DFS180 - M7 - Public Message Interface

M7 Release 6.21.189

Date	26.05.2026
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

Disclaimer

Terminology

Term	Description
Balancing Phase	An additional trading phase, which starts after the end of the normal trading phase, and ends in relation to the delivery start. Normal orders from market participants can be entered, but cannot trigger a trade execution. Only balance orders can trigger a trade execution (a balance order entry is usually limited to one specific member). The order book can be crossed during the balancing phase.
Client	Program or application acting as a client of the AMQP server.
LTS	Local Trading Solution. An instance of M7 that can operate in a multi-exchange mode with a connection to the master XBID instance or in a stand-alone mode providing for local trading.
Trader	Person that logs into the client to participate in the market.
XBID	Cross-Border Intra-Day (XBID) Solution for the European electricity intra-day market. The master instance to which an LTS can connect to gain the benefits of the Shared Order Book (SOB), a Capacity Management Module (CMM) and a Shipping Module (SM).

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 protocol[1] on the application layer of the OSI model.

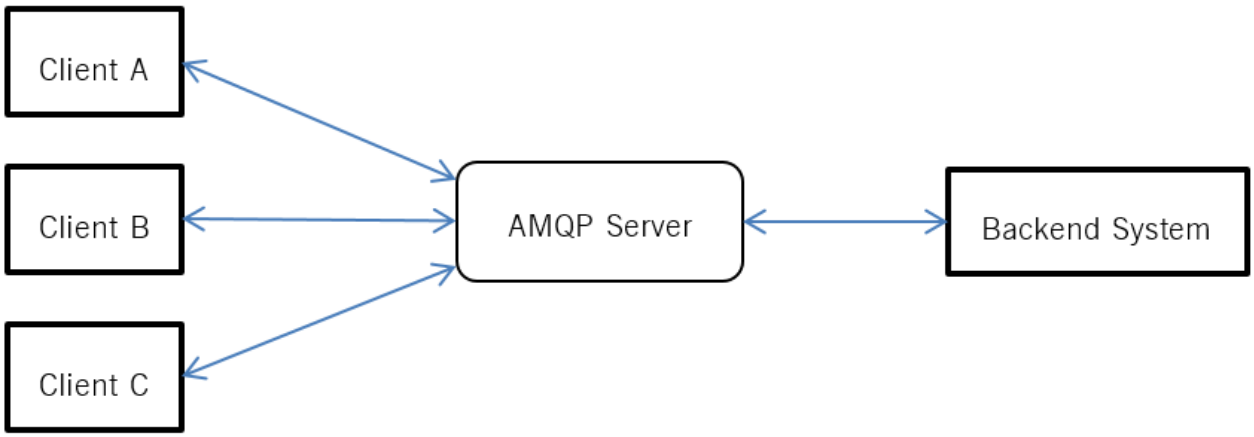
Payload of the messages sent over AMQP is formatted in XML. See www.w3.org/XML for information about XML.

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.

1.2 Public Message Interface Terms of Service

Thank you for using the M7 Trading Public Message Interface. By accessing and using this API, you agree to the terms of service described below. Additional terms of use may apply per the relevant market authority. We refer to the terms listed below and throughout this document, as well as terms within the accompanying API documentation, and any applicable policies and guidelines as the “Terms.” You agree to comply with the terms and that your usage and ability to access these services is based upon your compliance to these terms. Please read this documentation and all Terms carefully. Deutsche Börse AG reserves the right to revoke access to the M7 Trading Public Message Interface (API) and any other services if connected clients are not compliant with the Terms described in this documentation.

1.2.1 Usage of the M7 Trading Public Message Interface

1.2.1.1 Permitted Access

You will only access (or attempt to access) an API by the means described in the documentation of that API. You will use the application ID and user credentials assigned to you by the issuing authority only. Where required, SSL/TLS Client Certificates may only be used by the party they have been explicitly issued to. Redistribution of SSL/TLS certificates is not permitted.

1.2.1.2 Managed End Users

All end users of the connected client will comply with the terms described here, the terms of the applicable market authority, as well as all applicable local laws and regulations.

1.2.1.3 Compliance with Law, Third Party Rights, and Market Authorities

You agree to comply with all applicable laws and regulations, as well as third party rights (including without limitation laws regarding the import or export of data or software, privacy, and local laws). You will not use the API to encourage or promote illegal activity or violation of third-party rights. You will not violate any other terms of service with Deutsche Börse (or its affiliates).

1.2.1.4 Public Message Interface Limitations

Deutsche Börse sets limits on the use of its APIs (e.g. limits towards the volume and frequency of all or some requests that you may submit), per its discretion. You will comply with the documented limits and will not make attempt to circumvent them. Further guidelines are recommended in this documentation per suggested implementation against our API. Deutsche Börse reserves the right to revoke or suspend access to clients exhibiting API usage behavior contrary to the terms, guidelines, and suggested implementation practices described in this and related documentation. For the sake of clarity, access to the API may be suspended or revoke for behavior including but not limited to the following:

- Users producing unnecessarily large volumes of entities on the message broker (e.g. AMQP ‘queues’, ‘channels’, ‘connections’ or ‘consumers’)
- Clients found unable to acknowledge or consume their messages from the message broker in an appropriate time (see section Flow Control)
- Clients not implementing exponential back-off approach for repeated re-connection attempts, or
- Clients sending messages way above the limits.

Please refer to the documentation here, and the linked third party documentation for the suggested implementation of these concepts.

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

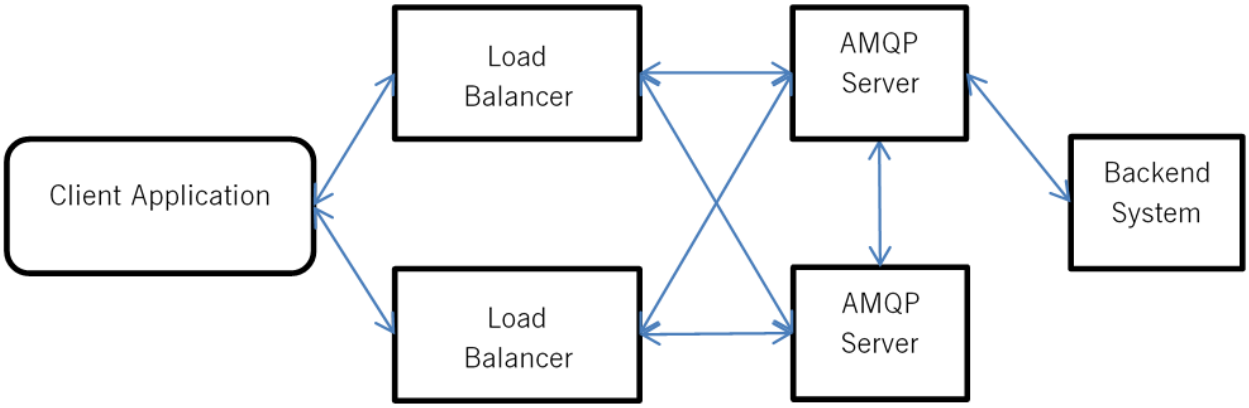
It is mandatory that all connections to the AMQP server are created with AMQP heartbeats enabled. The recommended heartbeat period is 30-60 seconds. Connections that do not have heartbeats enabled will time out on the firewall in the event that there is no traffic on the connection for a longer period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

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



Basic M7 Architecture

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

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

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

2.2 Application ID

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

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

3 Message Exchange Patterns

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

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

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

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

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

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

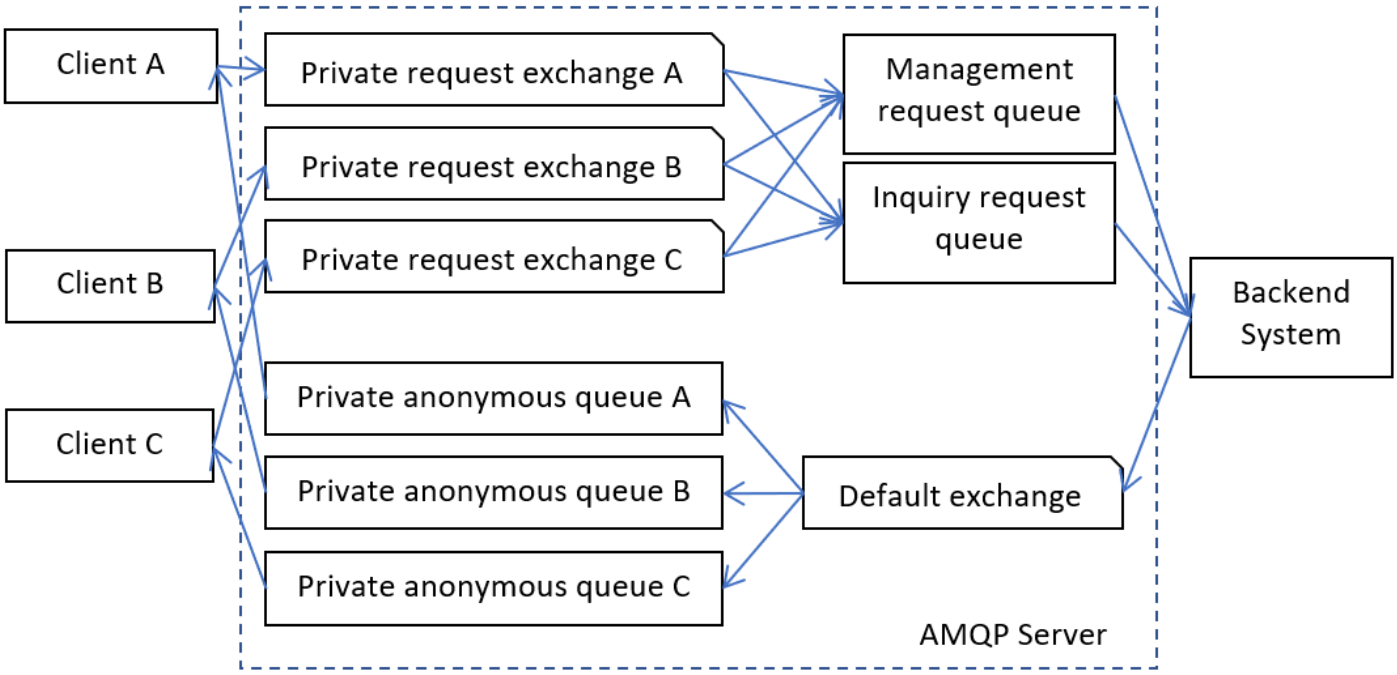
3.1 Request-Response Communication

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

3.1.1 AMQP Configuration

In case a new trader has been set up by the granting authority, the backend system will synchronise with the AMQP Server during runtime and create the necessary exchanges and bindings.

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 will not be set up initially by the AMQP Server but by 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.

First of all, each client has to send a login request using a valid trader that is configured in the M7 system. Without it, a trader cannot participate in the market. As a result of a login request, a “User response” is sent containing all information required to participate in the market. The login of a client is checked by the M7 system.

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 does not 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 M7 Trading system supports two types of requests:

- *Management-related 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, M7 will send an “AckResp” message to acknowledge the receipt of this request by

PIM Gateway for the further processing, or an “ErrResp” message to inform about possible errors. 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 M7 Trading RabbitMQ, it will receive an AckResp in reply to confirm the receipt of the order by PMI Gateway. 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 M7 Trading system confirms the receipt of the request with an AckResp message. Each individual request will be responded by an individual response message. If the system detects an error which prevents it from processing of the message (syntactical errors but also if the message content is against the M7 Trading documentation, for example if a buy ICB order has a positive peak price delta), it may reply directly with an ErrResp instead, sending it to the private response queue of the client.

When sending an order management request, the request may specify a client order id. The response sent by the M7 Trading 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 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.

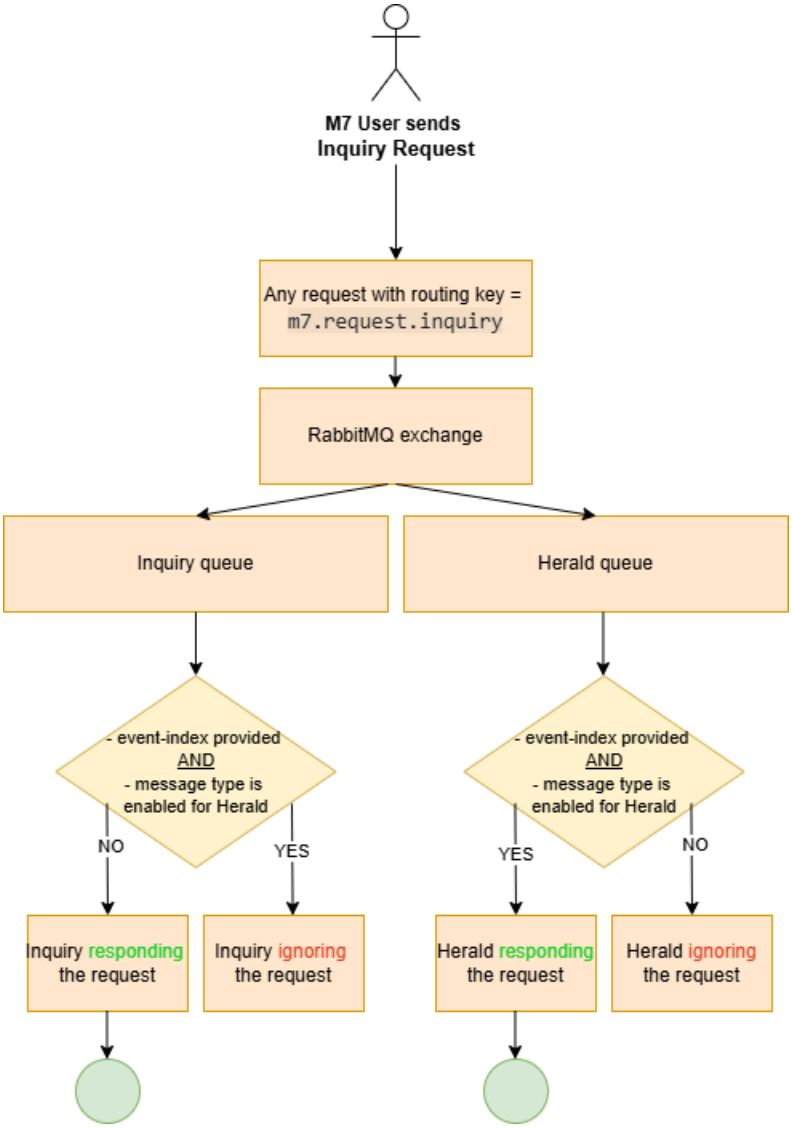
3.1.2.2 Inquiry Requests

The following inquiry request types are supported:

- *Private Message Requests* are used to retrieve the client specific private information held by the backend system.
- *Public Message Requests* are used to retrieve public information held by the backend system.
- *Trade Requests* are used to retrieve all trades for products that a client is assigned to.
- *Order Book Requests* are used to retrieve the current order book situation managed by the backend system.
- *Order Requests* are used to retrieve orders based on the client assignments.
- *Reference Data Requests* are used to retrieve reference data needed by the client for initialisation.
- *Throttling Status Requests* (if available for the exchange) are used to retrieve the current Throttling status of the member.

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.

Simplified architecture of the Inquiry Request processing:



3.1.3 How to Send Requests

In order to send a request to the M7 Trading backend system, the client shall perform the steps described in the following subchapters.

3.1.3.1 Response Queue Declaration

First of all, the client must declare at least one private exclusive response queue. This queue will be used during the entire time when the client is connected to the broker.

The maximum number of queues which a client can declare is 10. All declared queues must adhere to the following name pattern 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`

3.1.3.2 Request Preparation

Once the queue(s) have been declared, the client may proceed with the preparation of an AMQP request message.

A request is an XML-encoded message embedded in the message body. The detailed payload description of each public request can be found in Public Requests and Responses. For privileged requests mainly designated for Market Operators and Admins, please refer to *DFS180a*.

Each request must be built in line with the currently valid set of M7 Trading XSDs as well as with the business logic of the M7 Trading application.

In addition to the correct XML format, the client will also be required to set the message attributes in the AMQP message header as per AMQP Message Properties.

3.1.3.3 Event-index Inclusion

As of M7 6.16, it is recommended to include the latest value of the optional message property `event-index` in all inquiry requests sent to M7. For more information, please refer to Event-Index.

3.1.3.4 Sending Request

The client shall send the XML-encoded request from Request Preparation to the `m7.requestExchange.<login-id>` exchange. The routing key to be used depends on the type of the request:

- `m7.request.management` routing key for management requests
- `m7.request.inquiry` routing key for inquiry requests
- `m7.request.inquiry.throttling` for the current throttling status of the Member

The first request that the M7 Trading application will expect to receive is a LoginReq (see LoginReq).

3.1.3.5 Obtaining Response

Once the backend system has processed the request, it will send a response to the client's response queue specified in the `reply-to` attribute from the request. 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. If the processing fails (e.g. an unexpected disconnection from XBID occurs before the *OrdrExeRprt* could have been received from XBID, and the open expectation in system memory expires), the `correlation-id` might be lost, because its value is stored only in the open expectation. It is also possible to use *clientCorrelationId* attribute for request-response matching purposes.

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.

3.1.3.6 Invalid and Unrouteable Requests

If the backend system cannot process a request, because the request is incorrect or cannot be fulfilled, it will 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 when some mandatory AMQP message header attributes were omitted in the request (see Sending Request).

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.

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

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

3.1.4 How to Receive Responses

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

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

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

3.1.5 Acknowledgement of Responses

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

If 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 sthe erver side are constantly monitored and in case one of the queues hit a certain threshold, the connection might be actively disconnected by the server and the Application ID gets deactivated.

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

3.1.6 Connection Failure

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

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

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

3.1.7 Unacknowledged Requests

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

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

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

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

3.1.8 Message Overflow Handling

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

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

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

3.1.9 Flow Control

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

Several measures are taken to prevent this scenario.

3.1.9.1 Request Rate Limit

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

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

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

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

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

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

Example

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

3.1.9.2 Message Expiration

Clients may specify an expiration time for each request message. This allows the clients to protect themselves against the situation, when a request waiting in the request queue for a long time becomes obsolete, but eventually gets processed even if the client does not wish to execute the request anymore. If the expiration time has passed and the message has not been consumed from the queue, it's deleted by RabbitMQ. The RabbitMQ broker uses shorter expiration period of these two: either RabbitMQ default of 45s, or the value set by the client in AMQP Message Properties.

It is not possible to extend the default limit (45s) set by RabbitMQ. The expiration time of the message is the number of miliseconds that corresponds to the difference between the time when the message reaches the AMQP request queue and the time it is consumed by M7 Trading. If the request expires in the queue (expiration time passes before the message can be consumed by M7), it is deleted and no response or broadcast is sent to the client. If M7 consumes a message from a request queue and for some reason crashes before sending the response, the message is lost.

