NASDAQ OMX®

Genium INET℠

OMnet Message Reference

HKEX

Version: 2.0.0801

Document ID:	OMnet_MessRef_12
Documentation Release:	GENIUM_Product_a900
Release Date:	2013-07-02
Publication Date:	2013-07-02
Confidentiality:	Non-confidential
API Kit:	36763

Table of Contents

List of Tables

List of Figures

1 Summary of Changes

Only changes affecting messages included in this message reference are listed.

Changes between (39708) and (39708) for HKEX (a83/a89).

	Changed message	Changes	Comments
1	BO5	Textual changes in message description: section: Usage and Conditions: note #2: new	
2	BU120	Textual changes in message description: section: Usage and Conditions: example #2: titled-block #3: list #1: listitem #2: new titled-block #4: list #1: listitem #2: new	
3	TR71	Textual changes in message description: deleted empty section before before chapter “Answer, comments.”	

2 Document Information

2.1 References

Here is a list of OMnet related documents:

- *OMnet Message Reference Manual, Introduction*
- *OMnet Message Reference Manual*
- *OMnet Application Programmer's Interface Manual*
- *System Error Messages Reference Manual*

2.2 Reader's Roadmap

This message reference contains the following chapters:

Chapter	Description
Summary of Changes	The Summary of Changes table lists two kinds of changes: • Changes between two specific API builds. • Relevant changes made to the text in the manual describing the API. The Summary of Changes table does not list the following: • Changes in the internal order of fields within a structure. • The connection between an item that replaces another item. This means that if a message/struct/field/enumeration is replaced by another, the table will list the removed item as “Removed” and the added item as “Added.”
Messages	This chapter lists and describes all messages that are available in this configuration of the API. For more information, see the Messages Chapter below.
Common Structures	The most common structures are defined here.
Named Structures	Named structures are defined here.
Broadcast Overview	This chapter lists all broadcasts occurring in the manual. This is also where each broadcast’s Information Type Value is provided.
Detailed Field Information	This chapter provides a general description of all fields used by the structures defined in this reference. Any message-specific information regarding a field is provided in each respective message chapter.

2.2.1 The OMnet Messages Chapter

The OMnet API defines the information that can be exchanged between the system and an external application. It consists of a configurable set of messages, all of which are of one of the following types:

Type	Description
Transaction	Input to the system, a request for action (an order, for example).
Query + Answer	A query/request to the system (give me all trades since market opening, for example) that will trigger an answer from the system.
Broadcast	Information created by the system and distributed to all applications subscribing to this particular information (a closed deal, for example).

The way in which the data is encapsulated in the messages varies. The content could have a nested and fixed structure with a single top container, or a message could be a variable information message (VIM), meaning that a number of data structures follow sequentially, intervened by headers declaring the size and nature of the next data chunk.

Each message chapter has all or a subset of the following sections depending on the transaction type.

<table>
<tr><th>Section</th><th>Description</th></tr>
<tr><td>Fingerprint</td><td>Each message has a Fingerprint section containing the following information:
<table>
<tr><th>Heading</th><th>Description</th></tr>
<tr><td>Transaction type</td><td>Transaction type is the identification of the transaction; broadcast, query or answer.
For more information on how the Transaction type is designed, refer to OMnet Message Reference Manual, Introduction.</td></tr>
<tr><td>Calling sequence</td><td>The Calling sequence is the name of the callable routine for the transaction.
For more information, refer to OMnet Application Programmer’s Interface Manual.</td></tr>
<tr><td>Struct name</td><td>Is the name of the top structure in the message.</td></tr>
<tr><td>Info type</td><td>The info type is an attribute of the information object. Applicable for broadcasts only.
Refer to OMnet Application Programmer’s Interface Manual.</td></tr>
<tr><td>Segmented</td><td>Specifies if an answer or broadcast is segmented or not (true/false).
For details, refer to OMnet Message Reference Manual, Introduction.</td></tr>
<tr><td>Partitioned</td><td>Specifies if a transaction or query is partitioned or not (true/false).
For more information, refer to OMnet Message Reference Manual, Introduction.</td></tr>
<tr><td>Facility</td><td>Transactions are sent on paths through the system called facilities. The system is only able to rout a transaction correctly if it is sent on the correct facility.</td></tr>
</table></td></tr>
</table>

<table>
<tr><th>Section</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><th>Heading</th><th>Description</th></tr>
<tr><td></td><td>Refer to OMnet Application Programmer's Interface Manual.</td></tr>
<tr><td>Virtual Underlying</td><td>Virtual Underlying is a grouping concept that makes the dissemination of information and the subscription of information more efficient.

For broadcasts and queries supporting this concept, Virtual Underlying is set to “True.” For broadcasts and queries not supporting this concept, Virtual Underlying is not listed in the fingerpring table.

For details on this, refer to OMnet Message Reference Manual, Introduction.</td></tr>
</table>
</td></tr>
<tr><td>Related Messages</td><td>Lists any messages that in one way or another are related to the described message. It could be a query that returns the content of a related broadcast, or two related broadcasts disseminating similar content.</td></tr>
<tr><td>Purpose</td><td>The purpose of the message is described here.</td></tr>
<tr><td>Structure</td><td>The structure of the message is presented here.</td></tr>
<tr><td>Usage and Conditions</td><td>Message specific information regarding fields is provided here. The general description of all fields is presented in the Detailed Field Information chapter.</td></tr>
<tr><td>Structure Contents</td><td>Provides any additional information regarding the structures if needed.</td></tr>
<tr><td>Return Codes</td><td>Some messages may return codes indicating if it was successfully received and processed by the system. These codes are described in the Return Codes section.</td></tr>
<tr><td>Answer Structure</td><td>If the message is a query, the structure of the answer is presented here.</td></tr>
<tr><td>Answer Comments</td><td>If the message is a query, any needed information regarding the answer is provided here.</td></tr>
<tr><td>Answer Structure Contents</td><td>Provides any additional information regarding the answer structures if needed.</td></tr>
</table>

2.3 Navigating the Document

This manual uses links to facilitate easy and quick navigation through the structures. For example, it is simple to navigate "Summary of Changes" item > Message > Structure > Sub-structure > Named-Structure > Field and back.

Depending on the PDF reader you are using, the “Back” button may not be visible by default. The way in which you make it visible may also differ depending on the type of PDF reader you have. The following description applies to a number of Adobe Acrobat versions:

1. Open a PDF document in your Adobe Acrobat application.
2. Select View > Toolbars > More Tools (or View > Tools > Customize Toolbars, and so on) to open the More Tools/Customize Toolbars and so on dialog.

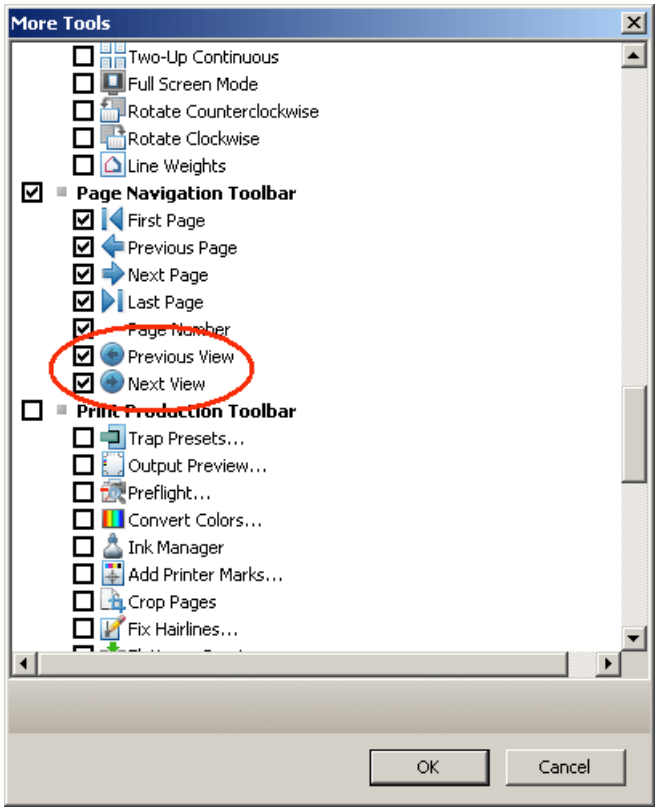

Figure 1: More Tools Dialog

3. Check the Page Navigation Toolbar and make sure that, at a minmum, the **Previous Next** and **Next View** buttons are selected. It is recommended that you make all of the Page Navigation Toolbar buttons visible since they all will aid you when you navigate the document.
4. Click **OK**. The buttons are now visible in your toolbar.

