Usr		SE	o	0..n	Structure	
		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
		sendSMS	A	o		Boolean	Send an SMS for order quotes.
		mobileNumber	A	o		Char(255)	The receiving mobile number
		mailAddress	A	o		Char(255)	Receiving mail address
		loginId	A	o		Char(255)	Login ID for the user
		usrId	A	m		Integer	The unique identifier of a user.
		revisionNo	A	m		Long	Revision number of this User. Always increasing upon a change. In a request, this is the revision number before the change. The back-end will verify if this revision number matches the current revision number. If not, ErrResp will be returned.
		name	A	m		Char(255)	The name of the user.
		usrCode	A	m		Char(6)	The user's user code. The maximum length is 6 characters.
		mbrId	A	m		Char(5)	The member Id that the user belongs to.
		defaultAcctId	A	m		Char(32)	The specified default account of the user.
		state	A	m		Char(4)	The current state of the User. Valid values: <ul style="list-style-type: none"> • 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.
		AssgAcctId	CE	o	0..n	Char(32)	Contains the accounts assigned to the user
		UsrRole	CE	m	1..n	Char(255)	Contains the user roles assigned to the user

6.6.12 BespokeContractReq

- **Type:** Management Request
- **Routing Keys:** `m7.request.management`
- **Roles:** All
- **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	1	Structure	
	name	A	m		Char(128)	The contract name. This is used for display purposes.
	strikePx	A	o		Long	In the event that the bespoke contract is an option, this field contains the strike price of the contract.
	CallPut	A	o		Char(1)	In the event that the bespoke contract is an option, this field contains a contract type. Valid values: <ul style="list-style-type: none"> • C: Call • P: Put
StandardHeader		SE	m	1	Structure	Standard header of each message.
	marketId	A	m		Char(255)	Market Identification Code (MIC) of the market to which the request is sent or from which the request originates.
	onBehalfUserId	A	o		Integer	UserID of the target user in case of On-behalf trading (see On-behalf Trading chapter in <i>DFS180</i>).
clientData		SE	o	0..1	Structure	For order management transactions the client data fields in this section can be used by the client to store information or meta-data about a request. None of this information is used by the backend and it is returned to the client with the response.
	clientDataInt	A	o		Integer	Integer for client's usage.
	clientDataString	A	o		Char(255)	String for client's usage.
	clientCorrelationId	A	o		Char(255)	Correlation Id for client's usage.
UndrIngContracts		SE	m	1	Structure	List of underlying contracts
	contractId	CE	m	1..n	Long	The underlying Contract identification

6.6.13 RiskSetInfoReq

- **Type:** Inquiry Request
- **Routing Keys:** `m7.request.inquiry`
- **Roles:** All
- **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	1	Structure	
StandardHeader		SE	m	1	Structure	Standard header of each message.
	marketId	A	m		Char(255)	Market Identification Code (MIC) of the market to which the request is sent or from which the request originates.
	onBehalfUserId	A	o		Integer	UserID of the target user in case of On-behalf trading (see On-behalf Trading chapter in <i>DFS180</i>).

XML Tag			Type	m/o	No.	Data Type	Short description
		clientData	SE	o	0..1	Structure	For order management transactions the client data fields in this section can be used by the client to store information or meta-data about a request. None of this information is used by the backend and it is returned to the client with the response.
		clientDataInt	A	o		Integer	Integer for client's usage.
		clientDataString	A	o		Char(255)	String for client's usage.
		clientCorrelationId	A	o		Char(255)	Correlation Id for client's usage.
		riskSetId	CE	o	0..1000	Long	The ID of the risk set to be returned. When provided, this risk set is returned; otherwise all risk sets are returned.

6.6.14 RiskSetInfoRprt

- **Type:** Inquiry Response, Broadcast
- **Response to:** RiskSetInfoReq (sent to private response queue)
- **Broadcasted:** Yes
- **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	1	Structure	
		StandardHeader	SE	m	1	Structure	Standard header of each message.
		marketId	A	m		Char(255)	Market Identification Code (MIC) of the market to which the request is sent or from which the request originates.
		onBehalfUserId	A	o		Integer	UserID of the target user in case of On-behalf trading (see On-behalf Trading chapter in <i>DFS180</i>).
		clientData	SE	o	0..1	Structure	For order management transactions the client data fields in this section can be used by the client to store information or meta-data about a request. None of this information is used by the backend and it is returned to the client with the response.
		clientDataInt	A	o		Integer	Integer for client's usage.
		clientDataString	A	o		Char(255)	String for client's usage.
		clientCorrelationId	A	o		Char(255)	Correlation Id for client's usage.
		RiskSetList	SE	m	1	Structure	
		RiskSet	SE	m	1..n	Structure	