During the management request processing in the core, M7 Trading also follows the `expiration` set in AMQP Message Properties. If no value is specified by the client, M7 Trading will use a default value of 45 seconds. If the expiration time has passed and therefore the message got expired, it's not processed by M7 Trading. For this purpose, the expiration time is compared to the timestamp when the message was consumed by PMI Gateway.

During the inquiry request processing, M7 Trading also follows the same expiration time (the `expiration` set in AMQP Message Properties or a default value of 45 seconds). For this purpose, the expiration time is compared to the timestamp when the message was consumed by M7 Trading inquiry module.

3.1.9.3 AMQP Server Flow Control

In the AMQP protocol, *flow control* is an emergency procedure used to halt the flow of messages from a peer. It works in the same way between 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.)

3.1.10 Message Type

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

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

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

3.2 Broadcast

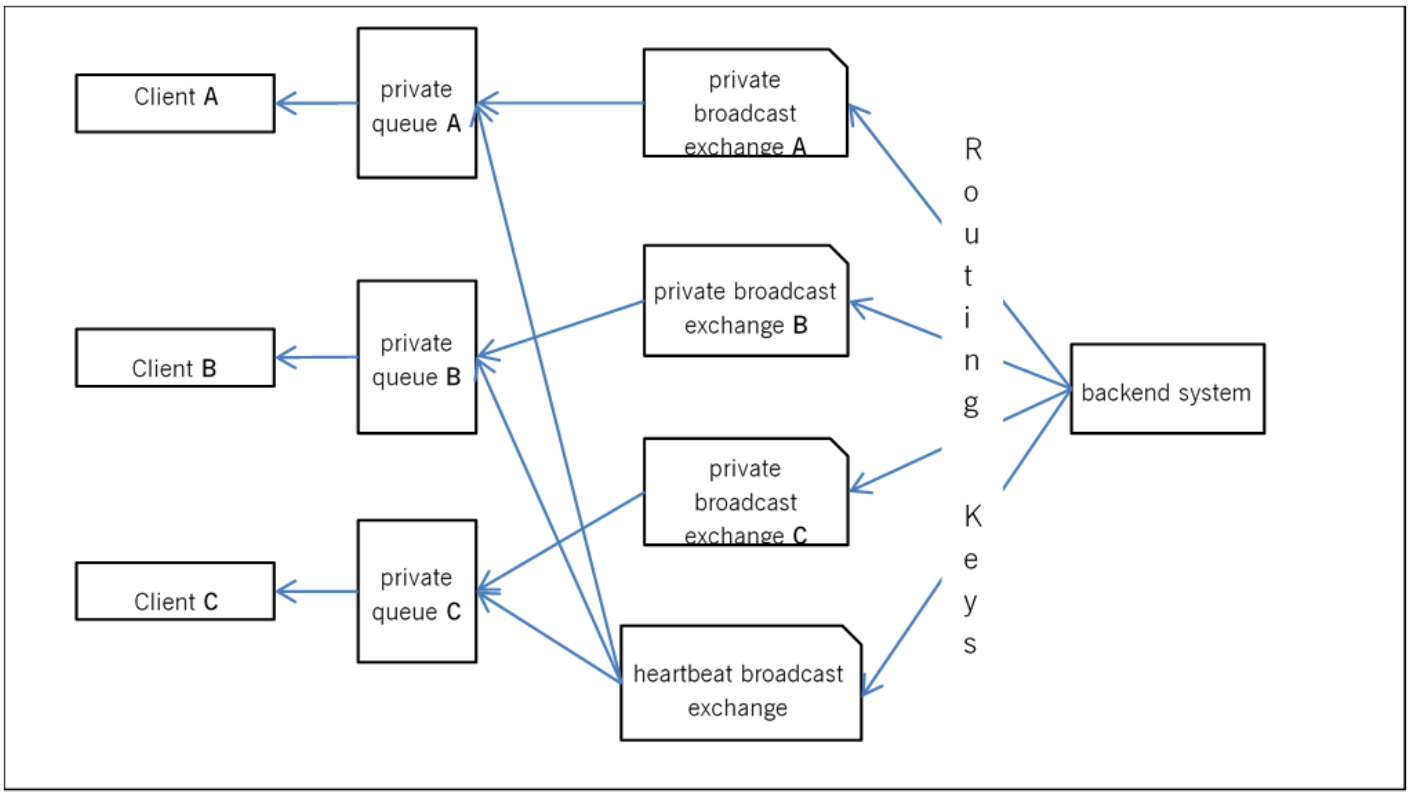
The backend system broadcasts three types of information:

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

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

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

The broadcast routing architecture of the backend system



Broadcast messages are encoded and 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 traders private queue which will be created by the AMQP server according to instructions the client specifies in the initial login procedure.

The private client's broadcast queue shall be

- created before the LoginReq message is sent
- created by the same connection that is used by the consumer
- created with the x-queue-master-locator parameter set to client-local (this configuration secures efficient message distribution through the RabbitMQ cluster). In case the parameter is not set, the broadcast queue will be created on client's behalf by the AMQP server and the most efficient message distribution path cannot be guaranteed.
- named `m7.broadcastQueue.<login-id>`
- configured to that 'x-message-ttl' and 'x-expires' parameters are set as low as possible

More details can be found in RabbitMQ documentation.

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

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

3.2.3 New Delivery Area Assignments and Broadcasts

When a new assignment of Product or Delivery Area to Balancing group is added, the binding for broadcasts is created dynamically upon this change done in M7 WebGUI.

When a new assignment of Delivery Area to Product (without updating the Balancing group assignemnt) is added, the binding for broadcasts is not created automatically. Users already assigned to the Product in question should log out of M7 Trading and log in again to get the corresponding binding created upon login.

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.

3.2.4 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`

3.2.5 Sequence Counting for Broadcast Messages

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

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

Rarely, it may happen that client gets a value which is lower than expected but still not necessarily a reason for the synchronization of market data. When the same message is received twice, for example, sequence 11 is received again after sequence 12, it may have been caused by incorrect message acknowledgment on the side of consumer application. In such case, it is safe to ignore the duplicated message, 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-synchronisation. In case of a sequence reset, all sequence IDs will be reset to 0.

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

3.2.6 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.

The recovery can be done in the following ways:

- A complete disconnection and re-connection to RabbitMQ;
- A new request for the missing data by an inquiry request on the existing RabbitMQ connection.

Event-index role and usage during Gap detection: **event-index cannot be used for Gap detection**. However, the event-index can be useful after a gap is detected (or immediately after login). In such situation (using TradeCaptureRprt broadcast as example in steps below):

1. Discard all existing market data of the particular type at the time the gap was detected (e.g., trades received via TradeCaptureRprt).
2. Stop processing broadcasts of relevant type and pile them up in order they arrive (e.g., TradeCaptureRprt broadcasts)
3. Request a new baseline data set via an Inquiry Request using the last known event-index before the gap (e.g., send TradeCaptureReq with event-index 95)
4. The Inquiry Response provides the requested data together with an event-index value “X”, which must be used as the new baseline (e.g., a TradeCaptureRprt is received with event-index 100)
5. Start applying piled up broadcasts to baseline data:
   - ignore any broadcast with an event-index “<= X” (e.g., skip TradeCaptureRprt broadcasts with event-index 100 or lower)
   - apply all broadcasts with an event-index “> X” in the order they were received (e.g., apply TradeCaptureRprt broadcasts with event-index 101 or higher)
6. Continue applying broadcast messages. The event-index is never decreased (reset) during session.

Please see Event-index for instructions on the recommended inclusion of `event-index` property.

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.

3.2.7 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);
- A connection for member;
- A connection for balancing group;

- A connection for heartbeats. The heartbeats are sent through the dedicated connection in order to minimise 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 includes also the number of connections used for the broadcast distribution from the M7 back-end to the AMQP Server as well as the distribution rules for each such connection.

3.2.8 How to Receive Broadcasts

As the trader's private broadcast queues are durable queues and are already set up 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.

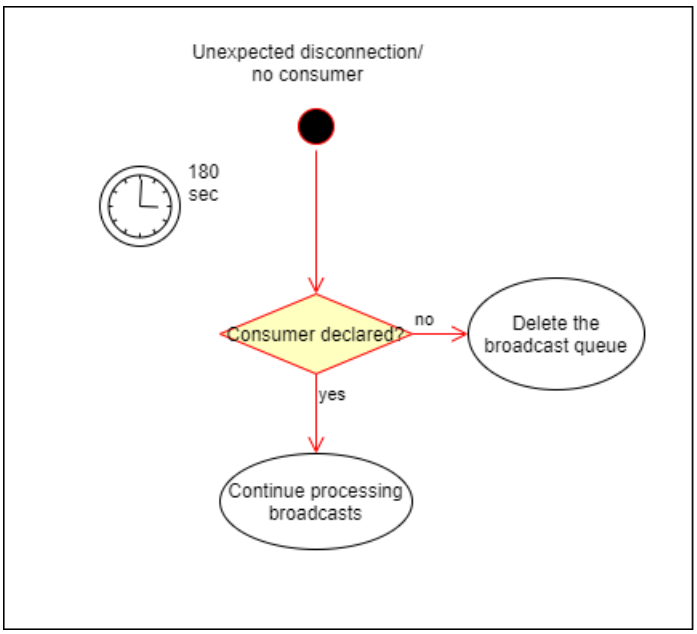
To stop receiving broadcasts, the client should remove the consumer from its private queue.

The broadcast queues within the AMQP Server are set up 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 slowly or because of any other connection problems.

3.2.8.1 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.

3.2.8.2 Event-index in Broadcasts

M7 includes the value of `event-index` message property in all broadcasts. For more details, please see Event-index.

3.2.9 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 expriration time.

3.2.10 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.

3.2.10.1 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).

3.2.10.2 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.

3.2.11 Failover Processing

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

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

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

3.2.12 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.

3.2.13 Message type

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

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

4 Security

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

4.1 Server Certificate

The backend system uses signed Server Certificates.

Usually the CA root certificates are known and trusted by the software development frameworks like Java or .NET. If not, clients need to add the CA root certificate to a list of trusted certificates. The needed CA root certificate files can be downloaded directly from the CAs Website[2].

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

4.1.1 Using CA Root Certificate in Java

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

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

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

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

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

4.2 Client Certificate

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

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

The client certificate:

- complies to the X.509v3 specification,
- uses a key length of 2048 bits,
- subject contains an unique identifier as the *Common Name*,
- is signed by Deutsche Börse CA,

- is provided in format PKCS#12 (.p12).

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

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

4.2.1 Using a Client Certificate in Java

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

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

4.2.2 Usage of ComTrader Certificates

The ComTrader certificate is strictly intended for use with the ComTrader frontend client. Any other use of the ComTrader certificate, including but not limited to its application in production environments with third-party APIs, is strictly prohibited.

It is imperative that all members adhere to the following guidelines:

Authorized Use: The ComTrader certificate must only be used with the ComTrader frontend client.

Prohibited Use: The use of the ComTrader certificate for any other purpose, including integration with third-party APIs, is forbidden.

All members must comply with this policy to ensure the integrity and security of the system. Failure to adhere to these guidelines may result in disciplinary action and revocation of access privileges.

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 Authorisation

Authorisation of client actions occurs on two levels.

4.4.1 Authorisation by an AMQP server

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

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

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

4.4.2 Authorisation by the Backend System

The backend system verifies privileges when processing requests from clients.

In the event of insufficient privileges, the request is rejected by the backend system. From the client point of view, this rejection occurs synchronously for inquiry requests as a negative response message is sent to the traders response queue and asynchronously for management requests as a negative response message is sent to the traders private exchange using the routing key `<schema-version>.trdr.<login-id>`.

5 Message Format

All messages sent between the backend system and clients have an XML-encoded payload and specific AMQP message properties. This section describes the xml payload format and the AMQP message properties in detail.

The xml payload format specification comes in the form of XML schemas, which specify the allowed structure and format of XML elements and attributes. These are covered in the section 6 Public Requests and Responses and *2 Admin Requests and Responses* in *DFS180a*.

5.1 General Information

5.1.1 AMQP Message Properties

The following message properties must or can be set when sending a request to M7:

AMQP Message Property	Description	Example	Occurrence
content-type	The message type and XML schema version used to encode the message body (payload). Valid content-type definitions: - x-m7/request; version=*y* - x-m7/response; version=*y* - x-m7/broadcast; version=*y* - x-m7/heartbeat; version=*y* - x-m7/error; version=*y* where *y* is the major XML schema version. For more information on the XML schema version please refer to XML Schema Version (content-type).	`x-m7/request, version=6.0`	Mandatory
reply-to	The name of the user's predefined private response queue to which the response has to be sent (See Request-Response Communication.)	`m7.private.responseQueue.<login-id>.queue<1>`	Mandatory
user-id	The Login Id of the user sending the request. Its value must be equal to the Login Id of the user used to open the connection.	M7DEM002	Mandatory
app-id	The Application Id given by the Granting Authority.	ComTrader	Mandatory
correlation-id	A unique request Id that can be used to match responses with requests. The uniqueness of `correlation-id` is the sole responsibility of each client and M7 Trading system will not check or enforcing it. `correlation-id` may appear in public broadcasts (e.g., in a PblcTradeConfRprt after sending an OrdrEntry request) and may, therefore, be seen by other clients. It is recommended to keep `correlation-id` unrecognisable.	`a92ca6d8-b561-45ba-ab86-931bcc8f6f51`	Mandatory
expiration	Time To Live of the request. It specifies after how many milliseconds the request should expire if not executed. By default, the expiration is set to 45 seconds. Should the client specify a different `expiration`, the shorter value of the two will be applied. See RabbitMQ Documentation for details.	`40000`	Mandatory

AMQP Message Property	Description	Example	Occurrence
content-encoding	An indicator whether the message is compressed (i.e. the content of the message is compressed using the gzip method). Valid values: ***gzip*** - The message is compressed. ***utf-8***, ***null*** - The message is not compressed.	`gzip`	Optional
headers	The version of the connecting application, server timestamp, and/or event-index. Selected messages will include metadata, pmi-processing For more details, please refer to App-version, Event-Index, Server-Timestamp, Pmi-processing, and Metadata.	`headers={event-index=1820, app-version=6.8.50, server-timestamp=1718087394807, pmi-processing=10}`	Optional

5.1.2 Message size restrictions

- Message body with headers is restricted to maximum size of 41 943 040 bytes of uncompressed size.
- Message body is restricted to maximum size of 40 894 464 bytes of uncompressed size.
- Message with headers is compressed by gzip. If it exceeds 2 621 440 bytes the message will be rejected.

5.1.3 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

5.1.4 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 2: Message layout of the Standard Message Header.

5.1.5 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.

5.1.6 XML Schema Version (content-type)

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 `content-type` field as a string.

The `content-type` should contain value that corresponds to the major schema version.

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.

Example

> For XSD schema version 6.15.40, the major XSD schema is 6.0.

5.1.7 App-version

In the `app-version`, the client may provide the version of the application being used for its connection to the M7 Trading system.

If the `app-version` is provided, M7 validates that:

1. It is higher than or equal to the minimum version for the given `app-id`.
2. Its format is either an integer number, or multiple integer numbers separated from each other by a dot sign (“.”). **Example**: *app-version*=10 or *app-version*=4.1.5
3. It is at most 32 characters long.

In case the `app-version` fails at least one of the above validations, the connection is refused.

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

5.1.8 Event-Index

As of M7 6.16, M7 introduces an `event-index` message property. From the client point of view, the main purpose of using event-index in the inquiry requests is primarily intended to speed up processing by routing the request to the faster component. From a functional perspective, the event-index serves as a guarantee that the client receives the most up-to-date data. This is necessary due to the asynchronous nature of data processing in our new component.

`event-index` is a header attribute which enables M7 to track asynchronous market events among M7 components. In this way, M7 can provide better performant responses to clients’ inquiry requests.

`event-index` increases every time M7 processes an event, e.g., order entry, contract opening or closing, both in LTS or XBID market or delivery area halt. The system sends the latest value of `event-index` in the market first in `UserRprt` after client’s login, and then with every broadcast.

M7 6.16 is the earliest version supporting this property, however, please note its activation is exchange-specific. There are no consequences for clients not to use it, whether it has been or not activated on the M7 side.

General Information

`event-index` is an optional attribute in `contentHeader`, type Long.

Its value is always increasing, but not necessarily sequential as not all events are relevant to all users due to their assignments.

Depending on the value of `event-index` the following behavior is expected:

1. Event-index(request) =< Event-index(market) –> The system sends the latest data.
2. Event-index(request) > Event-index(market) –> If after three self re-inquiries the `event-index` wasn't processed the system sends a timeout error, see DFS200 - 4.2 General errors, Error ID: 2095.
3. Event-index(request) < 0 or not a Long data type –> The system defaults the `event-index` to 0 and sends the latest data.

Inquiry requests that currently support `event-index` attribute: `TradeCaptureReq`, `PblcTradeConfReq`. The list of inquiry requests supporting the property will be gradually extended in future M7 releases.

As the attribute is optional the client should be able to receive inquiry responses both with or without the `event-index`.

Use of Event-Index During Login Procedure

When logging into M7, as a response to LoginReq, the client receives the UserRprt which has the current value of `event-index` in the market.

Note: the event-index does not reset during user session. From the user's perspective, the first event-index value visible after login is arbitrary. "Arbitrary" means any positive integer value, with no relation to value from previous session. It means the first value the user receives after login can be any value different than the last one previously observed and the user cannot predict that exact starting number.

The client should use this value in inquiry requests to get the initial set of the market and other information necessary to start a trading session. Usage of the `event-index` value from the UserRprt assures that clients get the latest market data available in the system.

Use of Event-Index in Gap Detection

In case the client detects a gap in a broadcast sequence, see Sequence Counting for Broadcast Messages, the value of an `event-index` from the latest broadcast can be used to re-inquire a request. Thus, M7 activates a new, more performant component to process the request.

Therefore, it is advised to start including the value of `event-index` in each inquiry request sent to M7 to get a quicker response.

Note: Event-index cannot be used for gap detection, i.e., to get back all the missed messages during the gap. However, the event-index can be useful after a gap is detected - see chapter 3.2.6 Gap Detection and Acknowledgments for further details.

5.1.9 Metadata

Starting from version 6.19, selected messages will include additional metadata within their headers. This metadata is intended for internal technical purposes and is not designed for end-user interaction. This feature is implemented to enhance the internal handling and processing of messages. Users are not required to interact with or modify this metadata. This metadata attribute is specific to the exchange.

The metadata consists of a base64-encoded byte array: `headers={..., metadata=N3hFeL+UKhErusxFLkrpYVYvcH6vXNtIGzQT5lbV, ...}`

5.1.10 Server-timestamp

The `server-timestamp` header is set by core in the moment when the message is converted to external format (Core Sender

time)

5.1.11 PMI-processing

The `pmi-processing` header, set by the PMI Gateway, records the time taken to process an OMT message and send an acknowledgment response (AckResp). This header is used for monitoring purposes.