Note:

If you are reading this pdf file via a web browser, make sure you enable the very same buttons there, too. You do this by right-clicking the toolbar and selecting the **Previous** and **Next View** buttons.

3 OMnet Messages

3.1 Reference Data

3.1.1 BU2 [Series Update BROADCAST]

3.1.1.1 Fingerprint

BROADCAST properties	
transaction type	BU2
calling sequence	omniapi_read_event_ext_ex or omniapi_read_event_block
struct name	series_update_bu2
info type	general

3.1.1.2 Related Messages

DQ2, the answer will take into account any modifications made.

3.1.1.3 Purpose

The Series Update broadcast is sent when a new series, or combinations if any, has been defined or updated in the central system.

Note: Preferably, the more modern (Delta Queries and Broadcasts concept) BU124 should be used instead of BU2 single orders and BU126 should be used instead of BU2 + BU5 for combinations.

3.1.1.4 Structure

The BU2 BROADCAST has the following structure:

```
struct series_update_bu2 {
   struct broadcast_type
   UINT16_T chg_type_n  // Change Type
   char[2] filler_2_s  // Filler
   struct da2 {
      struct series  // Named struct no: 50000
      struct upper_level_series
      INT32_T contract_size_i  // Contract Size
      INT32_T price_quot_factor_i  // Price, Quotation Factor
      UINT32_T series_sequence_number_u  // Series, Sequence Number
      UINT16_T state_number_n  // Trading State Number
      UINT16_T step_size_multiple_n  // Tick Size, Multiple
      char[32] ins_id_s  // Series, Identity
```

```
        char[12] isin_code_s  // ISIN Code
        UINT8_T suspended_c  // Suspended
        char[8] date_last_trading_s  // Date, Last Trading
        char[6] time_last_trading_s  // Time, Last Trading
        char[8] settlement_date_s  // Date, Settlement
        char[8] start_date_s  // Date, Start
        char[8] end_date_s  // Date, End
        char[8] date_delivery_start_s  // Date, Delivery Start
        char[8] date_delivery_stop_s  // Date, Delivery Stop
        UINT8_T series_status_c  // Series, Status
        char[32] long_ins_id_s  // Series Name, Long
        char[8] date_first_trading_s  // Date, First Trading
        char[6] time_first_trading_s  // Time, First Trading
        UINT8_T traded_in_click_c  // Traded in GENIUM
        char[8] abbr_name_s  // Abbreviated Name
        char[6] stock_code_s  // Stock Code
        UINT8_T ext_info_source_c  // External Information Source
        char[8] effective_exp_date_s  // Effective Expiration Date
        char[2] filler_2_s  // Filler
    }
}
```

3.1.1.5 Usage and Conditions

Change Type

states what type of update is at hand, as described in the field information section.

Trading State Number

contains the immediate ISS.

3.1.2 BU4 [Underlying Update BROADCAST]

3.1.2.1 Fingerprint

BROADCAST properties	
transaction type	BU4
calling sequence	omniapi_read_event_ext_ex or omniapi_read_event_block
struct name	underlying_update_bu4_bu19
info type	general

3.1.2.2 Related Messages

DQ4, the answer will take into account any modifications made.

3.1.2.3 Purpose

The Underlying Update broadcast is sent when a new underlying has been defined or updated in the central system.

Note: Preferably, the more modern BU120 should be used instead of BU4 (Delta Queries and Broadcasts concept).

3.1.2.4 Structure

The BU4 BROADCAST has the following structure:

```
struct underlying_update_bu4_bu19 {
   struct broadcast_type
   UINT16_T chg_type_n  // Change Type
   char[2] filler_2_s  // Filler
   struct da4_da19 {
      INT32_T subscription_price_i  // Subscription, Price
      INT32_T interest_rate_i  // Interest Rate
      UINT16_T commodity_n  // Commodity Code
      char[6] com_id_s  // Underlying Identity
      char[12] isin_code_s  // ISIN Code
      UINT16_T dec_in_price_n  // Decimals, Price
      char[8] date_release_s  // Date, Issue
      char[8] date_termination_s  // Date, Maturity
      char[8] date_dated_s  // Date, Dated
      char[32] name_s  // Name
      char[3] base_cur_s  // Currency, Trading
      UINT8_T deliverable_c  // Deliverable
      UINT16_T coupon_frequency_n  // Coupon Frequency
      INT64_T nominal_value_q  // Nominal Value
      UINT16_T day_count_n  // Day Count
      UINT16_T days_in_interest_year_n  // Days In Interest Year
      UINT32_T coupon_interest_i  // Coupon Interest
      UINT16_T coupon_settlement_days_n  // Coupon Settlement Days
      UINT8_T underlying_type_c  // Type, Underlying
      UINT8_T price_unit_c  // Price Unit, Underlying
      UINT16_T dec_in_nominal_n  // Decimals, Nominal
      UINT16_T state_number_n  // Trading State Number
      UINT16_T linked_commodity_n  // Linked Commodity Code
      UINT8_T fixed_income_type_c  // Fixed Income Type
      UINT8_T underlying_status_c  // Underlying Status
      char[6] underlying_issuer_s  // Underlying Issuer
      char[6] time_delivery_start_s  // Time, Delivery Start
      char[6] time_delivery_stop_s  // Time, Delivery Stop
      char[4] sector_code_s  // Sector Code
      UINT16_T items_n  // Items
      Array COUPON [max no: 80] {
         char[8] date_coupdiv_s  // Coupon/Dividend Date
         UINT32_T dividend_i  // Dividend
      }
      UINT8_T virtual_c  // Virtual
      char[4] member_circ_numb_s  // Member, Circular Number
      CHAR inv_scheme_c  // Investment Scheme
```

```
        char[8] date_closing_s  // Date, Closing
        char[8] date_last_s  // Date, Last
        char[2] country_id_s  // Name, Country
        UINT8_T cur_unit_c  // Currency Unit
        char[3] filler_3_s  // Filler
    }
}
```

3.1.2.5 Usage and Conditions

Change Type

states what type of update is at hand, as described in the field information section.

Trading State Number

will contain the immediate ISS.

3.1.3 BU5 [Combination Update BROADCAST]

3.1.3.1 Fingerprint

BROADCAST properties	
transaction type	BU5
calling sequence	omniapi_read_event_ext_ex or omniapi_read_event_block
struct name	combo_update_bu5
info type	general

3.1.3.2 Related Messages

DQ5, the answer will take into account any modifications made.

3.1.3.3 Purpose

The Combo Series Update broadcast is sent when a new combo series has been defined in the central system.

Note: Preferably, the more modern BU126 should be used instead of BU2 + BU5 for combinations (Delta Queries and Broadcasts concept).

3.1.3.4 Structure

The BU5 BROADCAST has the following structure:

```
struct combo_update_bu5 {
   struct broadcast_type
   UINT16_T chg_type_n  // Change Type
```

```
    char[2] filler_2_s  // Filler
    struct da5 {
       struct combo_series
       char[32] cbs_id_s  // Combo Series, Identity
       UINT8_T items_c  // Item
       char[3] filler_3_s  // Filler
       Array ITEM [max no: 4] {
          struct series  // Named struct no: 50000
          UINT16_T ratio_n  // Ratio
          CHAR op_if_buy_c  // Operation if Buy
          CHAR op_if_sell_c  // Operation if Sell
       }
    }
 }
```

3.1.3.5 Usage and Conditions

Change Type

states what type of update is at hand, as described in the field information section.