XML Tag				Type	m/o	No.	Data Type	Short description
			status	A	m		Char(4)	The type of action which was taken on the risk set. Valid values: <ul style="list-style-type: none"> ACTI: The risk set is active ADEL: The risk set is deleted
			riskSetId	A	m		Long	The ID of the risk set
			revisionNo	A	m		Long	In a report returning as a response, the current revision number of the Risk Set entity. In a report broadcasted due to a change, the revision number of the affected entity. In a modification request, it is the revision number before the change. The back-end will verify if this revision number matches the current revision number of the entity. If not, ErrResp will be returned.
			riskSetName	A	m		Char(64)	The name of the risk set. This value must be unique.
			CashLmtAParameters	SE	m	1	Structure	Contains "a" (price-dependent) cash limit parameters
			posBuyOrdr	A	m		Double	The value of a risk parameter for a buy order submitted with a positive price
			posSellOrdr	A	m		Double	The value of a risk parameter for a sell order submitted with a positive price
			posBuyTrade	A	m		Double	The value of a risk parameter for a buy side of the trade executed with a positive price
			posSellTrade	A	m		Double	The value of a risk parameter for a sell side of the trade executed with a positive price
			negBuyOrdr	A	m		Double	The value of a risk parameter for a buy order submitted with a negative price
			negSellOrdr	A	m		Double	The value of a risk parameter for a sell order submitted with a negative price
			negBuyTrade	A	m		Double	The value of a risk parameter for a buy side of the trade executed with a negative price
			negSellTrade	A	m		Double	The value of a risk parameter for a sell side of the trade executed with a negative price
			CashLmtAlphaParameters	SE	m	1	Structure	Contains alpha (price-independent) cash limit parameters
			buyOrdr	A	m		Double	The value of a risk parameter for a buy order
			sellOrdr	A	m		Double	The value of a risk parameter for a sell order
			buyTrade	A	m		Double	The value of a risk parameter for a buy side of the trade
			sellTrade	A	m		Double	The value of a risk parameter for a sell side of the trade

7 Message Overview & Access Rights

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

General Requests and Responses

Message	Trader	Broker	Market Operation	Market Maker	Sales	Data Vendor	Settlement Operation	Clearing user	Capacity info
Login Request (LoginReq)	X	X	X	X	X	X	X	X	
Logout Request (LogoutReq)	X	X	X	X	X	X	X	X	
Logout Report (LogoutRprt)	X	X	X	X	X	X	X	X	
System Info Request (SystemInfoReq)	X	X	X	X	X	X	X	X	
System Info Response (SystemInfoResp)	X	X	X	X	X	X	X	X	
Acknowledgement Response (AckResp)	X	X	X				X	X	
Error Response (ErrResp)	X	X	X	X	X	X	X	X	
Change password Request (ChgPwdReq)	X	X	X	X	X	X	X	X	

Order Entry and Maintenance

Message	Trader	Broker	Market Operation	Market Maker	Sales	Data Vendor	Settlement Operation	Clearing user	Capacity info
Order Entry (OrdEntry)	X	X	X	X					
Order Modify (OrdModify)	X	X	X	X					
Order Request (OrdReq)	X	X	X	X					
Order Execution Report (OrdExeRprt)	X	X	X	X					
Modify All Orders (ModifyAllOrdrs)	X	X	X	X					
Order Limit Request (OrdLmtReq)	X	X	X	X					
Order Limit Report (OrdLmtRprt)	X	X	X	X					

Trade Maintenance

Message	Trader	Broker	Market Operation	Market Maker	Sales	Data Vendor	Settlement Operation	Clearing user	Capacity info
Trade Recall Request (TradeRecallReq)	X	X	X	X					

Market Information

Message	Trader	Broker	Market Operation	Market Maker	Sales	Data Vendor	Settlement Operation	Clearing user	Capacity info
Public Order Books Request (PblcOrdRBooksReq)	X	X	X	X	X	X	X		
Public Order Books Response (PblcOrdRBooksResp)	X	X	X	X	X	X	X		
Public Order Books Delta Report (PblcOrdRBooksDeltaRprt)	X	X	X	X	X	X	X		
Cash Limit Request (CashLmtReq)	X	X	X	X		X		X	
Cash Limit Report (CashLmtRprt)	X	X	X	X		X		X	
Cash Limit Delta Report (CashLmtDeltaRprt)	X	X	X	X		X		X	
Commodity Limit Request (CommodityLmtReq)	X	X	X	X					
Commodity Limit Report (CommodityLmtRprt)	X	X	X	X					
Message Request (MsgReq)	X	X	X	X	X	X	X		
Message Report (MsgRprt)	X	X	X	X	X	X	X		
Trade Capture Request (TradeCaptureReq)	X	X	X	X	X		X	X	
Trade Capture Report (TradeCaptureRprt)	X	X	X	X	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	X	X	X	X	
Contract Information Report (ContractInfoRprt)	X	X	X	X	X	X	X	X	
Product Information Request (ProdInfoReq)	X	X	X	X	X	X	X	X	
Product Information Report (ProdInfoRprt)	X	X	X	X	X	X	X	X	
Market State Request (MktStateReq)	X	X	X	X	X	X	X		