5.1.12 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.

5.1.13 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 *decShftQty*, *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 3 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 3: An example of displaying quantity values

5.1.14 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).

5.1.15 Date Time Values in Messages

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

YYYY-MM-DD**T**hh:mm:ss.sss**Z** (2023-09-14T12:34:23.351Z)

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

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

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

The market time zone can be obtained using the SystemInfoResp message (see System Info Response).

5.1.16 Date Values in Messages

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

YYYY-MM-DD (2023-09-24)

5.1.17 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 balancing group.
- *On-behalf trading on a broker level* : Broker users are allowed to trade on-behalf for all traders that are assigned as the assigned members for their account regarding the user product configuration.
- *On-behalf trading on an admin level*: All Admin users are allowed to trade on-behalf for all traders in the system. In relation to the user product configuration (only products assigned to the trader can be traded on behalf even if the admin user itself has a wider assignment of products).

In order to submit requests to the backend as on-behalf requests on a broker or admin level, the standard message header field *onBehalfUserId* must be filled in with the user ID of the target user, for whom the request is sent on-behalf (see Standard Message Header). For on-behalf requests on a member level, the submission of the user Id of the target user ID is not required.

Although the standard message header is part of every message in the Public Message Interface, the field *onBehalfUserId* is only relevant for the following request types:

- Order Entry Request (OrdrEntry)
- Order Modify Request (OrdrModify)
- Order Request (OrdrReq)
- Order Execution Report (OrdrExeRprt)
- Trade Recall Request (TradeRecallReq)
- Message Request (MsgReq)
- Message Report (MsgRprt)
- Trade Capture Request (TradeCaptureReq)
- Trade Capture Report (TradeCaptureRprt)
- Order Limit Request (OrdrLmtReq)

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

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

5.1.18 Message Properties Summary Table

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

The message classname is present in the table header.

Property	Description
Type	Type of the message: - **Inquiry Request**: A message to retrieve information out of the backend system during the initial load phase or in case of a detected gap. - **Inquiry Response**: A response message to an Inquiry Request. - **Management Request**: A message to send new business information to the backend system to change business objects. - **Management Response**: An response message to a Management Request. - **Broadcast**: The message is sent as a broadcast, initiated by the backend system. One message can have several types.
Roles	User roles that are allowed to send or receive the message. Possible values: Trader, Market Operations, Market Makers, Brokers, Sales, Data Vendor, Settlement Operations and <ALL>
Routing Keys	The routing key(s) used to decide the message queue destination (request messages only).
Response To	Request message to which the response is a reply message (response messages only). If not specified, the message is a broadcast-only message.
Broadcast	Indicates if the response message can be sent as a broadcast (response messages only). If ‘NO’ then the message is only sent as a reply to a request.
Broadcast Routing Keys	Routing keys specified for a broadcast message. These are empty for non-broadcast responses (response messages only). Broadcast routing keys will be deprecated and may be decommissioned in future releases. For documentation purposes they will be replaced by “Broadcast audience” information.
Broadcast Audience	Contains an audience scope for the Broadcast message.

Property	Description
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.

5.1.19 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 Report(=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 when M7 detects an unexpected connection loss of the client. Valid values: • **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.
throttlingUserAction	A	o		Char(16)	An action that will be executed in case of order throttling is applied. Valid values: • **NONE_USER_ORDERS**: No action is executed. • **HIBE_USER_ORDERS**: All orders of this user will be hibernated. • **HIBE_BG_ORDERS**: All orders of balancing group of this user will be hibernated.

XML Tag	Type	m/o	No.	Data Type	Short description
throttlingMemberAction	A	o		Char(15)	An action that will be executed in case of order throttling is applied. Valid values: • **NONE_MBR_ORDERS**: No action is executed. • **HIBE_MBR_ORDERS**: All orders of this member will be hibernated.
authVerificationCode	A	o		Char(255)	Verification code for 2FA
orderbookBatching	A	o		Char(5)	The type of order book batching that will be executed for the user. For more details on the batching process, please refer to the *Order Book Batching* chapter in this document. Valid values: • **SHORT**: Order book batching based on the SHORT time interval (default option). • **LONG**: Order book batching based on the LONG time interval.
receivePblcOrdrBooksDeltaRprt	A	o		Boolean	Flag that indicates if this user should receive PblcOrdrBooksDeltaRprt broadcasts.
receivePblcTradeConfRprt	A	o		Boolean	Flag that indicates if this user should receive PblcTradeConfRprt broadcasts.
orderbookApiVersion	A	o		Char(4)	The version of Order Book API and Private Data API the user wishes to consume. Valid values: • **V6**: V6 API based on XML payload and AMQP protocol. All Order Book messages will be sent to the user's AMQP broadcast queue. This value is used as a default in case the user does not provide any. • **V7**: V7 API based on Protocol Buffers payload and WebSocket protocol. Token will be sent within UsrRprt and user can subscribe for Order Book messages in the WebSocket channel. • **BOTH**: V6 and V7 at the same time. Note that this option might be rejected if M7's broadcast capacity gets saturated.
privateDataApiVersion	A	o		Char(4)	The version of Order Book API and Private Data API the user wishes to consume. Valid values: • **V6**: V6 API based on XML payload and AMQP protocol. All Order Book messages will be sent to the user's AMQP broadcast queue. This value is used as a default in case the user does not provide any. • **V7**: V7 API based on Protocol Buffers payload and WebSocket protocol. Token will be sent within UsrRprt and user can subscribe for Order Book messages in the WebSocket channel. • **BOTH**: V6 and V7 at the same time. Note that this option might be rejected if M7's broadcast capacity gets saturated.
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 *DFS180*).

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.

6.1.2 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 *DFS180*).
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 txt=INACTIVITY.
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 *DFS180*).
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 *DFS180*).
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.

XML Tag	Type	m/o	No.	Data Type	Short description
contractStoreTimeInDays	A	m		Integer	The number of days that contracts are available in the system (today included). If the parameter equals 7 for example, the users are able to display their contracts and respective trades from the last 7 days, including today. The contracts, and the related trades are marked for removal from the system at the date and time calculated as (current date and time – contractStoreTimeInDays). Afterwards they are removed in bulk during the next few hours. Therefore some contracts/trades may be visible for several hours longer than e.g. 7 days exactly.
tradePoolStoreTimeInHours	A	o		Integer	The number of hours that trades are available in the system after contract expiration. This value limits the time window when settlement state of the trade can be changed.
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.
maxOrders	A	m		Integer	The maximum number of orders that are allowed to be sent within one order entry/modify message.
capabilities	A	o		Char(∞)	A list of the backend features available. If the feature is not in the list, it is disabled on backend side. Valid values: • **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**: The value is deprecated. • **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. • **LOCATIONS**: Locations are supported by the system.
allowedClearingAcctTypes	A	m		Char(255)	Comma separated valid values for the *clearingAcctType* attribute in e.g. OrdrEntry 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 *DFS180*).
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
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**: OrdrEntry; OrderModify; ModifyAllOrdrs: 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 a 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. If the system detects an error which prevents it from processing of the message (a syntactical error but also if the message content is against the M7 Trading documentation, e.g. if a buy ICB order has a positive peak price delta), it may reply directly with an ErrResp instead, sending it to the private response queue of the client.

If no errors were detected as described in the previous paragraph, M7 immediately acknowledges the request message being forwarded to Core for processing via AckResp message. This acknowledgement confirms successful message sending, not Core receipt. Because the PMI Gateway generates the AckResp, it may be omitted in rare instances (e.g. in case of a PMI Gateway restart), even is the message was successfully sent to the Core.

In rare circumstances, a Core system failover can occur after the PMI Gateway has sent the request and the acknowledgement (AckResp) has been issued. If this happens, the request might not be processed even though the client has already received the AckResp. In such cases, if you do not receive a message containing the results of the requested operation within a reasonable amount of time, please verify the status of your data by submitting the appropriate inquiry request.

Note: Because of new architecture constraints, clOrdrId 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 *DFS180*).

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.

6.1.7 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. OrdrEntry whilst not having a sufficient cash limit).

The Error Response is sent whenever the M7 Trading system determines a business exception has occurred, based on wrong entries in the request.

Error Responses can be sent synchronously or asynchronously (broadcast). The M7 Trading system sends the Error Response in a synchronous way whenever the received XML message contains syntactical errors or if the message content is against the M7 Trading documentation (e.g. if a buy ICB order has a positive peak price delta). In this case, the requests have not been processed by the M7 Core yet. Error Responses are sent by the M7 Core in an asynchronous way if any errors happen during processing of the request (e.g. if an order is submitted for an expired contract). Synchronous Error Responses get sent to the trader's private response queue, asynchronous get sent to the trader's private 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 localisation of the error text can be completed on the client side (the server side provides no localisation 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 *DFS180*).
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.
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.
clOrdrId	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 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). In addition, the password must not contain 3 or more consecutive characters from the user's LoginId (the check is case-insensitive). 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 [3] 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 [4] days before the password expiry, informing the user about the expiration date,
- sends a notification on the day of password expiry, containing a link to the password reset page.

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 *DFS180*).
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.

6.1.9 TotpPwdReq

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

This request can be used by any user to regenerate the time-based one time key.

This request is followed by a TotpPwdResp sent as a response.

XML Tag	Type	m/o	No.	Data Type	Short description
TotpPwdReq	**SE**	**m**	**1**	**Structure**	
currentPwd	A	m		Char(64)	The current 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 *DFS180*).
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.

6.1.10 TotpPwdResp

- **Type**: Inquiry Response
- **Response to**: TotpPwdReq
- **Broadcasted**: No
- **Routing Keys**: -
- **Roles**: All

The TotpPwdResp returns the newly generated secret key which will be used as an input key for the account registration in the user's personal Authenticator (generator of one time passwords).

XML Tag	Type	m/o	No.	Data Type	Short description
TotpPwdResp	**SE**	**m**	**1**	**Structure**	
secret	A	o		Char(64)	Generated secret if enabled. Empty if disabled.
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 *DFS180*).
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.

6.2 Order Entry and Maintenance

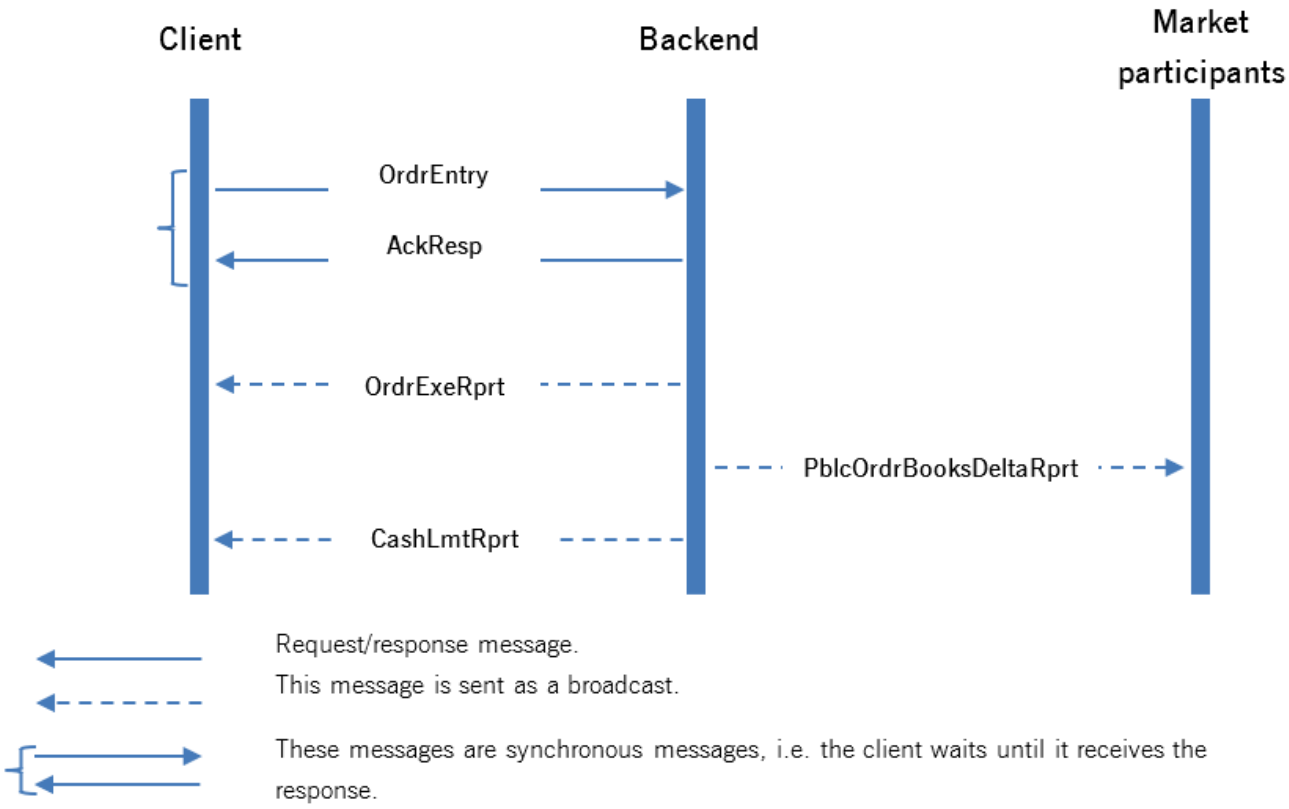
The messages described in this section are used to enter and maintain orders. These messages can be used only by trading participants and are not available for Market Operation users.

6.2.1 OrdrEntry

- **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 [5]:



Note: Actions caused by partial or full execution or invalid order parameters are not part of this diagram.

At least one acknowledgement (AckResp), exactly one Error Response (ErrResp) containing the information on the error(s) 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.

XML Tag	Type	m/o	No.	Data Type	Short description
OrdrEntry	**SE**	**m**	**1**	**Structure**	
listExecInst	A	o		Char(5)	Defines the execution instruction for the whole list of orders: • **NONE**: All orders are treated independently during validation. Only valid orders are processed further, and the processing result is sent in a single OrdrExeRprt, containing a generated LinkId. This value is also to be used if the OrdrEntry message contains only **one order**. • **VALID**: All orders must be valid, meaning they must pass the order validation of the backend system (e.g. the price of the order must be in the price range of the product). If one order does not pass the validation, the full list of submitted orders is rejected. • **LNKD**: Linked orders - the orders in the basket are linked and must be executed either all at once, or the whole list is rejected. A basket with the basket order restriction LNKD can only contain orders with the order restriction FOK. All orders in such a basket must be entered either for local products, or all for remote products (a combination of remote and local orders is not allowed). It is possible to submit orders for different products in a basket. Please check its availability with the System Info Request and the Product Information 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 *DFS180*).
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**	

XML Tag	Type	m/o	No.	Data Type	Short description
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.
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	Peak price delta for Iceberg orders. • 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 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)	Execution restriction of the order. Valid values: • **NON**: No restriction. This is the default setting. • **FOK** (Fill or Kill): The order is immediately fully executed or deleted. • **IOC** (Immediate and cancel): The order is executed immediately to its maximum extent. In the event of a partial execution, the remaining volume is removed from the order book. • **AON** (All or None): The order must be filled completely or not at all. The order stays in the order book until it is executed or removed by the system or user. If the product has an execution restriction of NON, then NON, FOK, IOC are allowed. Only NON is allowed for order type = **I**. AON is allowed for type = **B**. If the product has an execution restriction of AON, then FOK or AON are allowed. IOC is allowed for market sweep for order type = **B**.
txt	A	o		Char(250)	Text entered by the client. The content of this text will not be modified by the backend system. The maximum possible length is 250 characters. Restricted characters: Do not use the special character Unicode U+FFEE in the `Text` field.

XML Tag	Type	m/o	No.	Data Type	Short description
dlvryAreaId	A	o		Char(16)	Defines the delivery area of the order. This is mandatory for exchanges with a multiple delivery area setup.
clOrdrId	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.
type	A	m		Char(1)	Order type: • **O**: Regular limit order. • **B**: User defined block order. • **I**: Iceberg order. • **L**: Balance order. • **S**: Stop limit order. • **E**: On exchange prearranged trade • **N**: Private and confidential trade
dlvryStart	A	o		DateTime	The start of delivery for the underlying contract. Mandatory if contractId is omitted.
dlvryEnd	A	o		DateTime	The end of delivery for the underlying contract. Mandatory if contractId is omitted.
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 *Good for Session* order. Valid values: • **GFS** (Good for trading session): The order rests in the order book until it is either executed, removed by the user or 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. • **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: • **O**: Open position indicator • **C**: Close position indicator

XML Tag	Type	m/o	No.	Data Type	Short description
aot	A	o		Boolean	The indicator whether the order shall be automatically transferred to the corresponding linked contract after the trading in the specific delivery area ends in XBID. Default value: false.
location	A	o		Char(64)	Location within the delivery area. Set only for relevant products.
marketBased	A	o		Boolean	Type of the congestion order to indicate whether it is a market based order (Yes) or a non-market based order (No). Available and mandatory only for products with Locations enabled. Default value: true. If an order for a product with Locations enabled is sent without Market Based attribute, the default value is added by M7 backend.
contractReference	A	o		Char(64)	Reference to a contract between supplier and network operator (e.g. bidding obligation). Available only for products with Locations enabled.
facilityType	A	o		Char(64)	Type of the network connected facility that converts primary energy into electrical energy. Available only for products with Locations enabled.
usageFraction	A	o		Integer	Indicates the mix/max percentage value of usage of the order. Example: For BUY orders it represents the maximum the provider can down regulate their production unit without fully shutting it down. Available only for products with Locations enabled.
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. Deprecated.