3.1.4 BU9 [Series Backoffice Update BROADCAST]

3.1.4.1 Fingerprint

BROADCAST properties	
transaction type	BU9
calling sequence	omniapi_read_event_ext_ex or omniapi_read_event_block
struct name	series_bo_update_bu9
info type	general

3.1.4.2 Related Messages

DQ9, the answer will take into account any modifications made.

3.1.4.3 Purpose

The Series Backoffice Update broadcast is sent when a new series has been defined or updated in the central system, including expired ones and other non-tradable series, for example, payment series.

Note: Preferably, the more modern BU125 should be used instead of BU9 (Delta Queries and Broadcasts concept).

3.1.4.4 Structure

The BU9 BROADCAST has the following structure:

```
struct series_bo_update_bu9 {
   struct broadcast_type
   UINT16_T chg_type_n  // Change Type
   char[2] filler_2_s  // Filler
   struct da9 {
      struct series  // Named struct no: 50000
      struct upper_level_series
      INT32_T contract_size_i  // Contract Size
      INT32_T price_quot_factor_i  // Price, Quotation Factor
      UINT16_T state_number_n  // Trading State Number
      char[32] ins_id_s  // Series, Identity
      char[12] isin_code_s  // ISIN Code
      UINT8_T stopped_by_issue_c  // Stopped By Issue
      char[12] isin_code_old_s  // ISIN Code, Old Series
      char[8] date_notation_s  // Date, Notation
      char[8] date_last_trading_s  // Date, Last Trading
      char[6] time_last_trading_s  // Time, Last Trading
      char[8] date_delivery_start_s  // Date, Delivery Start
      char[8] date_delivery_stop_s  // Date, Delivery Stop
      UINT8_T deliverable_c  // Deliverable
      UINT8_T suspended_c  // Suspended
      UINT8_T series_status_c  // Series, Status
      UINT8_T tm_template_c  // Template Series
      UINT8_T tm_series_c  // Tailor Made Series
      char[8] settlement_date_s  // Date, Settlement
      char[8] start_date_s  // Date, Start
      char[8] end_date_s  // Date, End
      UINT8_T accept_collateral_c  // Accepted as Collateral
      char[8] date_first_trading_s  // Date, First Trading
      char[6] time_first_trading_s  // Time, First Trading
      UINT8_T traded_in_click_c  // Traded in GENIUM
      UINT8_T traded_c  // Traded
      char[8] effective_exp_date_s  // Effective Expiration Date
      CHAR filler_1_s  // Filler
   }
}
```

3.1.4.5 Usage and Conditions

Trading State Number

will contain the immediate ISS.

3.1.5 BU10 [Instrument Class Update BROADCAST]

3.1.5.1 Fingerprint

BROADCAST properties	
transaction type	BU10
calling sequence	omniapi_read_event_ext_ex or omniapi_read_event_block
struct name	inst_class_update_bu10_bu20

BROADCAST properties	
info type	general

3.1.5.2 Related Messages

DQ10, the answer will take into account any modifications made.

3.1.5.3 Purpose

The Instrument Class Update broadcast is sent when a new class, or combination class if any, has been defined or updated in the central system.

Note: Preferably, the more modern BU122 should be used instead of BU10 (Delta Queries and Broadcasts concept).

3.1.5.4 Structure

The BU10 BROADCAST has the following structure:

```
struct inst_class_update_bu10_bu20 {
   struct broadcast_type
   UINT16_T chg_type_n  // Change Type
   char[2] filler_2_s  // Filler
   struct da10_da20 {
      struct series  // Named struct no: 50000
      struct upper_level_series
      INT32_T price_quot_factor_i  // Price, Quotation Factor
      INT32_T contract_size_i  // Contract Size
      INT32_T exerc_limit_i  // Exercise, Limit
      INT32_T redemption_value_i  // Redemption Value
      INT32_T min_qty_increment_i  // Minimum Quantity Increment
      UINT16_T derivate_level_n  // Derivate Level
      UINT16_T dec_in_strike_price_n  // Decimals, Strike Price
      UINT16_T dec_in_contr_size_n  // Decimals, Contract Size
      UINT16_T rnt_id_n  // Ranking Type
      UINT16_T dec_in_premium_n  // Decimals, Premium
      UINT16_T items_n  // Items
      Array ITEM [max no: 12] {
         struct tick_size
      }
      UINT16_T dec_in_deliv_n  // Decimals, Delivery
      UINT16_T items_block_n  // Item, Block
      Array BLOCK_SIZE [max no: 4] {
         INT64_T maximum_size_u  // Block Size, Maximum Volume
         UINT32_T minimum_size_n  // Block Size, Minimum Volume
         UINT32_T block_n  // Block Size
         UINT8_T lot_type_c  // Lot, Type
         char[3] filler_3_s  // Filler
      }
      UINT16_T cleared_dec_in_qty_n  // Decimals, Quantity
      UINT16_T virt_commodity_n  // Virtual Underlying
      UINT16_T dec_in_fixing_n  // Decimals, Fixing
```

```
        char[3] base_cur_s  // Currency, Trading
        UINT8_T traded_c  // Traded
        UINT8_T exerc_limit_unit_c  // Exercise, Limit Unit
        char[14] inc_id_s  // Instrument Class, Identity
        char[10] trc_id_s  // Trade Report Class
        char[32] name_s  // Name
        CHAR is_fractions_c  // Fraction, Premium
        UINT8_T price_format_c  // Premium/Price Format
        UINT8_T strike_price_format_c  // Strike Price, Format
        UINT8_T cabinet_format_c  // Cabinet Format
        UINT8_T price_unit_premium_c  // Price Unit, Premium
        UINT8_T price_unit_strike_c  // Price Unit, Strike
        char[32] settl_cur_id_s  // Currency, Settlement
        char[3] credit_class_s  // Credit Class
        char[12] csd_id_s  // CSD, Identity
        UINT8_T trd_cur_unit_c  // Traded Currency Unit
        UINT8_T collateral_type_c  // Collateral types
        UINT8_T fixing_req_c  // FIXING_REQ_C
        CHAR[2] mbs_id_s  // Minimum Bid Schedule
        char[12] valuation_group_id_s  // VAG, Identity ; Of type: VAG_ID_S
        char[3] filler_3_s  // Filler
    }
}
```

3.1.5.5 Usage and Conditions

Change Type

states what type of update is at hand, as described in the field information section.

3.1.6 BU12 [Account Type Update BROADCAST]

3.1.6.1 Fingerprint

BROADCAST properties	
transaction type	BU12
calling sequence	omniapi_read_event_ext_ex or omniapi_read_event_block
struct name	account_type_update_bu12
info type	general

3.1.6.2 Related Messages

DQ12, the answer will take into account any modifications made.

3.1.6.3 Purpose

The Account Type Update broadcast is sent whenever a change has occured regarding an account type.

3.1.6.4 Structure

The BU12 BROADCAST has the following structure:

```
struct account_type_update_bu12 {
   struct broadcast_type
   UINT16_T chg_type_n  // Change Type
   char[2] filler_2_s  // Filler
   struct da12 {
      char[12] acc_type_s  // Account Type
      char[40] description_s  // Description
      UINT8_T open_close_c  // Open or Closed
      UINT8_T transitory_c  // Transitory
      UINT8_T market_maker_c  // Market Maker
      UINT8_T own_inventory_c  // Own Inventory
      UINT8_T exclusive_opening_sell_c  // Exclusive Opening Sell
      UINT8_T positions_allowed_c  // Positions, Allowed
      UINT8_T trades_allowed_c  // Trades, Allowed
      char[12] atr_id_s  // Account Type Rule
      CHAR origin_c  // Origin, Account Type
   }
}
```

3.1.6.5 Usage and Conditions

Change Type

states what type of update is at hand (addition, modification, deletion) as described in the field information section.