Message	Trader	Broker	Market Operation	Market Maker	Sales	Data Vendor	Settlement Operation	Clearing user	Capacity info
Market State Report (MktStateRprt)	X	X	X	X	X	X	X		
Settlement Process Information Request (StlmntProcessInfoReq)	(X) ¹²	(X) ¹²	X	X	X			X	
Settlement Process Information Report (StlmntProcessInfoRprt)	(X) ¹²	(X) ¹²	X	X	X			X	
Reference Price Request (RefPxReq)			X			X			
Reference Price Report (RefPxRprt)	X	X	X	X		X			
Implied Order Request (ImplOrdReq) ¹³	X	X	X	X					
Implied Order Report (ImplOrdRprt) ¹³	X	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'.

¹¹ The user must have the additional right 'Capacity info'.

¹² The access is configurable on the backend side for the 'Trader' role.

¹³ The message is obsolete and may be removed in the next versions.

Order Quotes

Message	Trader	Broker	Market Operation	Market Maker	Sales	Data Vendor	Settlement Operation	Clearing user	Capacity info
Order Quote Request (OrdQuoteReq)	X	X	X	X					
Order Quote Report (OrdQuoteRprt)	X	X	X	X					
Order Quote Setup Request (OrdQuoteSetupReq)	X	X		X					
Delete Quotes Request (DeleteQuotesReq)				X					
Delete Quotes Report (DeleteQuotesRprt)				X					
Quote request (QuoteReq)	X	X	X	X					
Quote response (QuoteResp)			X	X					

Reference Data

Message	Trader	Broker	Market Operation	Market Maker	Sales	Data Vendor	Settlement Operation	Clearing user	Capacity info
User Report (UserRprt)	X	X	X	X	X	X	X		
Account Change Report (AcctChangeRprt)	X	X	X	X	X	X	X		
Member Change Report (MbrChangeRprt)	X	X	X	X	X	X	X		
Delivery Area Information Request (DlvryAreaInfoReq)	X	X	X	X	X	X	X		
Delivery Area Information Report (DlvryAreaInfoRprt)	X	X	X	X	X	X	X		
Market Area Information Request (MktAreaInfoReq)	X	X	X	X	X	X	X		
Market Area Information Report (MktAreaInfoRprt)	X	X	X	X	X	X	X		
Account Information Request (AcctInfoReq)	X	X	X	X	X	X	X	X	
Account Information Report (AcctInfoRprt)	X	X	X	X	X	X	X	X	
All Users Request (AllUsersReq)	X	X	X	X					
All Users Response (AllUsersResp)	X	X	X	X					

Message	Trader	Broker	Market Operation	Market Maker	Sales	Data Vendor	Settlement Operation	Clearing user	Capacity info
Bespoke contract creation request (BespokeContractReq)		X	X						
Risk Set Information Request (RiskSetInfoReq)	X	X	X	X	X	X	X	X	
Risk Set Information Report (RiskSetInfoRprt)	X	X	X	X	X	X	X	X	

8 Forward Compatibility

8.1 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.x 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);

}
```

8.2 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.

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

2. Full reference can be found on <https://www.rabbitmq.com/protocol.html> ↩
3. Full reference can be found on <https://www.rabbitmq.com/protocol.html> ↩
4. VeriSign: <http://www.verisign.com/support/roots.html>. Comodo: https://support.comodo.com/index.php?_m=downloads&_a=view&parentcategoryid=1 ↩
5. By default, the parameter is set to 90 days. ↩
6. By default, the parameter is set to 10 days. ↩
7. *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 [Sequence counting for Broadcast Messages](#). ↩
8. 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 contract 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. ↩
9. The values depend on the client's configuration of the maximal number of hours that can be covered by one inquiry request (i.e. the difference in the endDate and startDate attributes). The specified request limits apply to the default configuration of 7 hours. ↩
10. The values depend on the client's configuration of the maximal number of hours that can be covered by one inquiry request (i.e. the difference in the endDate and startDate attributes). The specified request limits apply to the default configuration of 7 hours. ↩
11. The values depend on the client's configuration of the maximal number of hours that can be covered by one inquiry request (i.e. the difference in the endDate and startDate attributes). The specified request limits apply to the default configuration of 7 hours. ↩