6.2.2 OrdrModify

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

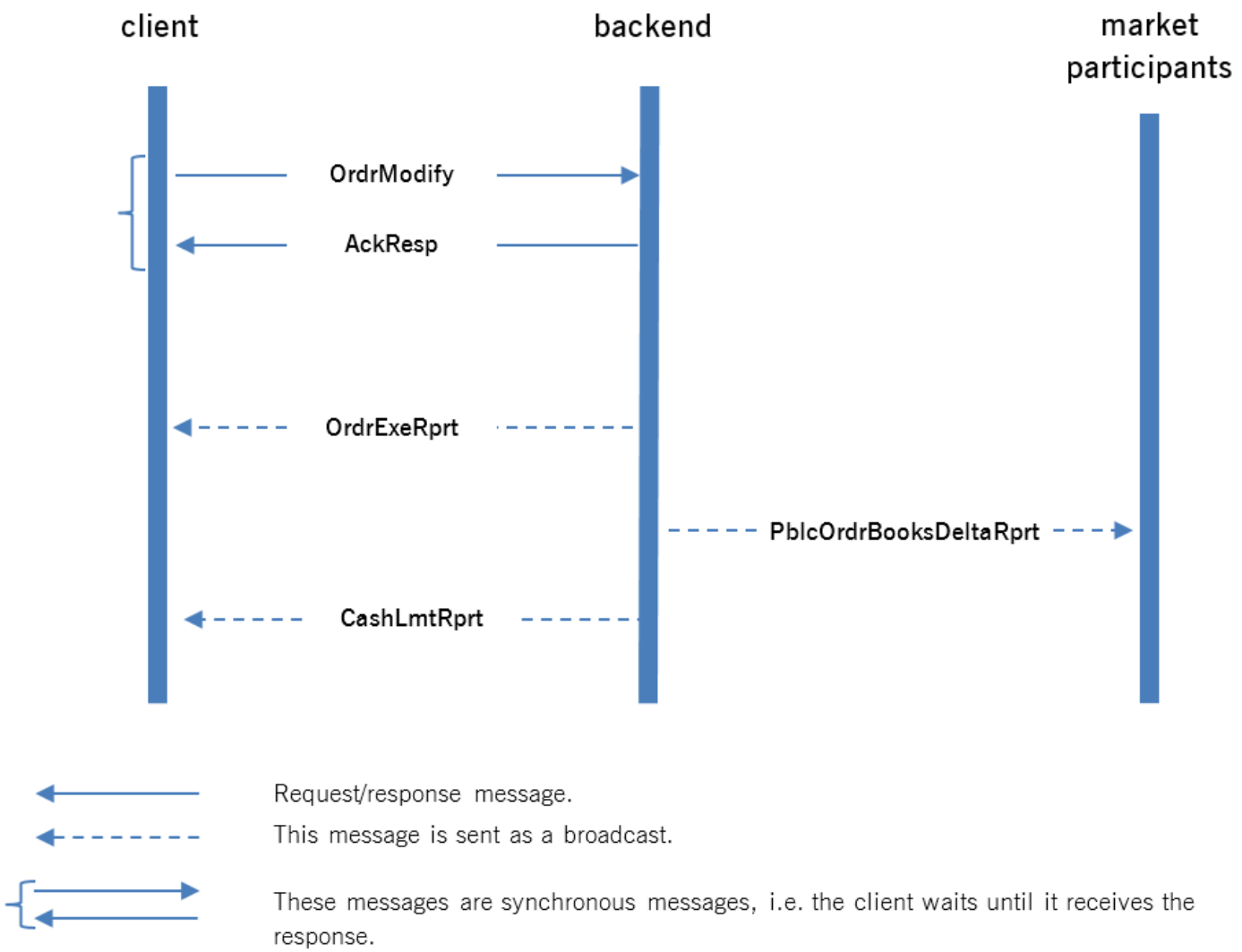
This message is used to modify one or several orders. If the order modification has an impact on the order execution priority, the old order is removed by the system and a new order with a new order id is entered instead. As a result, after an order modification is made which has an impact on the execution priority, an OrdrExeRprt 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 the addition of a new order that was just created in the system to reflect the modification.

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

Note: For remote orders, the behaviour of OrdrExeRprts may be different than described, as it depends on the processing of the OrdrExeRprts by XBID.

The message flow is shown in the below figure:



XML Tag	Type	m/o	No.	Data Type	Short description
OrdrModify	**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. • **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.

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 *DFS180*).
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. • 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.

XML Tag	Type	m/o	No.	Data Type	Short description
ordrExeRestriction	A	o		Char(3)	Execution restriction of the order. Valid values: • **FOK** (Fill or Kill): The order is immediately fully executed or deleted. • **IOC** (Immediate and cancel): The order is executed immediately to its maximum extent. In the case of a partial execution, the remaining volume is removed from the order book. • **AON** (All or None): The order must be filled completely or not at all. The order stays in the order book until it is executed or removed by the system or user. This execution restriction can be used only in combination with User Defined Block Orders. • **NON**: Any restriction. If the product has an execution restriction code of NON, then NON, FOK,IOC are allowed. If the product has an execution restriction code of AON, then FOK or AON are allowed.
txt	A	o		Char(250)	Text entered by the client. This text will not be modified by the backend.
type	A	m		Char(1)	Order type: • **O**: Regular limit order. • **B**: User defined block order. • **I**: Iceberg order. • **L**: Balance order. • **S**: Stop limit order. • **E**: On exchange prearranged trade • **N**: Private and confidential trade
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 *Good for Session* order. Valid values: • **GFS** (Good for trading session): The order rests in the order book until it is either executed, removed by the user or 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.

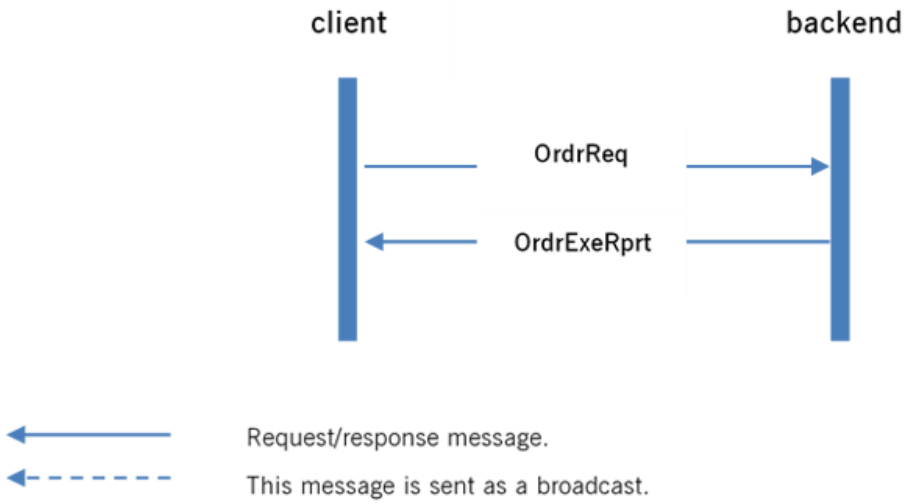
XML Tag	Type	m/o	No.	Data Type	Short description
openCloseInd	A	o		Char(1)	This is mandatory for Futures and Cross product spreads. For other Commodities it is not used. Valid values: • **O**: Open position indicator • **C**: Close position indicator
aot	A	o		Boolean	Indicates whether the order has been automatically transferred to the corresponding linked contract after the trading in the specific delivery area ended in XBID.
location	A	o		Char(64)	Location within the delivery area. Should be only set for products with locations enabled.
marketBased	A	o		Boolean	Type of the congestion order to indicate whether it is a market based order (Yes) or a non-market based order (No). Available and mandatory only for products with Locations enabled. Default value: true. If an order for a product with Locations enabled is sent without Market Based attribute, the default value is added by M7 backend.
contractReference	A	o		Char(64)	Reference to a contract between supplier and network operator (e.g. bidding obligation). Available only for products with Locations enabled.
facilityType	A	o		Char(64)	Type of the network connected facility that converts primary energy into electrical energy. Available only for products with Locations enabled.
usageFraction	A	o		Integer	Indicates the mix/max percentage value of usage of the order. Example: For BUY orders it represents the maximum the provider can down regulate their production unit without fully shutting it down. Available only for products with Locations enabled.
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. Deprecated.

6.2.3 OrdrReq

- **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
OrdrReq	**SE**	**m**	**1**	**Structure**	
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 *DFS180*).
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 OrdrExeRprt

- **Type**: Management Response; Broadcast
- **Response to**: OrdrEntry; OrdrModify; OrdrReq; ModifyAllOrdrs; (sent to the private response queue)

- **Broadcast**: Yes
- **Routing Keys**: `[schema-version].bg.[acctId]`, `[schema-version].loc.[prodName].[dlvryAreaId]`
- **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

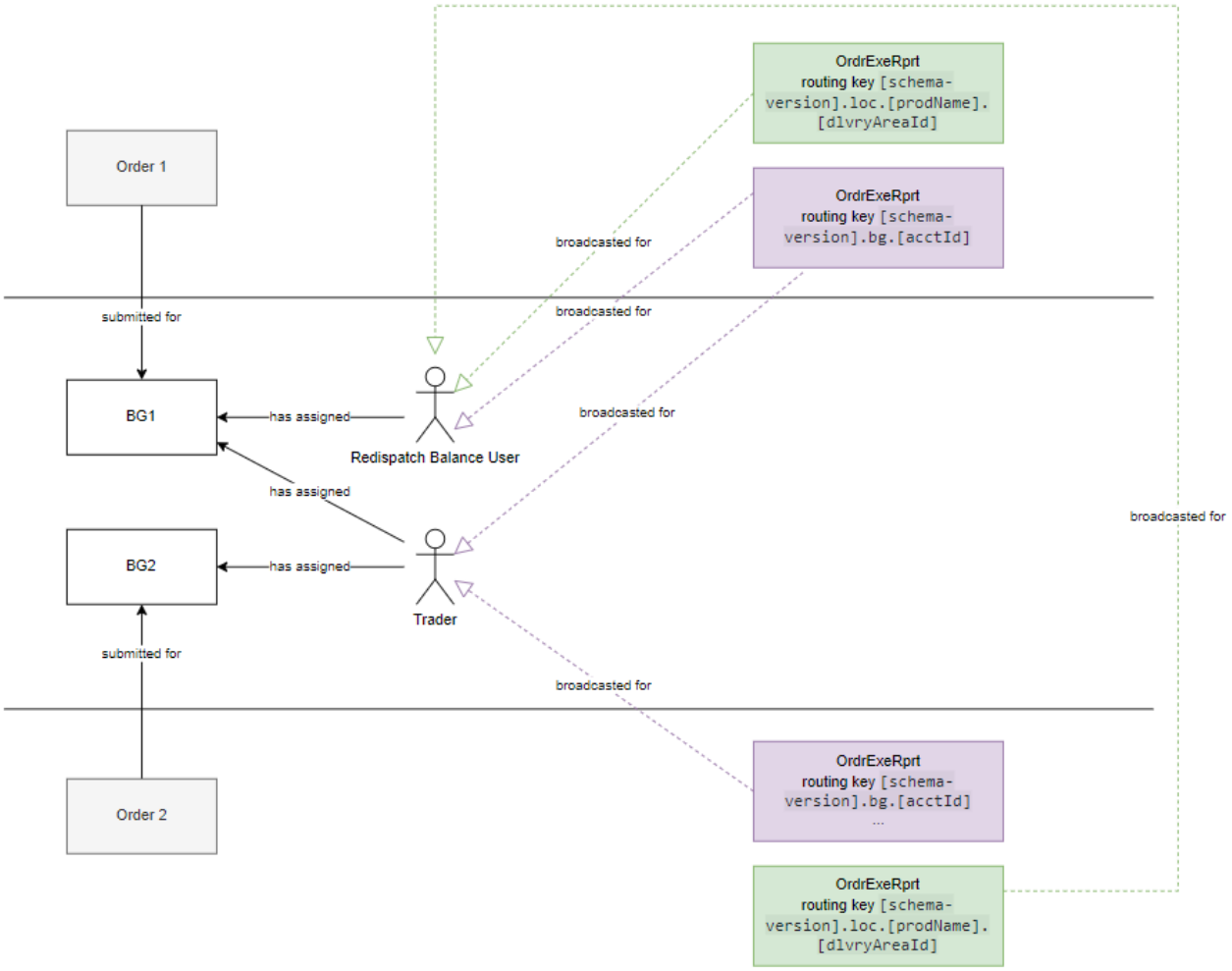
The Order Execution Report is sent in the following cases:

- 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.

If Locations are enabled for the product (i.e. product matcher type is *Energy Location-Price-Time* or *Energy Location-AON*), the OrdrExeRprt for the product's orders is broadcasted also to Redispatch Balance Users, using the routing key `[schema-version].loc.[prodName].[dlvryAreaId]`. In such case, Redispatch Balance Users are notified about orders submitted for assigned products and assigned delivery areas, regardless of the assignment for balancing group. That is, Redispatch Balance Users receive broadcasts for all existing balancing groups. If an order is submitted for a balancing group assigned to the Redispatch Balance User, they will receive two OrdrExeRprts with the same revisionNo, one for each routing key. Redispatch Balance Users should ignore the second broadcast with the same revisionNo. This situation is depicted in the schema below.



XML Tag	Type	m/o	No.	Data Type	Short description
OrdrExeRprt	**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 *DFS180*).
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.
OrdrList	**SE**	**o**	**0..1**	**Structure**	
Ordr	**SE**	**o**	**0..n**	**Structure**	
ordrId	A	m		Long	Order Id as returned by the backend system.
initialOrdrId	A	m		Long	In the event of an order modification, this value contains the Id of the first order in the modification chain.
parentOrdrId	A	o		Long	In the event of an order modification, this field contains the Id of the modified order.
clearingAcctType	A	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: • **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.
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).

XML Tag	Type	m/o	No.	Data Type	Short description
ordrExeRestriction	A	o		Char(3)	Execution restriction of the order. Valid values: • **NON**: No restriction. This is the default. • **FOK** (Fill or Kill): The order is immediately fully executed or deleted. • **IOC** (Immediate and cancel): The order is executed immediately to its maximum extend. In case of a partial execution, the remaining volume is removed from the order book. • **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.
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.
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.
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: • **O**: Regular limit order. • **B**: User defined block order. • **I**: Iceberg order. • **L**: Balance order. • **S**: Stop limit order. • **E**: On exchange prearranged trade • **N**: Private and confidential trade
state	A	m		Char(4)	The current state of the order in the system. Valid values: • **HIBE**: The order is entered into the backend system but is not exposed to the market. • **ACTI**: The order is entered and is immediately exposed to the market for execution. • **IACT**: The order is deleted. • **UKNW**: The order state is unknown (eg. LTS is disconnected from XBID).
aggressorIndicator	A	o		Char(1)	Indicates whether the executed order was a trade aggressor or trade originator. The field will only be present in case the Order Execution Report was triggered by a partial or full execution. • **Y** - Trade aggressor • **N** - Trade originator • **U** - Unknown, for executed orders of remote products and data before migration
usrCode	A	m		Char(6)	The user code of the user performing the last successful action on the order.

XML Tag	Type	m/o	No.	Data Type	Short description
revisionNo	A	m		Long	When an order is entered, the revision is set to 1. The revision is increased by 1 in case of a partial execution, hibernation or a modification without an execution priority change. In case of a modification with an execution priority change, the revision number for the deleted order (UDEL action) is increased by 1 and the revision number for the newly entered order is reset to 1 (UADD action). The behaviour is the same for local and remote orders.
timestmp	A	m		DateTime	The timestamp of the order entry as determined by the backend. This timestamp determines the execution priority in case of identical limit prices.
validityDate	A	o		DateTime	This field is mandatory in the event that validityRes equals GTD. It is used to define the date until which the order is valid. The remaining part of the order will be removed from the order book after this point in time. 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 *Good for Session* order. Valid values: • **GFS** (Good for trading session): The order rests in the order book until it is either executed, removed by the user or 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 *FOK* or *IOC*.

XML Tag	Type	m/o	No.	Data Type	Short description
action	A	m		Char(255)	Lists the action code. Valid values: • **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.
lastUpdateUsrInfo	A	m		Char(255)	Information (concatenation of accountId and usrCode) about the user who last updated the order, either on their own or on behalf. If the *action* is any of the system actions (FEXE, SADD etc.), then the update has been created by the system and the lastUpdateUsrInfo contains the system user (*7777777777777777SYSTEM*).
openCloseInd	A	o		Char(1)	Mandatory for Futures and Cross product spreads. For other Commodities the value is not used. Valid values: • **O**: Open position indicator • **C**: Close position indicator
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.
remoteOrdrId	A	o		Long	The Order Id as returned by the remote backend system (i.e. XBID SOB)
lastUpdateTm	A	m		DateTime	The timestamp of the last modification of the order object in the back-end. Is also updated for modifications which do not affect the execution priority (like the *text* field).
remoteLastUpdateTm	A	o		DateTime	The timestamp of the last modification of the order object in the remote backend system.

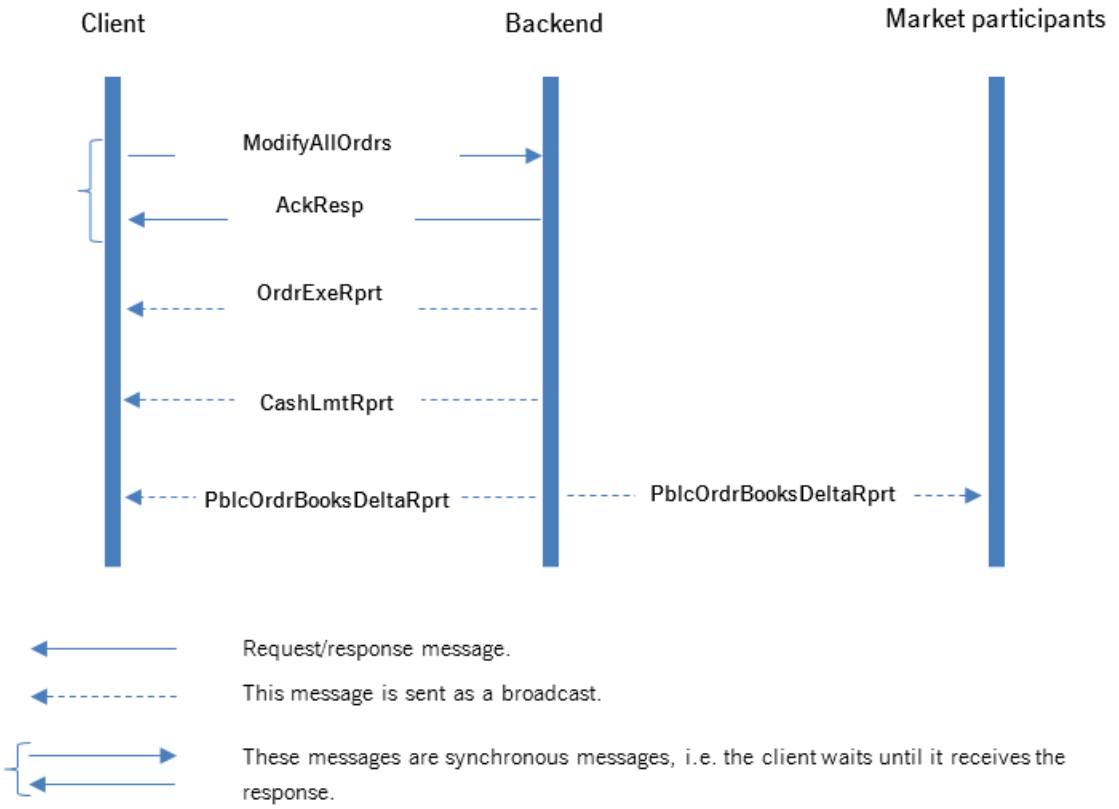
XML Tag	Type	m/o	No.	Data Type	Short description
preAotId	A	o		Long	Local order ID of the remote order that this order originated from during AOT. The field is populated if and only if this is a local order transferred from a remote order during AOT.
aot	A	o		Boolean	Indicates whether the order has been automatically transferred to the corresponding linked contract after the trading in the specific delivery area ended in XBID Default value: false.
location	A	o		Char(64)	Location within the delivery area. Set only for relevant products.
marketBased	A	o		Boolean	Type of the congestion order to indicate whether it is a market based order (Yes) or a non-market based order (No). Available and mandatory only for products with Locations enabled. Default value: true. If an order for a product with Locations enabled is sent without Market Based attribute, the default value is added by M7 backend.
contractReference	A	o		Char(64)	Reference to a contract between supplier and network operator (e.g. bidding obligation). Available only for products with Locations enabled.
facilityType	A	o		Char(64)	Type of the network connected facility that converts primary energy into electrical energy. Available only for products with Locations enabled.
usageFraction	A	o		Integer	Indicates the mix/max percentage value of usage of the order. Example: For BUY orders it represents the maximum the provider can down regulate their production unit without fully shutting it down. Available only for products with Locations enabled.
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. Deprecated.