3.1.7 BU13 [Account Fee Type Update BROADCAST]

3.1.7.1 Fingerprint

BROADCAST properties	
transaction type	BU13
calling sequence	omniapi_read_event_ext_ex or omniapi_read_event_block
struct name	account_fee_type_update_bu13
info type	general

3.1.7.2 Related Messages

DQ13, the answer will take into account any modifications made.

3.1.7.3 Purpose

The Account Fee Type Update broadcast is sent whenever a change has occured regarding an account fee type.

3.1.7.4 Structure

The BU13 BROADCAST has the following structure:

```
struct account_fee_type_update_bu13 {
   struct broadcast_type
   UINT16_T chg_type_n  // Change Type
   char[2] filler_2_s  // Filler
   struct da13 {
      char[12] fee_type_s  // Account Fee Type
      char[40] description_s  // Description
   }
}
```

3.1.7.5 Usage and Conditions

Change Type

states what type of update is at hand (addition, modification, deletion) as described in the field information section.

3.1.8 BU18 [Non-Trading Days Update BROADCAST]

3.1.8.1 Fingerprint

BROADCAST properties	
transaction type	BU18
calling sequence	omniapi_read_event_ext_ex or omniapi_read_event_block
struct name	non_trading_days_update_bu18
info type	general

3.1.8.2 Related Messages

DQ18, the answer will take into account any modifications made.

3.1.8.3 Purpose

The Non Trading Days Update broadcast is sent whenever a change has occured regarding non-trading days.

3.1.8.4 Structure

The BU18 BROADCAST has the following structure:

```
struct non_trading_days_update_bu18 {
   struct broadcast_type
   UINT16_T chg_type_n  // Change Type
   char[2] filler_2_s  // Filler
   struct da18 {
      UINT8_T country_c  // Country Number
      UINT8_T market_c  // Market Code
      char[8] date_non_trading_s  // Date, Non Trading
      UINT8_T closed_for_trading_c  // Closed, trading
      UINT8_T closed_for_settlement_c  // Closed, settlement
      UINT8_T closed_for_clearing_c  // Closed, clearing
      char[3] filler_3_s  // Filler
   }
}
```

3.1.8.5 Usage and Conditions

Change Type

states what type of update is at hand (addition, modification, deletion) as described in the field information section.

3.1.9 BU19 [Underlying Backoffice Update BROADCAST]

3.1.9.1 Fingerprint

BROADCAST properties	
transaction type	BU19
calling sequence	omniapi_read_event_ext_ex or omniapi_read_event_block
struct name	underlying_update_bu4_bu19
info type	general

3.1.9.2 Related Messages

DQ19, the answer will take into account any modifications made.

3.1.9.3 Purpose

The Underlying Update broadcast is sent when a new underlying has been defined or updated in the central system.

Note: Preferably, the more modern BU121 should be used instead of BU19 (Delta Queries and Broadcasts concept).

3.1.9.4 Structure

The BU19 BROADCAST has the following structure:

```
struct underlying_update_bu4_bu19 {
   struct broadcast_type
   UINT16_T chg_type_n  // Change Type
   char[2] filler_2_s  // Filler
   struct da4_da19 {
      INT32_T subscription_price_i  // Subscription, Price
      INT32_T interest_rate_i  // Interest Rate
      UINT16_T commodity_n  // Commodity Code
      char[6] com_id_s  // Underlying Identity
      char[12] isin_code_s  // ISIN Code
      UINT16_T dec_in_price_n  // Decimals, Price
      char[8] date_release_s  // Date, Issue
      char[8] date_termination_s  // Date, Maturity
      char[8] date_dated_s  // Date, Dated
      char[32] name_s  // Name
      char[3] base_cur_s  // Currency, Trading
      UINT8_T deliverable_c  // Deliverable
      UINT16_T coupon_frequency_n  // Coupon Frequency
      INT64_T nominal_value_q  // Nominal Value
      UINT16_T day_count_n  // Day Count
      UINT16_T days_in_interest_year_n  // Days In Interest Year
      UINT32_T coupon_interest_i  // Coupon Interest
      UINT16_T coupon_settlement_days_n  // Coupon Settlement Days
      UINT8_T underlying_type_c  // Type, Underlying
      UINT8_T price_unit_c  // Price Unit, Underlying
      UINT16_T dec_in_nominal_n  // Decimals, Nominal
      UINT16_T state_number_n  // Trading State Number
      UINT16_T linked_commodity_n  // Linked Commodity Code
      UINT8_T fixed_income_type_c  // Fixed Income Type
      UINT8_T underlying_status_c  // Underlying Status
      char[6] underlying_issuer_s  // Underlying Issuer
      char[6] time_delivery_start_s  // Time, Delivery Start
      char[6] time_delivery_stop_s  // Time, Delivery Stop
      char[4] sector_code_s  // Sector Code
      UINT16_T items_n  // Items
      Array COUPON [max no: 80] {
         char[8] date_coupdiv_s  // Coupon/Dividend Date
         UINT32_T dividend_i  // Dividend
      }
      UINT8_T virtual_c  // Virtual
      char[4] member_circ_numb_s  // Member, Circular Number
      CHAR inv_scheme_c  // Investment Scheme
      char[8] date_closing_s  // Date, Closing
      char[8] date_last_s  // Date, Last
      char[2] country_id_s  // Name, Country
      UINT8_T cur_unit_c  // Currency Unit
      char[3] filler_3_s  // Filler
   }
```

```
}
```

3.1.9.5 Usage and Conditions

Change Type

states what type of update is at hand, as described in the field information section.

Trading State Number

will contain the immediate ISS.

3.1.10 BU20 [Instrument Class Backoffice Update BROADCAST]

3.1.10.1 Fingerprint

BROADCAST properties	
transaction type	BU20
calling sequence	omniapi_read_event_ext_ex or omniapi_read_event_block
struct name	inst_class_update_bu10_bu20
info type	general

3.1.10.2 Related Messages

DQ20, the answer will take into account any modifications made.

3.1.10.3 Purpose

The Instrument Class Update broadcast is sent when a new class has been defined or updated in the central system.

> **Note:** Preferably, the more modern BU123 should be used instead of BU20 (Delta Queries and Broadcasts concept).