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**	
mbrId	A	o		Char(5)	Unique identifier of a member. *Only one of attributes mbrId, usrId or accId must be provided.*
usrId	A	o		Integer	Unique identifier of a user *Only one of attributes mbrId, usrId or accId must be provided.*
acctId	A	o		Char(32)	Unique identifier of an account. *Only one of attributes mbrId, usrId or accId must be provided.*
dlvryAreaId	A	o		Char(16)	Orders for the given usrId and list of delivery areas in dlvryAreaId will be Deactivated or Deleted. This element can only be supplied when usrId is provided in the message. If left out, all delivery areas assigned to usrId are affected.

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 (mbrId 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. • **DELE**: Deletes all orders. Not available for modifications on member level (mbrId 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 *DFS180*).
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 *DFS180*).
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 OrdrLmtRprt

- **Type**: Inquiry Response; Broadcast
- **Response to**: OrdrLmtReq (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 OrdrLmtResp is returned as a response to the OrdrLmtReq or as a broadcast as a result of an event that changes an Order Limit. The message lists all the Order entry limits of the inquired member. These limits are then checked at every order entry process for appropriate member.

XML Tag	Type	m/o	No.	Data Type	Short description
OrdrLmtRprt	**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 *DFS180*).

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.
OrdrLmtList	**SE**	**m**	**1**	**Structure**	**List of order limits**
OrdrLmt	**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.

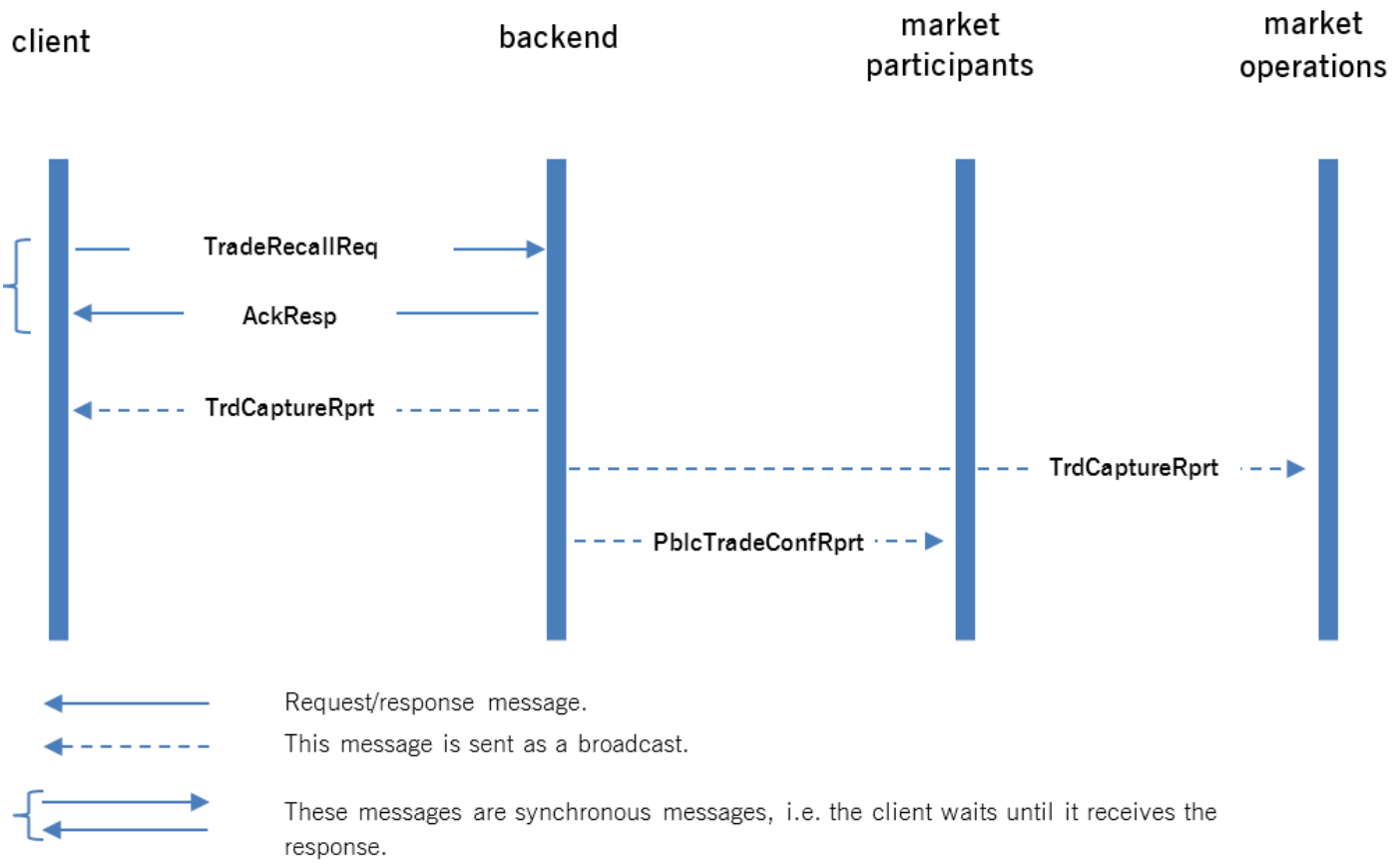
6.3 Trade Maintenance

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

XML Tag	Type	m/o	No.	Data Type	Short description
TradeRecallReq	**SE**	**m**	**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 *DFS180*).
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 Order Book Batching

6.4.2.1 Background

The M7 Trading system has been designed to operate in the 24/7 mode.

Market activity is increasing continuously and one of the impacted areas is the order books. During periods of heightened market activity called "peaks", the system must be able to process and distribute a significant number of order book delta (OBK delta) messages in a timely manner.

It is important to highlight that the number of OBK delta messages to be processed and distributed does not only include the messages produced by the M7 LTS itself. It also includes the OBK delta messages received through the XBID solution due to the trading activity of other power exchanges.

6.4.2.2 Order Book Batching

To smoothen the heightened trading activity, M7 Trading application features Order Book Batching (OBK Batching) mechanism which enables the system to optimise the way the OBK delta messages are built and the frequency of their distribution to M7 users.

When OBK Batching is enabled, the system does not broadcast OBK deltas in real time. Instead, it accumulates all order book changes within a defined batch interval and broadcasts a single OBK delta message at the *End of the Batch interval* (please see more about **End of a Batch** in Batching Mechanism). This delta reflects the net effect of all changes, not the sequence of intermediate events. Users always receive one batched OBK delta message per Routing Key.

The system supports two Batching intervals called "SHORT" and "LONG". When both intervals are configured, each user (on session level) will be getting OBK delta messages batched either based on the LONG, or the SHORT Batch interval. For more details, please refer to the following chapters, especially OBK Delta Messages Flow. Note:

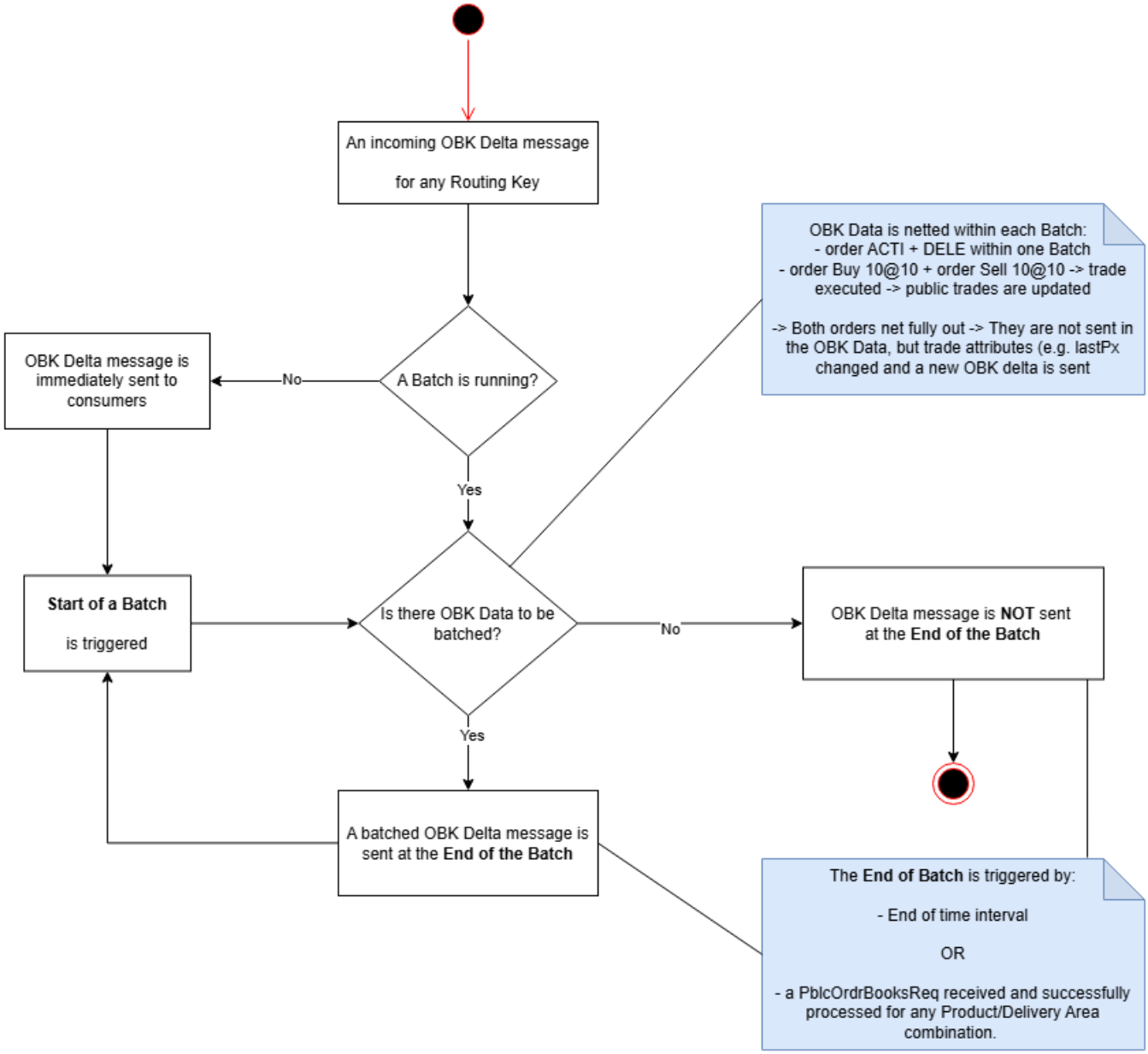
The batching mechanism is available only for V6 OBK delta messages.

6.4.2.3 Batching Mechanism

The OBK Batching mechanism works as follows:

- the **Start of a Batch** is triggered by an incoming OBK delta message for any Routing Key. The *Start of Batch* applies to **all Routing Keys** of the logged-in user.
    - when a Batch is not running and OBK delta message is produced for any Routing Key (= *Product* X *Delivery Area* combination), then this OBK delta message is sent immediately to M7 users. At the same time, this OBK delta triggers the Start of the Batch.
    - when the Batch is running and there is OBK data to be batched, then a batched OBK delta message is sent at the *End of the Batch* (see below). At the same time, this triggers the Start of a new Batch.
    - when the Batch is running and there is no OBK data to be batched, then **no** OBK delta message is sent at the *End of the Batch* (see below). As there is no OBK delta sent out, there is **no** trigger to Start a new Batch. **Note:** the heartbeat message also triggers the Start of a Batch; however, this is not operationally relevant because heartbeats occur at a fixed 5-second interval.
- the **End of a Batch** is triggered by
    - the end of time interval configured for the Batch (LONG or SHORT), OR
    - a PblcOrdrBooksReq received and successfully processed for any Routing Key, i.e. (*Product* X *Delivery Area* combination).

The OBK Batching mechanism is depicted by the following diagram:



Notes:

- Both LONG and SHORT running batches are stopped whenever a PblcOrdrBooksReq is received and successfully processed.
- When both SHORT and LONG Batch intervals are configured, the system produces two independent OBK delta flows. Each user's session gets either SHORT or LONG OBK delta flow, depending on the client used (ComTrader vs. API client) and/or the user's election in the LoginReq. It is possible that, at one moment there will be a running Batch for the LONG interval, but no running Batch for the SHORT interval. This is shown in the *Example 6* in Examples - SHORT and LONG Batching. **Note:** from an implementation perspective, the same code is used for both SHORT and LONG batching modes. As a result, the batching behavior at the end of a batch is identical for both.

6.4.2.4 OBK Delta Messages Flow

The OBK Batching for LONG Batch interval is always turned on in the M7 system. It is applied to all users who connect via ComTrader as well as users who sent the Login Request with *orderbookBatching* parameter set to the value "LONG".

The OBK Batching for SHORT Batch interval is available in the M7 system. It can be turned on when required, and thus, it will be applied to users who sent the Login Request with *orderbookBatching* parameter set to the value “SHORT”.

If a user connects via API and does not provide the *orderbookBatching* parameter in the Login Request, then the OBK batching is applied based on the default configuration of the environment. If a default value was not configured, then “LONG” will be applied.

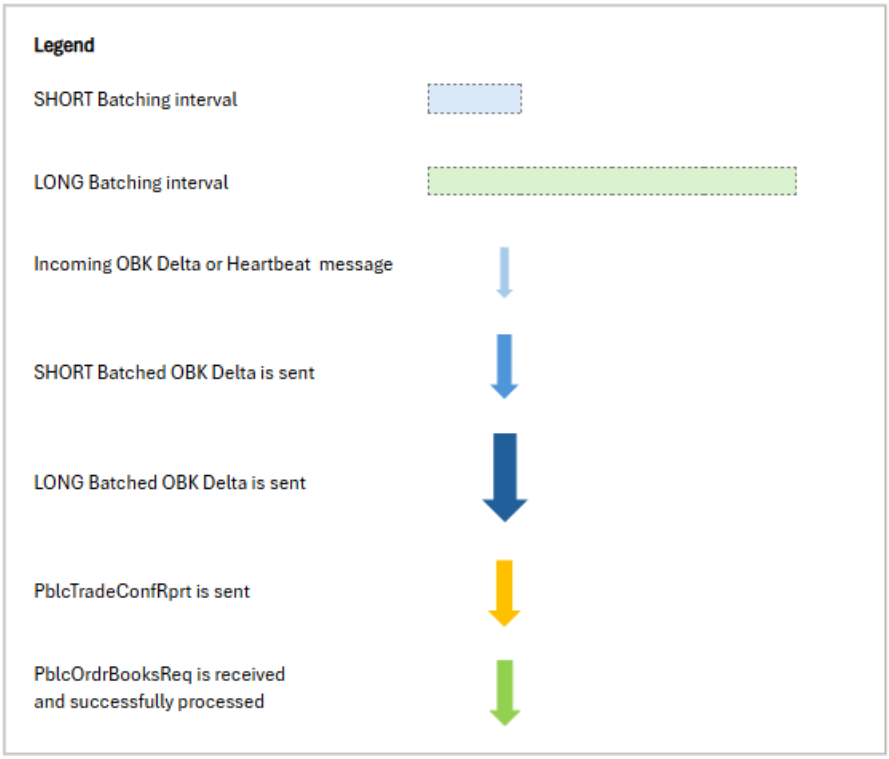
System Configuration	User connects via	LoginReq	Resulting OBK Delta flow received by user
SHORT Batching Disabled	ComTrader	—	OBK delta messages batched with LONG Batch interval
SHORT Batching Disabled	API	orderbookBatching = LONG	OBK delta messages batched with LONG Batch interval
SHORT Batching Disabled	API	orderbookBatching = SHORT	No batching is applied on OBK delta messages
SHORT Batching Disabled	API	orderbookBatching not provided	OBK delta messages batched based on default `orderbookBatching` configuration. If the default configuration is set to SHORT, no batching is applied on OBK delta messages
SHORT Batching Enabled	ComTrader	—	OBK delta messages batched with LONG Batch interval
SHORT Batching Enabled	API	orderbookBatching = LONG	OBK delta messages batched with LONG Batch interval
SHORT Batching Enabled	API	orderbookBatching = SHORT	OBK delta messages batched with SHORT Batch interval
SHORT Batching Enabled	API	orderbookBatching not provided	OBK delta messages batched based on default `orderbookBatching` configuration

Currently, Short batching is disabled. All OBK batching configurations including the duration of SHORT and LONG Batch intervals are exchange-specific. The change of the Short Batching configuration requires a failover of the M7 Core component in a rolling fashion.

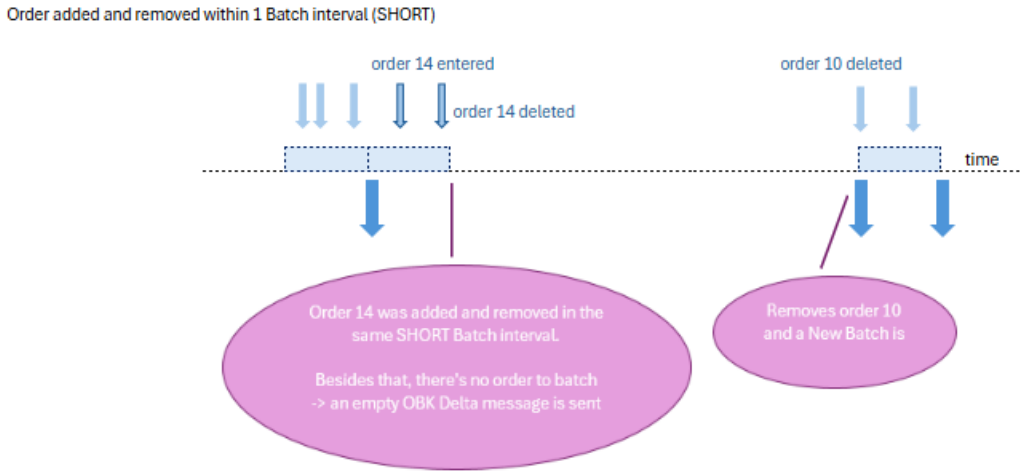
6.4.2.5 Examples - SHORT and LONG Batching

The following examples show the SHORT and LONG OBK batching behaviour.

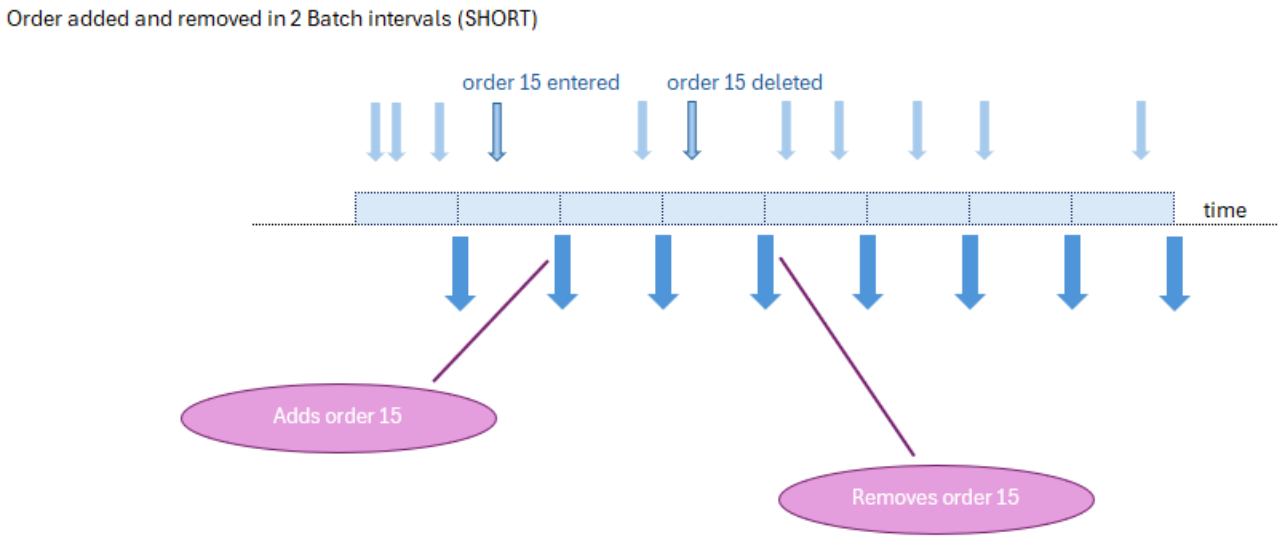
Please see below the legend that is shared across all examples.



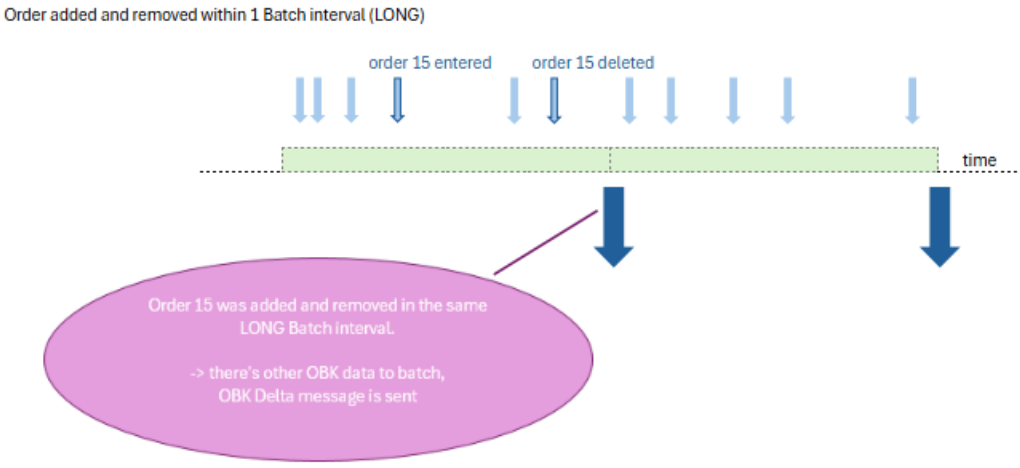
Example 1 - Order added and removed within 1 SHORT Batch interval



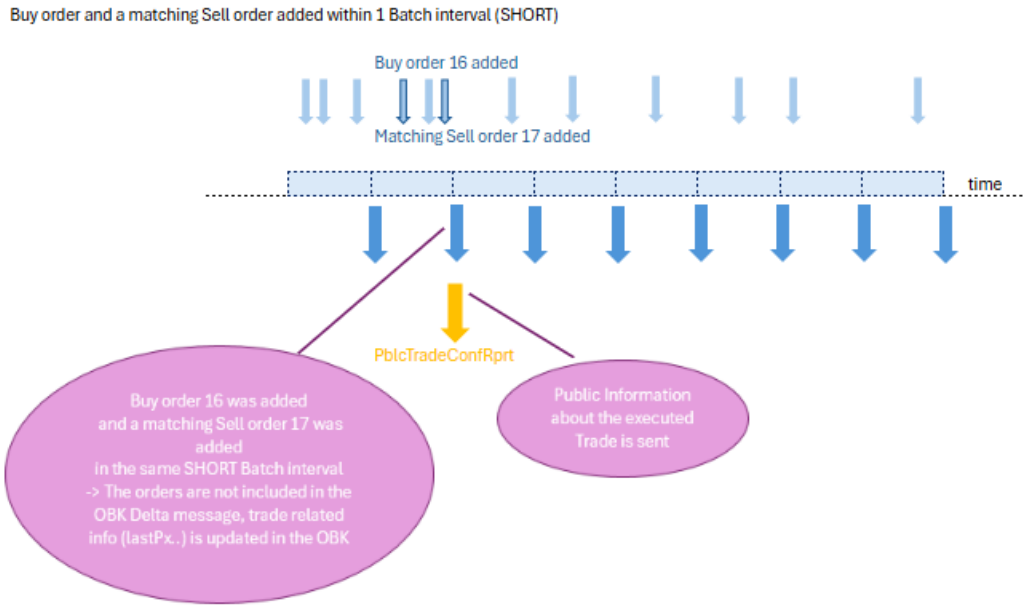
Example 2 - Order added and removed in 2 SHORT Batch intervals



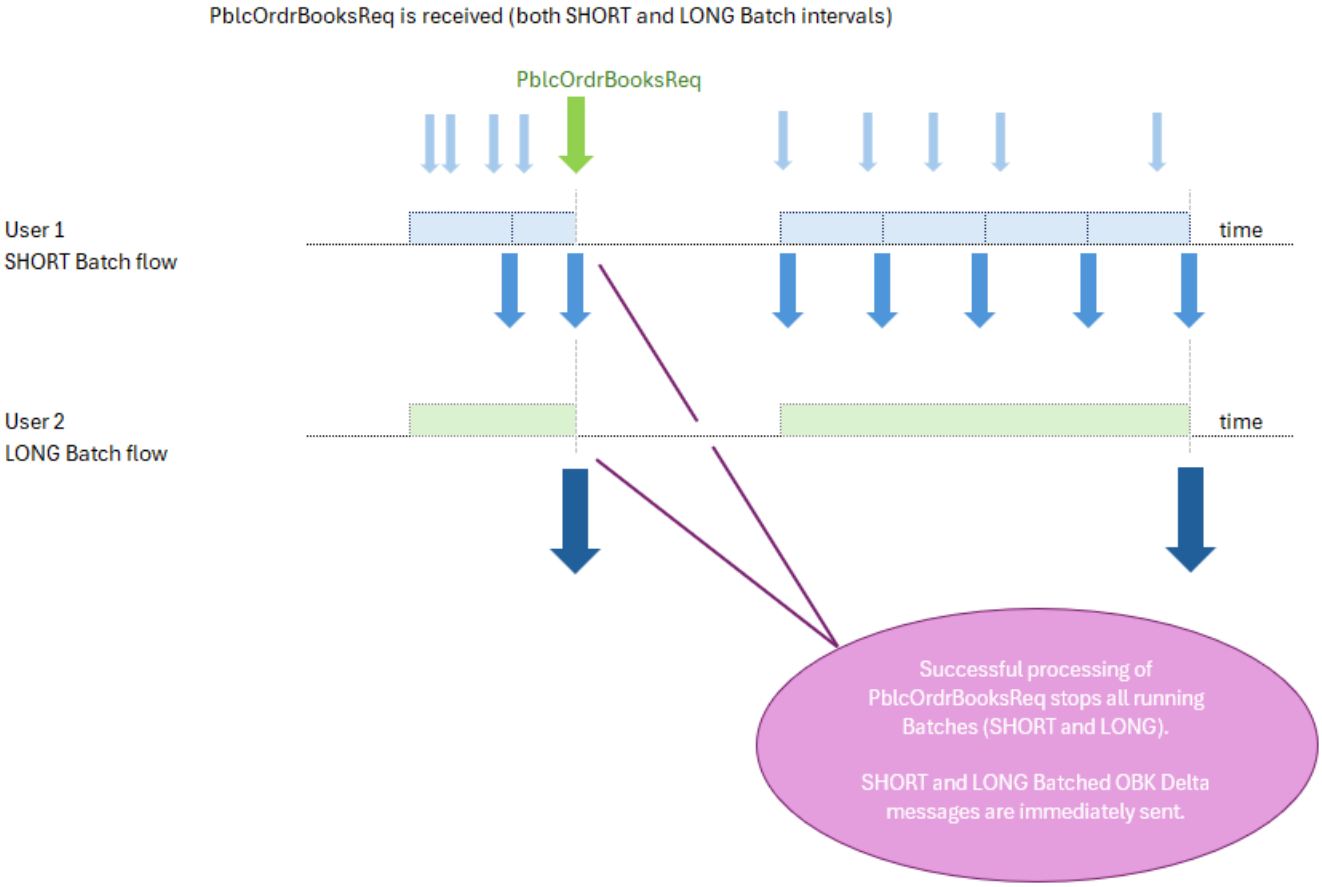
Example 3 - Order added and removed within 1 LONG Batch interval



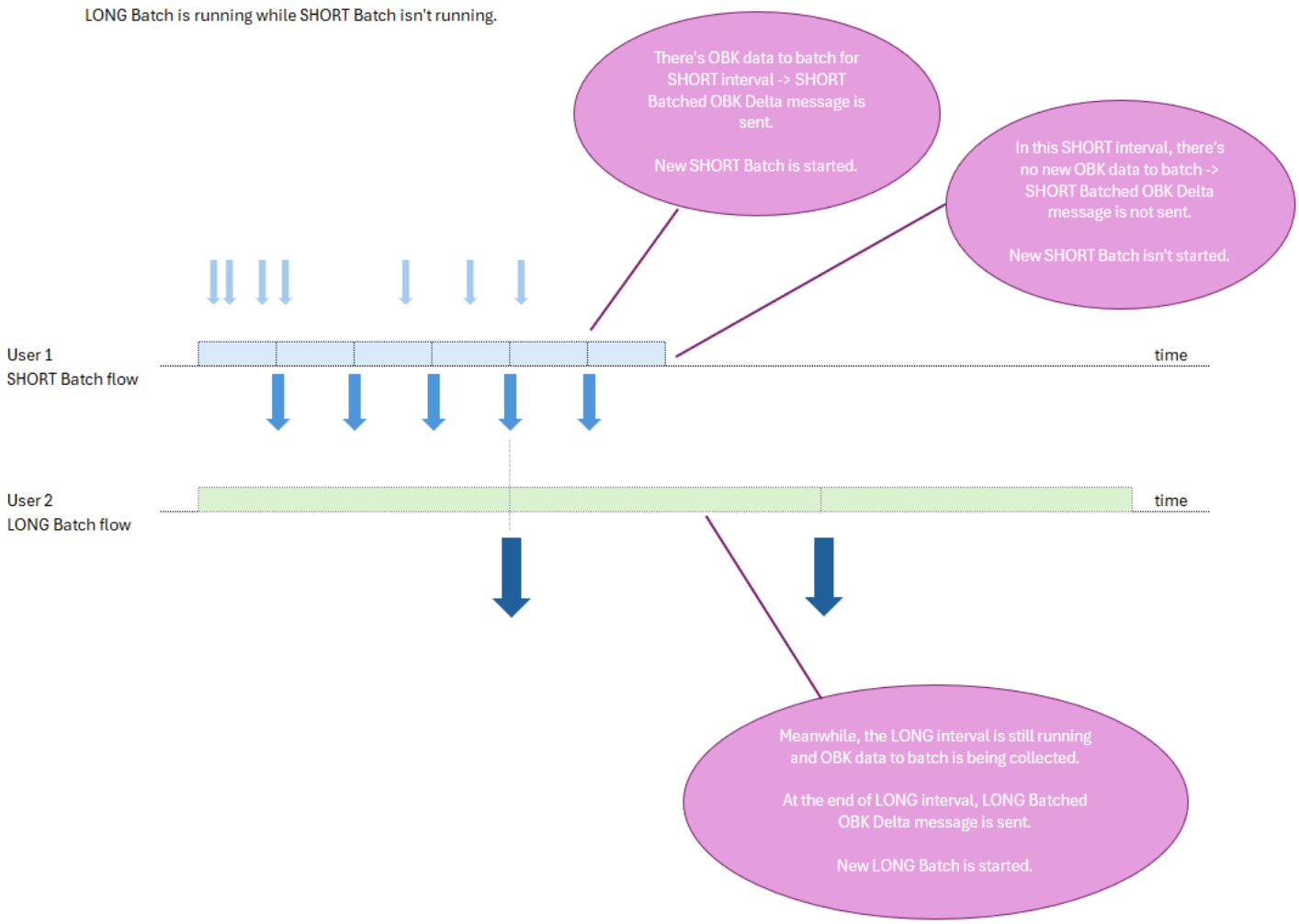
Example 4 - Buy order and matching Sell order added within 1 SHORT Batch interval



Example 5 - PblcOrdrBooksReq is received - SHORT and LONG Batch interval immediately stop



Example 6 - SHORT and LONG Batch intervals run independently

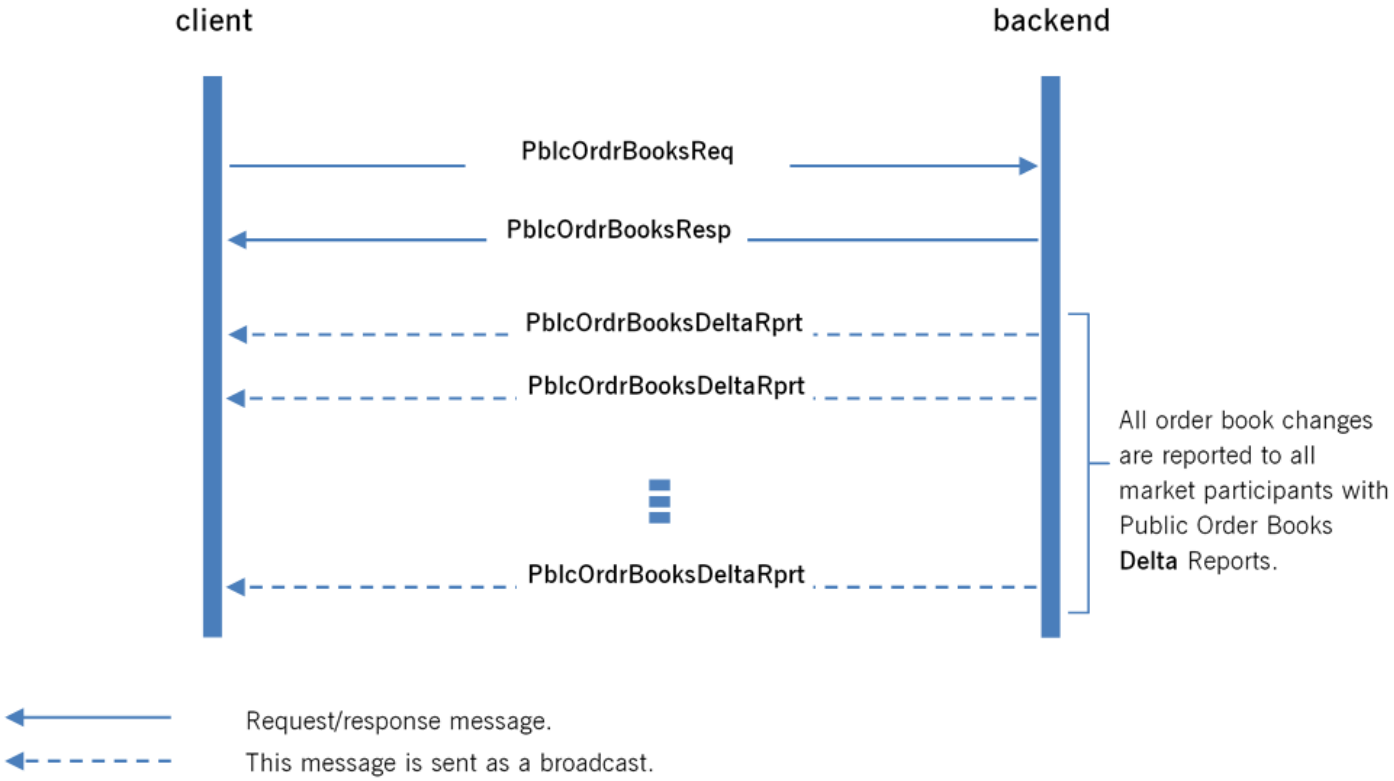


6.4.3 PblcOrdrBooksReq

- **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. M7 filters out orders from XBID for remote contracts that cannot be traded locally. 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: • **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: • **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 *DFS180*).
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.
dlvryAreaId	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.

6.4.4 PblcOrdrBooksResp

- **Type**: Inquiry Response
- **Response to**: PblcOrdrBooksReq (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. In the case of remote products, M7 filters out orders using the messages that are broadcast by XBID for contracts that are not in-a trading phase according to the local schedule.

Subsequent changes to the public order books as a result of an order entry, modification, deletion or execution (including any order book relevant change received from XBID) 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
PblcOrdrBooksResp	**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 *DFS180*).
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	Total quantity of individual orders that were matched in a trade on selected contract and delivery area. If the trade is created from 2 orders on the same delivery area, then TQty includes quantity of both orders. For remote trades TQty is provided by XBID.
lastTradeTime	A	o		DateTime	Timestamp of the last execution.
pxDir	A	o		Integer	Defines the direction of the price movement. Valid values: • **-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. At the same time, the first *revisionNo* received can be higher than 1, and *revisionNo* can also rise by more than 1 due to internal batching in the core.
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. If the order was entered for a remote product, the field contains: • the local Order ID (not the one from the XBID system) for the orders of own PX, or • the remote Order ID (the one from the XBID system) for the orders of other PXs.
qty	A	m		Integer	The quantity of the order which is exposed in that delivery area. This value may be different for one order depending on the observed delivery area and the available cross border capacity.
px	A	m		Long	The limit price of the order.
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: • **O**: Regular limit order(default value) • **B**: User defined block order. • **L**: Balance order.
openCloseInd	A	o		Char(1)	Mandatory for Futures and Cross product spreads. For other Commodities this value is not used. Valid values: • **O**: Open position indicator • **C**: Close position indicator
ClgHse	**SE**	**o**	**0..n**	**Structure**	
clgHseCode	A	m		Char(255)	Clearing House Code. Deprecated.
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: • 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.