3.1.10.4 Structure

The BU20 BROADCAST has the following structure:

```
struct inst_class_update_bu10_bu20 {
   struct broadcast_type
   UINT16_T chg_type_n  // Change Type
   char[2] filler_2_s  // Filler
   struct da10_da20 {
      struct series  // Named struct no: 50000
      struct upper_level_series
      INT32_T price_quot_factor_i  // Price, Quotation Factor
```

```
            INT32_T contract_size_i  // Contract Size
            INT32_T exerc_limit_i  // Exercise, Limit
            INT32_T redemption_value_i  // Redemption Value
            INT32_T min_qty_increment_i  // Minimum Quantity Increment
            UINT16_T derivate_level_n  // Derivate Level
            UINT16_T dec_in_strike_price_n  // Decimals, Strike Price
            UINT16_T dec_in_contr_size_n  // Decimals, Contract Size
            UINT16_T rnt_id_n  // Ranking Type
            UINT16_T dec_in_premium_n  // Decimals, Premium
            UINT16_T items_n  // Items
            Array ITEM [max no: 12] {
               struct tick_size
            }
            UINT16_T dec_in_deliv_n  // Decimals, Delivery
            UINT16_T items_block_n  // Item, Block
            Array BLOCK_SIZE [max no: 4] {
               INT64_T maximum_size_u  // Block Size, Maximum Volume
               UINT32_T minimum_size_n  // Block Size, Minimum Volume
               UINT32_T block_n  // Block Size
               UINT8_T lot_type_c  // Lot, Type
               char[3] filler_3_s  // Filler
            }
            UINT16_T cleared_dec_in_qty_n  // Decimals, Quantity
            UINT16_T virt_commodity_n  // Virtual Underlying
            UINT16_T dec_in_fixing_n  // Decimals, Fixing
            char[3] base_cur_s  // Currency, Trading
            UINT8_T traded_c  // Traded
            UINT8_T exerc_limit_unit_c  // Exercise, Limit Unit
            char[14] inc_id_s  // Instrument Class, Identity
            char[10] trc_id_s  // Trade Report Class
            char[32] name_s  // Name
            CHAR is_fractions_c  // Fraction, Premium
            UINT8_T price_format_c  // Premium/Price Format
            UINT8_T strike_price_format_c  // Strike Price, Format
            UINT8_T cabinet_format_c  // Cabinet Format
            UINT8_T price_unit_premium_c  // Price Unit, Premium
            UINT8_T price_unit_strike_c  // Price Unit, Strike
            char[32] settl_cur_id_s  // Currency, Settlement
            char[3] credit_class_s  // Credit Class
            char[12] csd_id_s  // CSD, Identity
            UINT8_T trd_cur_unit_c  // Traded Currency Unit
            UINT8_T collateral_type_c  // Collateral types
            UINT8_T fixing_req_c  // FIXING_REQ_C
            CHAR[2] mbs_id_s  // Minimum Bid Schedule
            char[12] valuation_group_id_s  // VAG, Identity ; Of type: VAG_ID_S
            char[3] filler_3_s  // Filler
         }
      }
```

3.1.10.5 Usage and Conditions

Change Type

states what type of update is at hand, as described in the field information section.

3.1.11 BU28 [Central Group Update BROADCAST]

3.1.11.1 Fingerprint

BROADCAST properties	
transaction type	BU28
calling sequence	omniapi_read_event_ext_ex or omniapi_read_event_block
struct name	central_group_update
info type	general

3.1.11.2 Related Messages

DQ28, the answer will take into account any modifications made.

3.1.11.3 Purpose

The Central Group Update broadcast is sent when a new central group has been defined or modified in the central system.

3.1.11.4 Structure

The BU28 BROADCAST has the following structure:

```
struct central_group_update {
   struct broadcast_type
   UINT16_T chg_type_n  // Change Type
   char[2] filler_2_s  // Filler
   struct da28 {
      char[12] central_group_s  // Central Group Name
      UINT16_T segment_number_n  // Segment Number
      UINT16_T items_n  // Items
      Array ITEM [max no: 30] {
         char[32] long_ins_id_s  // Series Name, Long
         UINT16_T leg_number_n  // Leg Number
         UINT8_T sort_type_c  // Sort Criteria
         CHAR filler_1_s  // Filler
      }
   }
}
```

3.1.11.5 Usage and Conditions

Segment Number

is used if the whole central group cannot be placed in one broadcast. If not all Series can be sent, the segment number is incremented with one until the whole Central Group is distributed. The last broadcast is sent with segment number = 0.

Series Name, Long

or short, may contain wildcard.

Change Type

states what type of update is at hand, as described in the field information section.

3.1.12 BU50 [Non-Settlement Days Update BROADCAST]

3.1.12.1 Fingerprint

BROADCAST properties	
transaction type	BU50
calling sequence	omniapi_read_event_ext_ex or omniapi_read_event_block
struct name	non_trad_settl_days_update_bu50
info type	general

3.1.12.2 Related Messages

DQ50, the answer will take into account any modifications made.

3.1.12.3 Purpose

This broadcast is sent when the non-trading days have changed in the central system.

3.1.12.4 Structure

The BU50 BROADCAST has the following structure:

```
struct non_trad_settl_days_update_bu50 {
   struct broadcast_type
   UINT16_T chg_type_n  // Change Type
   char[2] filler_2_s  // Filler
   struct da50 {
      struct series  // Named struct no: 50000
      char[8] date_non_trading_s  // Date, Non Trading
   }
}
```

3.1.12.5 Usage and conditions

Change Type

states what type of update is at hand, as described in the field information section.

3.1.13 BU87 [Market Maker Protection Update BROADCAST]

3.1.13.1 Fingerprint

BROADCAST properties	
transaction type	BU87
calling sequence	omniapi_read_event_ext_ex or omniapi_read_event_block
struct name	mm_protection_update
info type	dedicated

3.1.13.2 Related Messages

DC87, DQ87

3.1.13.3 Purpose

This broadcast is sent when the market maker protection parameters have been updated.

3.1.13.4 Structure

The BU87 BROADCAST has the following structure:

```
struct mm_protection_update {
   struct broadcast_type
   UINT16_T chg_type_n  // Change Type
   char[2] filler_2_s  // Filler
   struct da87 {
      INT64_T quantity_protection_q  // Quantity protection
      INT64_T delta_protection_q  // Delta protection
      INT32_T exposure_time_interval_i  // Exposure Time Interval
      INT32_T frozen_time_i  // Frozen Time
      UINT16_T commodity_n  // Commodity Code
      char[2] country_id_s  // Name, Country
      char[5] ex_customer_s  // Customer, Identity
      UINT8_T include_futures_c  // Include futures
      char[2] filler_2_s  // Filler
   }
}
```

3.1.14 BU120 [Delta Underlying Update VIB]

3.1.14.1 Fingerprint

VIB properties	
transaction type	BU120
calling sequence	omniapi_read_event_ext_ex or omniapi_read_event_block
struct name	The message complies with the VIM concept and has no top struct. The sequence of possible structs is described in the Structure section.
info type	general

3.1.14.2 Related Messages

DQ120

3.1.14.3 Purpose

This broadcast is used to send out information about a new underlying or an underlying that has been changed.

3.1.14.4 Structure

The BU120 VIB has the following structure:

```
struct broadcast_segment_hdr
struct item_hdr
struct sub_item_hdr
struct ns_delta_header  // Named struct no: 37001
Sequence {
   struct item_hdr
   Sequence {
      struct sub_item_hdr
      Choice {
         struct ns_remove  // Named struct no: 37002
         struct ns_underlying_basic  // Named struct no: 37201
         struct ns_fixed_income  // Named struct no: 37202
         struct ns_coupon_dates  // Named struct no: 37203
      }
   }
}
```

3.1.14.5 Usage and Conditions

For general information on the content of broadcasts and answers to queries, refer to section **DQ120**.

Broadcast BU120 will distribute all underlyings regardless of Status (active or suspended).

There may be consecutive broadcasts needed to disseminate all information. In this case the first broadcast will contain 1 in the Segment Number field. The field is then incremented by one in each of the following consecutive broadcasts.

The last broadcast will contain 0 (zero) in the Segment Number field.

If only one broadcast is needed, the Segment Number field will contain 0.

The broadcast does not contain any value in the full answer time-stamp.

Example

0 coupons

Only one broadcast is needed.

- Broadcast Segment Header (Segment Number = 0)
- Delta Header
- Underlying, Basic Data

Example

150 coupons

Three broadcasts are needed.

First broadcast

- Broadcast Segment Header (Segment Number = 1)
- Delta Header
- Underlying, Basic Data
- Underlying, Coupon Date (approximately first 50 coupons)

Second broadcast

- Broadcast Segment Header (Segment Number = 2)
- Delta Header
- Underlying, Coupon Date (approximately next 50 coupons)

Third broadcast

- Broadcast Segment Header (Segment Number = 0)
- Delta Header
- Underlying, Coupon Date (last around 50 coupons)

The NS_DELTA_HEADER structure will be the first item of the variable items.

3.1.15 BU121 [Delta Underlying Update for Back Office VIB]

3.1.15.1 Fingerprint

VIB properties	
transaction type	BU121
calling sequence	omniapi_read_event_ext_ex or omniapi_read_event_block
struct name	The message complies with the VIM concept and has no top struct. The sequence of possible structs is described in the Structure section.
info type	general

3.1.15.2 Related Messages

DQ121

3.1.15.3 Purpose

This broadcast is used to send out information about a new underlying or an underlying that has been changed.