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: • **O**: Regular limit order(default value) • **B**: User defined block order. • **L**: Balance order.
openCloseInd	A	o		Char(1)	Mandatory for Futures and Cross product spreads. For other Commodities this value is not used. Valid values: • **O**: Open position indicator • **C**: Close position indicator
ClgHse	**SE**	**o**	**0..n**	**Structure**	
clgHseCode	A	m		Char(255)	Clearing House Code. Deprecated.

6.4.5 PblcOrdrBooksDeltaRprt

- **Type**: Broadcast
- **Response to**: n/a
- **Broadcast**: Yes
- **Routing Keys**: `[schema-version].prddlvr.[prodName].[dlvryAreaId]` , `[schema-version].prddlvr.[prodName].[dlvryAreaId].[routingKeySuffix]` , `[schema-version].prddlvr.batched.[prodName].[dlvryAreaId]`
- **Broadcast audience**: All users with the assignment of a particular product and delivery area.
- **Roles**: [All]

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

It contains a list of orders that have been added to the market, or changed as a result of the above mentioned actions. A quantity of 0 indicates that the order has to be removed from the order book.[6]

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

If the Orderbook batcher is enabled, Public Order Books Delta Reports will be sent without `correlation-id` in the AMQP Message Properties and the routing key will depend on the selected interval (SHORT or LONG). While for the LONG interval, the `[schemaversion].prddlvr.batched.[prodName].[dlvryAreaId]` routing key will be used, in case of the SHORT interval or when the Orderbook batcher is disabled, the routing key `[schema-version].prddlvr.[prodName].[dlvryAreaId]` will be used.

If the Orderbook batcher is enabled and Routing key split feature is enabled, then the `[schema-version].prddlvr.[prodName].[dlvryAreaId].[routingKeySuffix]` routing key will be used for SHORT interval. Routing key split can be enabled on environment level only. Routing key split is a performance improvement measure. It does not apply to environment(s) where Orderbook batcher is disabled. It does not apply to LONG interval where Orderbook batcher is enabled. Routing Key Suffix is a

single digit number. Suffix is assigned to user during log in. Suffix is same value until log out. Suffix assigned might change with each login and as such it should be ignored.

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

XML Tag	Type	m/o	No.	Data Type	Short description
PblcOrdrBooksDeltaRprt	**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 *DFS180*).
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	Total quantity of individual orders that were matched in a trade on selected contract and delivery area. If the trade is created from 2 orders on the same delivery area, then TQty includes quantity of both orders. For remote trades TQty is provided by XBID.
lastTradeTime	A	o		DateTime	Timestamp of the last execution.
pxDir	A	o		Integer	Defines the direction of the price movement. Valid values: • **-1**: Price has decreased • **0**: Price is unchanged • **1**: Price has increased

XML Tag	Type	m/o	No.	Data Type	Short description
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. At the same time, the first *revisionNo* received can be higher than 1, and *revisionNo* can also rise by more than 1 due to internal batching in the core.
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. If the order was entered for a remote product, the field contains: • the local Order ID (not the one from the XBID system) for the orders of own PX, or • the remote Order ID (the one from the XBID system) for the orders of other PXs.
qty	A	m		Integer	The quantity of the order which is exposed in that delivery area. This value may be different for one order depending on the observed delivery area and the available cross border capacity.
px	A	m		Long	The limit price of the order.
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: • **O**: Regular limit order(default value) • **B**: User defined block order. • **L**: Balance order.
openCloseInd	A	o		Char(1)	Mandatory for Futures and Cross product spreads. For other Commodities this value is not used. Valid values: • **O**: Open position indicator • **C**: Close position indicator
ClgHse	**SE**	**o**	**0..n**	**Structure**	

XML Tag	Type	m/o	No.	Data Type	Short description
clgHseCode	A	m		Char(255)	Clearing House Code. Deprecated.
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: • 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.
ordrType	A	o		Char(1)	The following values can be used: • **O**: Regular limit order(default value) • **B**: User defined block order. • **L**: Balance order.
openCloseInd	A	o		Char(1)	Mandatory for Futures and Cross product spreads. For other Commodities this value is not used. Valid values: • **O**: Open position indicator • **C**: Close position indicator
ClgHse	**SE**	**o**	**0..n**	**Structure**	
clgHseCode	A	m		Char(255)	Clearing House Code. Deprecated.

6.4.6 CashLmtReq

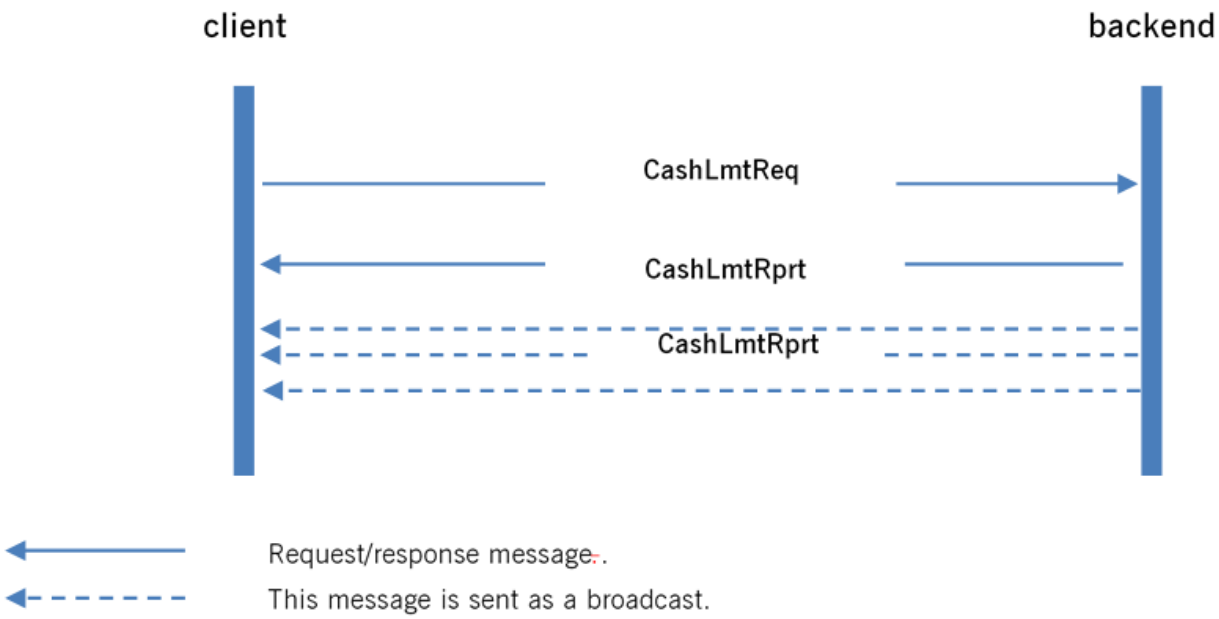
- **Type**: Inquiry Request
- **Routing Keys**: `m7.request.inquiry`
- **Roles**: Trader, Market Operation, Broker, Market Maker, Clearing user
- **Request Limits**: 14/70

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**	
mbrId	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 *DFS180*).
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.

6.4.7 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**	
mbrId	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 *DFS180*).
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.
decShft	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.

XML Tag	Type	m/o	No.	Data Type	Short description
cashLmt	A	m		Long	The cash limit that is available for the member.
decShft	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: • **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.

6.4.8 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.
decShft	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.

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 *DFS180*).
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.

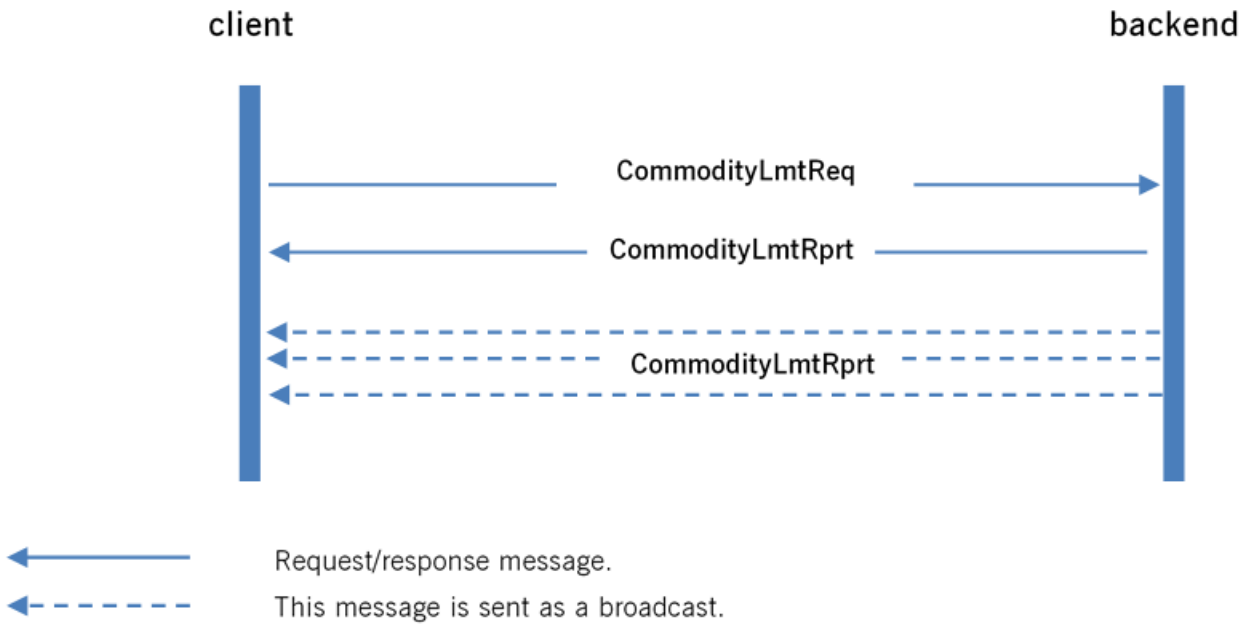
6.4.9 CommodityLmtReq

- **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**	
mbrId	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 *DFS180*).
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.

6.4.10 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 *DFS180*).
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.

6.4.11 MsgReq

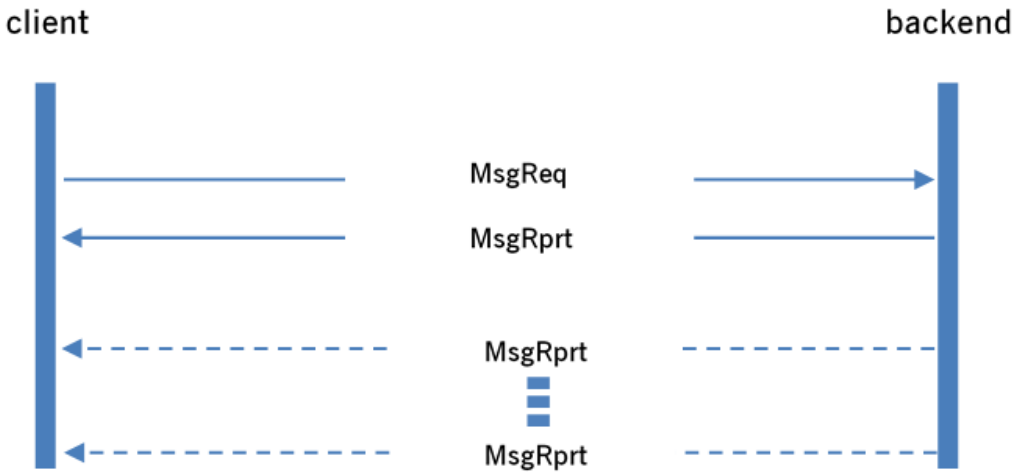
- **Type**: Inquiry Request
- **Routing Keys**: `m7.request.inquiry`
- **Roles**: All
- **Request Limits**: 14/70

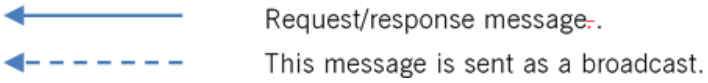
This inquiry message is used to retrieve the public and/or private messages issued by the back-end system in the past.

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.
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: • **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 *DFS180*).
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.

6.4.12 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
- when an order failed to be transferred during an automated order transfer (see *DFS160a*).

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

XML Tag	Type	m/o	No.	Data Type	Short description
MsgRprt	**SE**	**m**	**1**	**Structure**	
truncated	A	o		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 *DFS180*).
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**	
msgId	A	m		Long	The message Id assigned by the back-end system.
type	A	m		Char(7)	Defines the message type. Valid values: • **PUBLIC**: The message is a public message. • **PRIVATE**: The message is a private message.
messageCode	A	m		Integer	Message code of the message.
prod	A	o		Char(255)	Underlying product
acctId	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: • **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.
sellDlvryAreaId	A	o		Char(16)	In case of an order execution, this field contains the delivery area of the sell side.
buyDlvryAreaId	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 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)

6.4.13 TradeCaptureReq

- **Type**: Inquiry Request
- **Routing Keys**: `m7.request.inquiry`
- **Roles**: Trader, Market Operation, Broker, Market Maker, Sales
- **Request Limits**: 56/280[7]

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, or 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: • **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 • **DELIVERY**: trades with delivery occurring 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 *DFS180*).
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.

6.4.14 TradeCaptureRprt

- **Type**: Management Response, Broadcast
- **Response to**: TradeCaptureReq (sent to private response queue)
- **Broadcasted**: Yes
- **Routing Keys**: `[schema-version].hlftrd.[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].hlftrd.[acctId]`. Market Operation users will receive the complete trade information (both sell side and buy side). The Trade Capture Report is sent to Market Operation users with the routing key `[schema-version].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 *DFS180*).

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.
TradeList	**SE**	**o**	**0..1**	**Structure**	
Trade	**SE**	**o**	**0..n**	**Structure**	
tradeId	A	m		Long	The trade ID of the trade. In XBID-connected environments remote trades and LTS trades share the same trade ID sequence. Therefore a gap in LTS trade IDs may occur in the course of messages.
state	A	m		Char(4)	The current state of the trade. Valid values: • **CNCL**: The trade was cancelled by market operations. • **RGRA**: The requested recall was granted by market operations. • **ACTI**: Thr 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. • **CREQ**: A cancellation was requested from the local market operations. Not applicable to TradeStlmntRprt. • **CREJ**: A cancellation was rejected by the global market operations. Not applicable to TradeStlmntRprt. • **RSFA**: The request was sent for approval to SOB (XBID). 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: • **false**: Not pre-arranged trade • **true**: Pre-arranged trade

XML Tag	Type	m/o	No.	Data Type	Short description
prearrangeType	A	o		Char(3)	The type of pre-arranged trade. • **OTC**: over the counter trade • **OPT**: the value is deprecated • **PNC**: the value is deprecated
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.
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. Valid values: • **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. Deprecated.
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.
location	A	o		Char(64)	Location within the delivery area. Should be only set for products with locations enabled.
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).
dlvryAreaId	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.

XML Tag	Type	m/o	No.	Data Type	Short description
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: • **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.
clOrdrId	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: • **O**: Open position indicator • **C**: Close position indicator
brokerUserId	A	o		Integer	UserID of the broker user
clgAcctId	A	o		Integer	Clearing account Id. Deprecated.
remoteOrdrId	A	o		Long	Order Id as returned by remote backend system (ie. XBID SOB)
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).
dlvryAreaId	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: • **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.
clOrdrId	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.

XML Tag	Type	m/o	No.	Data Type	Short description
openCloseInd	A	o		Char(1)	This is mandatory for Futures and Cross product spreads. For other Commodities it is not used. Valid values: • **O**: Open position indicator • **C**: Close position indicator
brokerUserId	A	o		Integer	UserID of the broker user
clgAcctId	A	o		Integer	Clearing account Id. Deprecated.
remoteOrdrId	A	o		Long	Order Id as returned by remote backend system (ie. XBID SOB)

6.4.15 PblcTradeConfReq

- **Type**: Inquiry Request
- **Routing Keys**: `m7.request.inquiry`
- **Roles**: Trader, Market Operation, Broker, Data Vendor, Resdispatch Balance User
- **Request Limits**: 56/280[8]

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 *DFS180*).
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
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.

6.4.16 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, Broker, Data Vendor, Resdispatch Balance User

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
PblcTradeConfRprt	**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 *DFS180*).
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**	
PblcTradeConf	**SE**	**o**	**0..n**	**Structure**	

XML Tag	Type	m/o	No.	Data Type	Short description
tradeId	A	m		Long	Trade Id of the underlying trade.
state	A	m		Char(4)	The current state of the trade. Valid values: • **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.
px	A	m		Long	The execution price in Eurocents (1 Euro = 100).
tradeExecTime	A	m		DateTime	The trade execution time.
revisionNo	A	m		Long	The revision number of the trade. This is increased by one every time the trade is changed.
buyDlvryAreaId	A	o		Char(16)	The delivery area of the buy side.
sellDlvryAreaId	A	o		Char(16)	The delivery area of the sell side.
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.

6.4.17 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.

XML Tag	Type	m/o	No.	Data Type	Short description
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 *DFS180*).
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)	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.

6.4.18 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.

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 *DFS180*).
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**	
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)	The current state of the contract / contract-delivery area combination. Valid values:
					• **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:
					  ◦ The end of a trading phase of a global product (i.e. ContractInfoRprt has been received from XBID), in case there are no remote orders and its linked product has a longer trading phase than the remote product. No automatic order transfer is performed.
					  ◦ An automatic order transfer for a delivery area has been performed, or skipped/reverted due to the valid reasons (e.g. a disconnection from XBID). For more details on the AOT process please refer to *DFS160a*.
					• **HIBE**: Hibernate: The contract was manually deactivated by market operations.
					• **IACT**: Inactive: The contract is inactive and not available for trading.

XML Tag	Type	m/o	No.	Data Type	Short description
tradingPhase	A	m		Char(4)	Specifies the contract phase. Valid values: • **CONT**: Continuous Trading. Used for Contracts in state ACTI. • **CLSD**: Closed Phase, Trading is not possible during this phase. Used for Contracts in states other than ACTI.
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.
CallPut	A	o		Char(1)	In the event that the bespoke contract is an option, this field contains a contract type. Valid values: • **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 (i.e. XBID SOB)
DlvryAreaState	**SE**	**o**	**0..n**	**Structure**	**The type defines the state of a contract for a specific delivery area.**
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.

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

state	A	o		Char(4)	The current state of the contract / contract-delivery area combination. Valid values:
					• **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:
					  • The end of a trading phase of a global product (i.e. ContractInfoRprt has been received from XBID), in case there are no remote orders and its linked product has a longer trading phase than the remote product. No automatic order transfer is performed.
					  • An automatic order transfer for a delivery area has been performed, or skipped/reverted due to the valid reasons (e.g. a disconnection from XBID). For more details on the AOT process please refer to *DFS160a*.
					• **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. Valid values: • **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.
tradingPhaseEnd	A	o		DateTime	The end date and time of the current/next trading phase in the delivery area.
UndrlngContracts	**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

6.4.19 ProdInfoReq

- **Type**: Iquiry 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 *DFS180*).
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.

6.4.20 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.

Key	Value	Prod Type	Value
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	The product supports stop orders.
linkedOrderProduct	true	all	Orders can be linked together for this product.
otcAllowed	true	all	OTC trade registration is allowed for this product.
otcOnly	true	all	The product is for OTC trading only.
commodityLimitEnabled	true	COM	The commodity limit is enabled for the product.
productIsWithinDelivery	true	COM	Trading of the product is allowed when delivery has started.
leadTime		COM	Defines the lead time in minutes for products within delivery.
autoOrderMatcher	true	all	The product uses the automatcher (regular price-time priority matcher).
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:

- 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 *DFS180*).

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.
ProdList	**SE**	**o**	**0..1**	**Structure**	
Prod	**SE**	**o**	**0..n**	**Structure**	
prodType	A	m		Char(3)	Indicates the type of the product. Valid values: • **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.
dsplName	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.
locationsEnabled	A	o		Boolean	Indicates whether the location matching is 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.

XML Tag	Type	m/o	No.	Data Type	Short description
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. For XBID style user defined blocks, this attribute contains the delivery length expressed in number of the original product canonical contracts (e.g. 20-23 has delUnits = 3).
minPx	A	m		Long	The minimum price allowed for orders entered in contracts belonging to this product.
maxPx	A	m		Long	The maximum price allowed for orders entered in contracts belonging to this product.
tickSize	A	m		Integer	Defines the minimum increment for limit prices for this product. The value is entered as an integer, but the decimal price shift is applied (please refer to the example in minimum peak size in *DFS120*).
decShftPx	A	m		Integer	The decimal shift of the price information. A value of 2 results in a display in 1/100 of specified currency.
exchangeId	A	m		Char(255)	The exchange Id of the exchange where the orders for this product are matched.
assetName	A	o		Char(64)	A Text input field used to define the asset name.
assetExchangeId	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. • **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.

XML Tag	Type	m/o	No.	Data Type	Short description
contActDay	A	o		Char(3)	The week day at which the contract is activated. Valid values: • **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.
contExpiryDay	A	o		Char(3)	The week day at which the contract expires. Valid values: • **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: • **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: • **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.**

XML Tag	Type	m/o	No.	Data Type	Short description
dlvryAreaId	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.
tradingDays	A	m		Char(∞)	List of trading days. Valid values: • **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: • **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: • **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).

XML Tag	Type	m/o	No.	Data Type	Short description
ContractNamePattern	CE	o	0..1	Char(255)	Regular expression for the contract name.

6.4.21 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 *DFS180*).
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 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.

XML Tag	Type	m/o	No.	Data Type	Short description
state	A	m		Char(4)	Contains the current market state. Valid values: • **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 *DFS180*).
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 HubToHubReq

- **Type**: Inquiry Request
- **Routing Keys**: `m7.request.inquiry`
- **Roles**: Trader, Data Vendor, Balance User, Admin, Sales User. The user must have additional right 'Capacity info'.
- **Request Limits**: 50/500

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

XML Tag	Type	m/o	No.	Data Type	Short description
HubToHubReq	**SE**	**m**	**1**	**Structure**	
area	A	m		Char(16)	The attribute represents a Delivery Area. The application requires the value to be provided in the request.
dlvryDay	A	m		Date	Delivery date. ATC values are valid for this date.
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 *DFS180*).

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.

6.4.24 HubToHubResp

- **Type**: Inquiry Response
- **Response to**: HubToHubReq (sent to the private response queue)
- **Roles**: Trader, Data Vendor, Balance User, Admin, Sales User. The user must have the additional right 'Capacity info'.
- **Broadcasted**: No
- **Broadcast Routing Keys**: –
- **Broadcast Audience**: –

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

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

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

XML Tag	Type	m/o	No.	Data Type	Short description
HubToHubResp	**SE**	**m**	**1**	**Structure**	
revisionNo	A	m		Long	The H2H matrix version. The version number is generated internally and increases upon any updates of the matrix for requested deliver day and area.
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 *DFS180*).
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.
HubToHubAtcList	**SE**	**o**	**0..n**	**Structure**	

XML Tag	Type	m/o	No.	Data Type	Short description
dlvryStart	A	m		DateTime	The delivery start date
dlvryEnd	A	m		DateTime	The delivery end date
timestmp	A	m		DateTime	The timestamp when the ATC data was received from the Capacity system.
HubFrom	**SE**	**o**	**0..n**	**Structure**	
frm	A	m		Char(16)	The outgoing Area. With relation to the attributes *IN* and *OUT* this area is perceived as the main one. That is, the value *IN* represents the capacity available for flow to the *area from*; and the value *OUT* represents the capacity available for flow from the *area from*.
Atc	**SE**	**o**	**0..n**	**Structure**	
to	A	m		Char(16)	The target Area.
in	A	m		Long	The ATC value *IN* for target Area. That is, capacity from *target area* to *area from*.
out	A	m		Long	The ATC value *OUT* for target Area. That is, capacity from *area from* to *target area*.

6.4.25 H2HAreaInfoReq

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

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

XML Tag	Type	m/o	No.	Data Type	Short description
H2HAreaInfoReq	**SE**	**m**	**1**	**Structure**	
StandardHeader	**SE**	**m**	**1**	**Structure**	**Every message will contain a header at the beginning of the 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 *DFS180*).
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.

6.4.26 H2HAreaInfoResp

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

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

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

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

XML Tag	Type	m/o	No.	Data Type	Short description
H2HAreaInfoResp	**SE**	**m**	**1**	**Structure**	
StandardHeader	**SE**	**m**	**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 *DFS180*).
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.
H2HAreaList	**SE**	**o**	**0..1**	**Structure**	
H2HArea	**SE**	**o**	**0..n**	**Structure**	**Market/Delivery Area information.**
areaId	A	m		Char(16)	Market/Delivery Area Id. In the power market, this corresponds to the EIC code of the market/delivery area.
name	A	o		Char(255)	Name of the market/delivery area, usually used for display purposes.
longName	A	o		Char(255)	Long name of the market/delivery area.

6.4.27 HubToHubNtf

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

- **Broadcast Audience**: All users with the additional right ‘Capacity info’

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

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

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

XML Tag	Type	m/o	No.	Data Type	Short description
HubToHubNtf	**SE**	**m**	**1**	**Structure**	
revisionNo	A	m		Long	The H2H matrix version. The version number is generated internally and increases upon any updates of the matrix. For consistency check purposes it verifies that: • The revisionNo of the last broadcast is higher than the revisionNo of the previous one • Possible gaps between the broadcasts should be detected using a standard broadcast sequence.
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 *DFS180*).
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.
HubToHubAtcList	**SE**	**o**	**0..n**	**Structure**	
dlvryStart	A	m		DateTime	The delivery start date
dlvryEnd	A	m		DateTime	The delivery end date
timestmp	A	m		DateTime	The timestamp when the ATC data was received from the Capacity system.
HubFrom	**SE**	**o**	**0..n**	**Structure**	
frm	A	m		Char(16)	The outgoing Area. With relation to the attributes *IN* and *OUT* this area is perceived as the main one. That is, the value *IN* represents the capacity available for flow to the *area from*; and the value *OUT* represents the capacity available for flow from the *area from*.

XML Tag	Type	m/o	No.	Data Type	Short description
Atc	**SE**	**o**	**0..n**	**Structure**	
to	A	m		Char(16)	The target Area.
in	A	m		Long	The ATC value *IN* for target Area. That is, capacity from *target area* to *area from*.
out	A	m		Long	The ATC value *OUT* for target Area. That is, capacity from *area from* to *target area*.

6.4.28 HubToHubHeartbeat

- **Type**: Broadcast
- **Response to**: –
- **Roles**: [All]
- **Broadcasted**: Yes
- **Broadcast Routing Keys**: `[schema-version].hubToHub`
- **Broadcast Audience**: All users with the additional right ‘Capacity info’

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

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

XML Tag	Type	m/o	No.	Data Type	Short description
HubToHubHeartbeat	**SE**	**m**	**1**	**Structure**	
sobConnectionState	A	o		Char(128)	Information about the state of the connection to XBID. Possible values are CONNECTED/DISCONNECTED. When M7 is configured to check the connection to XBID and it does not receive a heartbeat from XBID within the given period (default configuration is 30 seconds), M7 will send a HubtoHubHeartbeat with the sobConnectionState =DISCONNECTED.
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 *DFS180*).
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.

6.4.29 StlmntProcessInfoReq

- **Type**: Inquiry Request
- **Routing Keys**: `m7.request.inquiry`
- **Roles**: Market Operation, Sales. Access for Traders is configurable.
- **Request Limits**: 56/280[9]

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 *DFS180*).
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.

6.4.30 StlmntProcessInfoRprt

- **Type**: Inquiry Response, Broadcast
- **Response to**: StlmntProcessInfoReq (sent to private response queue)
- **Broadcasted**: Yes
- **Routing Keys**: `[schema-version].settlement`, `[schema-version].bg.stlmnt.[acctId]`
- **Broadcast audience**: Admins, Sales
  Configurable (with routing key `[schema-version].bg.stlmnt.[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 *DFS180*).
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: • **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

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

6.4.31 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: • **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.

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 *DFS180*).
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.

6.4.32 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 *DFS180*).
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.

XML Tag	Type	m/o	No.	Data Type	Short description
refPxType	A	o		Char(1)	The type of the reference price updated. Valid values: • **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.

6.5 Order Throttling

The described messages are available only if the Order Throttling feature is activated.

6.5.1 ThrottlingStatusReq

- **Type**: Inquiry Request
- **Routing Keys**: `m7.request.inquiry.throttling`
- **Roles**: Trader
- **Request Limits**: By default, the system allows 1 request each 3 seconds per user. No long term request limit is applied.

Throttling Status Request is used to retrieve the current throttling status for the Trader's Member (e.g. actual count of the order management transactions for the respective throttling rule, current throttling rules configuration).

If the Order Throttling is activated, the response to this request is a ThrottlingStatusResp. Otherwise, an ErrResp is returned.

XML Tag	Type	m/o	No.	Data Type	Short description
ThrottlingStatusReq	**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 *DFS180*).
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 ThrottlingStatusResp

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

- **Roles**: Trader
- **Broadcasted**: No
- **Broadcast Routing Keys**: –
- **Broadcast Audience**: –

This message is returned as a response to the Throttling Status Request. It is sent to the private response queue of the user requesting the Throttling Status Request.

XML Tag	Type	m/o	No.	Data Type	Short description
ThrottlingStatusResp	**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 *DFS180*).
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.
ThrottlingMemberStatus	**SE**	**m**	**1**	**Structure**	**List of throttling member statuses.**
mbrId	A	m		Char(5)	The unique identifier of the member.
timestamp	A	m		DateTime	Date and time of the OMT count calculation.
status	A	m		Char(14)	Current throttling status of the given Member (NO_RESTRICTION, WARNING, RESTRICTED).
ShortRule	**SE**	**m**	**1**	**Structure**	**Definition of throttling rule.**
observationPeriodLength	A	m		Integer	Duration in seconds of floating window.
tolerancePeriodLength	A	m		Integer	Duration in seconds after which post warning throttling is activated in case number of OMTs not decreased bellow soft limit.
reconnectionCoolDown	A	m		Integer	Duration in seconds of member suspension (throttling activated).
omtLimitL1	A	m		Integer	Number of OMTs (Order management transactions) in observation period which triggers warning.
omtLimitL2	A	m		Integer	Maximal number of OMTs (Order management transactions) allowed in observation period.
Status	**SE**	**m**	**1**	**Structure**	**Current status of throttling rule evaluated based its configuration.**

XML Tag	Type	m/o	No.	Data Type	Short description
status	A	m		Char(14)	Current throttling status of the given Member on the given Order Throttling Rule (NO_RESTRICTION, WARNING, RESTRICTED).
releaseTimestamp	A	o		DateTime	Date and time when member restriction will be released if the Rule Status is Restricted. Date and time when the Tolerance ends if the Rule Status is Warning.
currentOmtCount	A	m		Integer	Current number of OMTs within actual period.
LongRule	**SE**	**m**	**1**	**Structure**	**Definition of throttling rule.**
observationPeriodLength	A	m		Integer	Duration in seconds of floating window.
tolerancePeriodLength	A	m		Integer	Duration in seconds after which post warning throttling is activated in case number of OMTs not decreased bellow soft limit.
reconnectionCoolDown	A	m		Integer	Duration in seconds of member suspension (throttling activated).
omtLimitL1	A	m		Integer	Number of OMTs (Order management transactions) in observation period which triggers warning.
omtLimitL2	A	m		Integer	Maximal number of OMTs (Order management transactions) allowed in observation period.
Status	**SE**	**m**	**1**	**Structure**	**Current status of throttling rule evaluated based its configuration.**
status	A	m		Char(14)	Current throttling status of the given Member on the given Order Throttling Rule (NO_RESTRICTION, WARNING, RESTRICTED).
releaseTimestamp	A	o		DateTime	Date and time when member restriction will be released if the Rule Status is Restricted. Date and time when the Tolerance ends if the Rule Status is Warning.
currentOmtCount	A	m		Integer	Current number of OMTs within actual period.

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 *DFS180*).
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.
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
token	A	o		Char(255)	Unique token for connecting to additional services (e.g. V7 API).
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: • **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
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: • **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.
deliveryRiskApplied	A	o		Boolean	Provided for exchanges with the delivery risk feature enabled. If enabled, it indicates if the delivery risk applies to the member. Its application affects cash limit calculation. For more details, please refer to *Delivery Risk* in *DFS170*. 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 *DFS180*).
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 DlvryAreaInfoReq

- **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
DlvryAreaInfoReq	**SE**	**m**	**1**	**Structure**	
acctId	A	o		Char(32)	ID of the account. If not specified, all Delivery Areas assigned to requesting user 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 *DFS180*).
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**	

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 *DFS180*).
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**	
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: • **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.
mktAreaId	A	m		Char(16)	The EIC id of the market area to which the delivery area belongs to.
dlvryAreaId	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: • **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.**
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 *DFS180*).
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.**

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 *DFS180*).
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**	
revisionNo	A	m		Long	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. 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.
mktAreaId	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)	In a CreateMktAreaReq, it is the desired state of the market area. In a ModifyMktAreaReq, it is the current state of the market area. Valid values: • **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 *DFS180*).
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 *DFS180*).

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.
AcctList	**SE**	**o**	**0..1**	**Structure**	**List of accounts**
Acct	**SE**	**o**	**0..n**	**Structure**	**Account element**
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
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: • **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. Deprecated.

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.
onBehalfUserId	A	o		Integer	UserID of the target user in case of On-behalf trading (see On-behalf Trading chapter in *DFS180*).
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: • **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

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 users of the system, except for Reporting User role. Reporting Users are not entitled to any rights in the system besides the retrieval of the reports and therefore are not included. 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 *DFS180*).
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.
mbrId	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 and mailAddress 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.

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 *DFS180*).
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.
UsrList	**SE**	**m**	**1**	**Structure**	
Usr	**SE**	**o**	**0..n**	**Structure**	
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: • **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: • **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 *DFS180*).
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.
UndrlngContracts	**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 *DFS180*).

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 *DFS180*).
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: • **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	Redispatch Balance user & Balance user	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	X	
Error Response (ErrResp)	X	X	X	X	X	X	X	X	
Change password Request (ChgPwdReq)	X	X	X	X	X	X	X	X	
Throttling Status Request (ThrottlingStatusReq)	X			X					

Order Entry and Maintenance

Message	Trader	Broker	Market Operation	Redispatch Balance user & Balance user	Sales	Data Vendor	Settlement Operation	Clearing user	Capacity info
Order Entry (OrdrEntry)	X	X	X	X					
Order Modify (OrdrModify)	X	X	X	X					
Order Request (OrdrReq)	X	X	X	X					
Order Execution Report (OrdrExeRprt)	X	X	X	X					

Message	Trader	Broker	Market Operation	Redispatch Balance user & Balance user	Sales	Data Vendor	Settlement Operation	Clearing user	Capacity info
Modify All Orders (ModifyAllOrdrs)	X	X	X	X					
Order Limit Request (OrdrLmtReq)	X	X	X	X					
Order Limit Report (OrdrLmtRprt)	X	X	X	X					

Trade Maintenance

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

Market Information

Message	Trader	Broker	Market Operation	Redispatch Balance user & Balance user	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	Trader	Broker	Market Operation	Redispatch Balance user & Balance user	Sales	Data Vendor	Settlement Operation	Clearing user	Capacity info
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	X		X			
Public Trade Confirmation Report (PblcTradeConfRprt)	X	X	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		
Market State Report (MktStateRprt)	X	X	X	X	X	X	X		
Hub-to-Hub Matrix Request (HubToHubReq)[8]									X
Hub-to-Hub Matrix Response (HubToHubResp)[9]									X
Hub-to-Hub Matrix Broadcast (HubToHubNtf)[10]									X
Hub-to-Hub Hearbeat (HubToHubHeartbeat)[11]									X
Hub-to-Hub Area Info Request (H2HAreaInfoReq)[11]	X		X	X	X	X			X

Message	Trader	Broker	Market Operation	Redispatch Balance user & Balance user	Sales	Data Vendor	Settlement Operation	Clearing user	Capacity info
Hub-to-Hub Area Info Response (H2HAreaInfoResp)[11]	X		X	X	X	X			X
Settlement Process Information Request (StlmntProcessInfoReq)	(X)[12]	(X)[12]	X		X			X	
Settlement Process Information Report (StlmntProcessInfoRprt)	(X)[12]	(X)[12]	X		X			X	
Reference Price Request (RefPxReq)			X	X		X			
Reference Price Report (RefPxRprt)	X	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.

Reference Data

Message	Trader	Broker	Market Operation	Redispatch Balance user & Balance user	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		

Message	Trader	Broker	Market Operation	Redispatch Balance user & Balance user	Sales	Data Vendor	Settlement Operation	Clearing user	Capacity info
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					
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. VeriSign: http://www.verisign.com/support/roots.html. Comodo: https://support.comodo.com/index.php?_m=downloads&_a=view&parentcategoryid=1↩

3. By default, the parameter is set to 90 days. ↩

4. By default, the parameter is set to 10 days. ↩

5. *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.↩

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

7. 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.↩

8. 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.↩

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.↩