3.1.15.4 Structure

The BU121 VIB has the following structure:

```
struct broadcast_segment_hdr
struct item_hdr
struct sub_item_hdr
struct ns_delta_header  // Named struct no: 37001
Sequence {
   struct item_hdr
   Sequence {
      struct sub_item_hdr
      Choice {
         struct ns_remove  // Named struct no: 37002
         struct ns_underlying_basic  // Named struct no: 37201
         struct ns_fixed_income  // Named struct no: 37202
         struct ns_coupon_dates  // Named struct no: 37203
      }
   }
}
```

3.1.15.5 Usage and Conditions

Broadcast BU121 (Back Office variant) will distribute all underlyings regardless of Status (active or suspended).

The NS_DELTA_HEADER structure will be the first item of the variable items.

For general information on the content of broadcasts and answers to queries, refer to section **DQ120**.

3.1.16 BU122 [Delta Instrument Class Update VIB]

3.1.16.1 Fingerprint

VIB properties	
transaction type	BU122
calling sequence	omniapi_read_event_ext_ex or omniapi_read_event_block
struct name	The message complies with the VIM concept and has no top struct. The sequence of possible structs is described in the Structure section.
info type	general

3.1.16.2 Related Messages

DQ122

3.1.16.3 Purpose

This broadcast is used to send out information about a new Instrument Class or an Instrument Class that has been changed.

3.1.16.4 Structure

The BU122 VIB has the following structure:

```
struct broadcast_segment_hdr
struct item_hdr
struct sub_item_hdr
struct ns_delta_header  // Named struct no: 37001
Sequence {
   struct item_hdr
   Sequence {
      struct sub_item_hdr
      Choice {
         struct ns_remove  // Named struct no: 37002
         struct ns_inst_class_basic  // Named struct no: 37101
         struct ns_price_tick  // Named struct no: 37102
         struct ns_block_size  // Named struct no: 37103
         struct ns_inst_class_secur  // Named struct no: 37105
      }
   }
}
```

3.1.16.5 Usage and Conditions

Broadcast BU122 will distribute all instrument classes regardless of Traded (Yes or No).

The NS_DELTA_HEADER structure will be the first item of the variable items.

For **NS Price Tick**, the instrument is traded in price or yield. **NS Price Tick Corr** gives the corresponding price if the trade is in yield, or the corresponding yield if the trade is in price.

For general information on the content of broadcasts and answers to queries, refer to section **DQ120**.

3.1.17 BU123 [Delta Instrument Class Update for Back Office VIB]

3.1.17.1 Fingerprint

VIB properties	
transaction type	BU123
calling sequence	omniapi_read_event_ext_ex or omniapi_read_event_block
struct name	The message complies with the VIM concept and has no top struct. The sequence of possible structs is described in the Structure section.
info type	general

3.1.17.2 Related Messages

DQ123

3.1.17.3 Purpose

This broadcast is used to send out information about a new Instrument Class or an Instrument Class that has been changed.

3.1.17.4 Structure

The BU123 VIB has the following structure:

```
struct broadcast_segment_hdr
struct item_hdr
struct sub_item_hdr
struct ns_delta_header  // Named struct no: 37001
Sequence {
   struct item_hdr
   Sequence {
      struct sub_item_hdr
      Choice {
         struct ns_remove  // Named struct no: 37002
         struct ns_inst_class_basic  // Named struct no: 37101
         struct ns_price_tick  // Named struct no: 37102
         struct ns_block_size  // Named struct no: 37103
         struct ns_inst_class_secur  // Named struct no: 37105
      }
   }
}
```

3.1.17.5 Usage and Conditions

Broadcast BU123 (Back Office variant) will distribute all instrument classes regardless of Traded (Yes or No).

The NS_DELTA_HEADER structure will be the first item of the variable items.

For **NS Price Tick**, the instrument is traded in price or yield. **NS Price Tick Corr** gives the corresponding price if the trade is in yield, or the corresponding yield if the trade is in price.

For general information on the content of broadcasts and answers to queries, refer to section **DQ120**.

3.1.18 BU124 [Delta Instrument Series Update VIB]

3.1.18.1 Fingerprint

VIB properties	
transaction type	BU124
calling sequence	omniapi_read_event_ext_ex or omniapi_read_event_block
struct name	The message complies with the VIM concept and has no top struct. The sequence of possible structs is described in the Structure section.
info type	general

3.1.18.2 Related Messages

DQ124

3.1.18.3 Purpose

This broadcast is used to send out information about a new Instrument Series or an Instrument Series that has been changed.

3.1.18.4 Structure

The BU124 VIB has the following structure:

```
struct broadcast_segment_hdr
struct item_hdr
struct sub_item_hdr
struct ns_delta_header  // Named struct no: 37001
Sequence {
   struct item_hdr
   Sequence {
      struct sub_item_hdr
      Choice {
         struct ns_remove  // Named struct no: 37002
         struct ns_inst_series_basic  // Named struct no: 37301
         struct ns_inst_series_basic_single  // Named struct no: 37302
```

```
            struct ns_inst_series_id  // Named struct no: 37310
         }
      }
   }
```

3.1.18.5 Usage and Conditions

Broadcast BU124 will distribute all series regardless of Last Trade Date, Traded (Yes or No), and Status (Active or Suspended).

The NS_DELTA_HEADER structure will be the first item of the variable items.

For general information on the content of broadcasts and answers to queries, refer to section **DQ120**.

3.1.19 BU125 [Delta Instrument Series Update for Back Office VIB]

3.1.19.1 Fingerprint

VIB properties	
transaction type	BU125
calling sequence	omniapi_read_event_ext_ex or omniapi_read_event_block
struct name	The message complies with the VIM concept and has no top struct. The sequence of possible structs is described in the Structure section.
info type	general

3.1.19.2 Related Messages

DQ125

3.1.19.3 Purpose

This broadcast is used to send out information about a new Instrument Series or an Instrument Series that has been changed.

3.1.19.4 Structure

The BU125 VIB has the following structure:

```
struct broadcast_segment_hdr
struct item_hdr
struct sub_item_hdr
struct ns_delta_header  // Named struct no: 37001
Sequence {
   struct item_hdr
   Sequence {
      struct sub_item_hdr
      Choice {
         struct ns_remove  // Named struct no: 37002
```

```
            struct ns_inst_series_basic  // Named struct no: 37301
            struct ns_inst_series_basic_single  // Named struct no: 37302
            struct ns_inst_series_bo  // Named struct no: 37306
            struct ns_inst_series_id  // Named struct no: 37310
         }
      }
   }
```

3.1.19.5 Usage and Conditions

Broadcast BU125 (Back Office variant) will distribute all series regardless of Last Trade Date, Traded (Yes or No), and Status (Active or Suspended).

The NS_DELTA_HEADER structure will be the first item of the variable items.

For general information on the content of broadcasts and answers to queries, refer to section **DQ120**.

3.1.20 BU126 [Combo Series Update VIB]

3.1.20.1 Fingerprint

VIB properties	
transaction type	BU126
calling sequence	omniapi_read_event_ext_ex or omniapi_read_event_block
struct name	The message complies with the VIM concept and has no top struct. The sequence of possible structs is described in the Structure section.
info type	general

3.1.20.2 Related Messages

Related queries: DQ120, DQ122, DQ124, DQ126 (and DQ121, DQ123, DQ125 which are Back Office related)

Related broadcasts: BU120, BU122, BU124 (and BU121, BU123, BU125 which are Back Office related)

3.1.20.3 Purpose

This broadcast is used to send out information about a new combination series or an combination series that has been changed.

3.1.20.4 Structure

The BU126 VIB has the following structure:

```
struct broadcast_segment_hdr
Sequence {
   struct item_hdr
   Sequence {
```

```
            struct sub_item_hdr
            Choice {
               struct ns_inst_series_basic  // Named struct no: 37301
               struct ns_combo_series_leg  // Named struct no: 37308
               struct ns_inst_series_id  // Named struct no: 37310
            }
         }
      }
```

3.1.20.5 Usage and Conditions

Note that this broadcast and the related DQ126 do not support the delta concept that the queries and broadcasts listed in "Related Messages" above support.

3.1.21 BU136 [Combo Series Update for Back Office VIB]

3.1.21.1 Fingerprint

VIB properties	
transaction type	BU136
calling sequence	omniapi_read_event_ext_ex or omniapi_read_event_block
struct name	The message complies with the VIM concept and has no top struct. The sequence of possible structs is described in the Structure section.
info type	general

3.1.21.2 Related Messages

DQ136

3.1.21.3 Purpose

This broadcast is used to send out information about a new combination series or a combination series that has been changed (also historical combination series).

3.1.21.4 Structure

The BU136 VIB has the following structure:

```
struct broadcast_segment_hdr
Sequence {
   struct item_hdr
   Sequence {
      struct sub_item_hdr
      Choice {
         struct ns_inst_series_basic  // Named struct no: 37301
         struct ns_combo_series_leg  // Named struct no: 37308
         struct ns_inst_series_id  // Named struct no: 37310
```

```
        }
    }
}
```

3.1.21.5 Usage and Conditions

Note:
This broadcast and the related DQ136 do not support the delta concept.

3.1.22 DC3 [Add TM Combo QUERY]

3.1.22.1 Fingerprint

QUERY properties	
transaction type	DC3
calling sequence	omniapi_query_ex
struct name	add_tm_combo
facility	EP5
partitioned	false
segmented	false
answers	DI3

ANSWER properties	
transaction type	DI3
struct name	answer_add_tm_combo
segmented	false

3.1.22.2 Purpose

The purpose of this transaction is to add a Tailor-Made Combination. The transaction is sent as a query, because the added Combination is returned as an answer.

3.1.22.3 Structure

The DC3 QUERY has the following structure:

```
struct add_tm_combo {
   struct transaction_type
   struct series  // Named struct no: 50000
   UINT16_T no_of_legs_n  // Legs, Number Of
   char[2] filler_2_s  // Filler
   Array ITEM [max no: 4] {
```

```
        struct series  // Named struct no: 50000
        UINT16_T ratio_n  // Ratio
        CHAR op_if_buy_c  // Operation if Buy
        CHAR op_if_sell_c  // Operation if Sell
    }
}
```

3.1.22.4 Usage and conditions

Series

in the transaction header is used only for RTR, and should be zeroed.

Operation if Buy

specifies whether to buy or sell the Series when buying the combination.

Operation if Sell

specifies whether to buy or sell the Series when selling the combination.

Example

This input creates a combination where instrument 1 is bought and instrument 2 is sold to a ratio 1 to 2 when buying the combination.

Number of Legs	2
Instrument 1:	
Ratio	1
Operation if Buy	B
Operation if Sell	S
Instrument 2:	
Ratio	2
Operation if Buy	S
Operation if Sell	B

3.1.22.5 Answer Structure

The DI3 ANSWER has the following structure:

```
struct answer_add_tm_combo {
    struct transaction_type
    struct series  // Named struct no: 50000
}
```

3.1.22.6 Answer, comments

The answer received contains the binary code of the created TM Combo as in BU2.

The DI3 answer can however also contain the binary code of an already existing Combo series corresponding to what is sent in the DC3, as well as an already existing Combo series that is a mirrored version of what is sent in the DC3. In order to handle order entry of Tailor Made Combos correctly, a front-end application must be able to handle a case where the DI3 answer contains the binary code of an existing mirrored combo series, and then enter the order on the opposite side as negative/positive depending on original entry details.

3.1.23 DC87 [Set Market Maker Protection TRANSACTION]

3.1.23.1 Fingerprint

TRANSACTION properties	
transaction type	DC87
calling sequence	omniapi_tx_ex
struct name	set_mm_protection
facility	EP0
partitioned	false

3.1.23.2 Related Messages

BU87, DQ87

3.1.23.3 Purpose

This transaction is used to set new market maker protection parameters per underlying.

3.1.23.4 Structure

The DC87 TRANSACTION has the following structure:

```
struct set_mm_protection {
   struct transaction_type
   struct series  // Named struct no: 50000
   struct da87 {
      INT64_T quantity_protection_q  // Quantity protection
      INT64_T delta_protection_q  // Delta protection
      INT32_T exposure_time_interval_i  // Exposure Time Interval
      INT32_T frozen_time_i  // Frozen Time
      UINT16_T commodity_n  // Commodity Code
      char[2] country_id_s  // Name, Country
      char[5] ex_customer_s  // Customer, Identity
      UINT8_T include_futures_c  // Include futures
      char[2] filler_2_s  // Filler
   }
}
```

3.1.23.5 Usage and conditions

Series

Should be filled with 0 (zero)

3.1.24 DQ2 [Series QUERY]

3.1.24.1 Fingerprint

QUERY properties	
transaction type	DQ2
calling sequence	omniapi_query_ex
struct name	query_series
facility	EP0
partitioned	false
segmented	true
answers	DA2

ANSWER properties	
transaction type	DA2
struct name	answer_series
segmented	true

3.1.24.2 Related Messages

BU2

3.1.24.3 Purpose

The purpose of this transaction is to retrieve all tradable series in the system, including combinations if any.

Note: Preferably, the more modern (Delta Queries and Broadcasts concept) DQ124 should be used instead of DQ2 single orders and DQ126 should be used instead of DQ2 + DQ5 for combinations.

3.1.24.4 Structure

The DQ2 QUERY has the following structure:

```
struct query_series {
   struct transaction_type
   struct series  // Named struct no: 50000
```

```
    UINT16_T segment_number_n  // Segment Number
    char[2] filler_2_s  // Filler
}
```

3.1.24.5 Usage and Conditions

Series

may be zeroed (all markets) or completed as **Country Number** and **Market Code** or a complete **Instrument Type**.

3.1.24.6 Answer Structure

The DA2 ANSWER has the following structure:

```
struct answer_series {
   struct transaction_type
   UINT16_T segment_number_n  // Segment Number
   UINT16_T items_n  // Items
   Array ITEM [max no: 300] {
      struct series  // Named struct no: 50000
      struct upper_level_series
      INT32_T contract_size_i  // Contract Size
      INT32_T price_quot_factor_i  // Price, Quotation Factor
      UINT32_T series_sequence_number_u  // Series, Sequence Number
      UINT16_T state_number_n  // Trading State Number
      UINT16_T step_size_multiple_n  // Tick Size, Multiple
      char[32] ins_id_s  // Series, Identity
      char[12] isin_code_s  // ISIN Code
      UINT8_T suspended_c  // Suspended
      char[8] date_last_trading_s  // Date, Last Trading
      char[6] time_last_trading_s  // Time, Last Trading
      char[8] settlement_date_s  // Date, Settlement
      char[8] start_date_s  // Date, Start
      char[8] end_date_s  // Date, End
      char[8] date_delivery_start_s  // Date, Delivery Start
      char[8] date_delivery_stop_s  // Date, Delivery Stop
      UINT8_T series_status_c  // Series, Status
      char[32] long_ins_id_s  // Series Name, Long
      char[8] date_first_trading_s  // Date, First Trading
      char[6] time_first_trading_s  // Time, First Trading
      UINT8_T traded_in_click_c  // Traded in GENIUM
      char[8] abbr_name_s  // Abbreviated Name
      char[6] stock_code_s  // Stock Code
      UINT8_T ext_info_source_c  // External Information Source
      char[8] effective_exp_date_s  // Effective Expiration Date
      char[2] filler_2_s  // Filler
   }
}
```

3.1.24.7 Answer, comments

The answer received contains a list of series. Each response is prefaced with the transaction type (DA2) and an item field specifying the number of records contained in the response.

Series

is returned regardless of the setting of the field traded_in_click_c.

Valid standard combination series will be included in the answer.

Upper Level Series

exists as a series if it is a traded, not expired series, otherwise ignore it.

Contract Size

This is the calculated contract size for the new series after an adjustment. For normal series (no adjustment) the Contract Size is 0. To receive the normal contract size and number of decimals in the contract size, use DQ10.

Price Quotation Factor

This is the calculated Price Quotation Factor for the new series after an adjustment. For normal series (no adjustment) the Price Quotation Factor is 0. To receive the normal Price Quotation Factor and number of decimals, use DQ10.

Trading State Number

will be 0 when sent in this answer. It will contain the immediate ISS only when distributing the instrument series in the broadcast BU2. To get the immediate ISS use the UQ15 query.

3.1.25 DQ3 [Instrument Type QUERY]

3.1.25.1 Fingerprint

QUERY properties	
transaction type	DQ3
calling sequence	omniapi_query_ex
struct name	query_instrument
facility	EP0
partitioned	false
segmented	true
answers	DA3

ANSWER properties	
transaction type	DA3
struct name	answer_instrument

ANSWER properties	
segmented	true

3.1.25.2 Purpose

The purpose of this transaction is to retrieve instrument types for all tradable series in the system, including combinations if any.

3.1.25.3 Structure

The DQ3 QUERY has the following structure:

```
struct query_instrument {
   struct transaction_type
   struct series  // Named struct no: 50000
   UINT16_T segment_number_n  // Segment Number
   char[2] filler_2_s  // Filler
}
```

3.1.25.4 Usage and conditions

Series

may be zeroed (all markets) or completed as **Country Number** and **Market Code** or a complete **Instrument Type**.

3.1.25.5 Answer Structure

The DA3 ANSWER has the following structure:

```
struct answer_instrument {
   struct transaction_type
   UINT16_T segment_number_n  // Segment Number
   UINT16_T items_n  // Items
   Array ITEM [max no: 100] {
      struct series  // Named struct no: 50000
      UINT32_T min_show_vol_u  // Order, Min Show Volume
      UINT16_T hidden_vol_meth_n  // Method, Hidden Volume
      UINT16_T pub_inf_id_n  // Public Order Info
      char[8] int_id_s  // Instrument, Identity
      char[32] name_s  // Name
      UINT8_T maintain_positions_c  // Maintain Positions
      UINT8_T traded_c  // Traded
      UINT8_T post_trade_proc_c  // Post Trade processed
      UINT8_T pos_handling_c  // Position handling
      UINT8_T directed_trade_information_c  // Directed Trade Information
      UINT8_T public_deal_information_c  // Public Deal Information
      char[2] filler_2_s  // Filler
   }
}
```

3.1.25.6 Answer, comments

The answer received contains a list of types. Each response is prefaced with the transaction type (DA3) and an item field specifying the number of records contained in the response.

3.1.26 DQ4 [Underlying QUERY]

3.1.26.1 Fingerprint

QUERY properties	
transaction type	DQ4
calling sequence	omniapi_query_ex
struct name	query_underlying
facility	EP0
partitioned	false
segmented	true
answers	DA4

ANSWER properties	
transaction type	DA4
struct name	answer_underlying
segmented	true

3.1.26.2 Related Messages

BU4

3.1.26.3 Purpose

The purpose of this transaction is to retrieve underlyings for all tradable series in the system.

Note: Preferably, the more modern DQ120 should be used instead of DQ4 (Delta Queries and Broadcasts concept).

3.1.26.4 Structure

The DQ4 QUERY has the following structure:

```
struct query_underlying {
   struct transaction_type
   struct series  // Named struct no: 50000
   UINT16_T segment_number_n  // Segment Number
   char[2] filler_2_s  // Filler
```

```
}
```

3.1.26.5 Usage and conditions

Series

may be zeroed (all markets) or completed as **Country Number** and **Market Code** or a complete **Instrument Type**.

3.1.26.6 Answer Structure

The DA4 ANSWER has the following structure:

```
struct answer_underlying {
   struct transaction type
   UINT16_T segment number n  // Segment Number
   UINT16_T items n  // Items
   Array ITEM [max no: 50] {
      INT32_T subscription price i  // Subscription, Price
      INT32_T interest rate i  // Interest Rate
      UINT16_T commodity n  // Commodity Code
      char[6] com id s  // Underlying Identity
      char[12] isin code s  // ISIN Code
      UINT16_T dec in price n  // Decimals, Price
      char[8] date release s  // Date, Issue
      char[8] date termination s  // Date, Maturity
      char[8] date dated s  // Date, Dated
      char[32] name s  // Name
      char[3] base cur s  // Currency, Trading
      UINT8_T deliverable c  // Deliverable
      UINT16_T coupon frequency n  // Coupon Frequency
      INT64_T nominal value q  // Nominal Value
      UINT16_T day count n  // Day Count
      UINT16_T days in interest year n  // Days In Interest Year
      UINT32_T coupon interest i  // Coupon Interest
      UINT16_T coupon settlement days n  // Coupon Settlement Days
      UINT8_T underlying type c  // Type, Underlying
      UINT8_T price unit c  // Price Unit, Underlying
      UINT16_T dec in nominal n  // Decimals, Nominal
      UINT16_T state number n  // Trading State Number
      UINT16_T linked commodity n  // Linked Commodity Code
      UINT8_T fixed income type c  // Fixed Income Type
      UINT8_T underlying status c  // Underlying Status
      char[6] underlying issuer s  // Underlying Issuer
      char[6] time delivery start s  // Time, Delivery Start
      char[6] time delivery stop s  // Time, Delivery Stop
      char[4] sector code s  // Sector Code
      UINT16_T items n  // Items
      Array COUPON [max no: 80] {
         char[8] date coupdiv s  // Coupon/Dividend Date
         UINT32_T dividend i  // Dividend
      }
      UINT8_T virtual c  // Virtual
```

```
        char[4] member_circ_numb_s  // Member, Circular Number
        CHAR inv_scheme_c  // Investment Scheme
        char[8] date_closing_s  // Date, Closing
        char[8] date_last_s  // Date, Last
        char[2] country_id_s  // Name, Country
        UINT8_T cur_unit_c  // Currency Unit
        char[3] filler_3_s  // Filler
    }
}
```

3.1.26.7 Answer, comments

For each underlying a record is received and they are prefaced with a transaction type (DA4) and an Item field, specifying the number of records.

Trading State Number

will be 0 (zero) in the answer of DQ4. When distributing the underlying in the broadcast BU4 the Trading State Number contains the immediate ISS only. To get the immediate ISS use the UQ15 query.

Decimals, Price

are used to interpret the Price Information for the Underlying.

3.1.27 DQ5 [Combination QUERY]

3.1.27.1 Fingerprint

QUERY properties	
transaction type	DQ5
calling sequence	omniapi_query_ex
struct name	query_combo
facility	EP0
partitioned	false
segmented	true
answers	DA5

ANSWER properties	
transaction type	DA5
struct name	answer_combo
segmented	true

3.1.27.2 Related Messages

BU5

3.1.27.3 Purpose

The reason for performing this query is to get the translation from each standard combination Series to the different single Series.

Preferably, the more modern DQ126 should be used instead of DQ2 + DQ5 for combinations (Delta Queries and Broadcasts concept).

3.1.27.4 Structure

The DQ5 QUERY has the following structure:

```
struct query_combo {
   struct transaction_type
   struct series  // Named struct no: 50000
   UINT16_T segment_number_n  // Segment Number
   char[2] filler_2_s  // Filler
}
```

3.1.27.5 Usage and conditions

Series

may be zeroed (all markets) or completed as **Country Number** and **Market Code** or a complete **Instrument Type**.

3.1.27.6 Answer Structure

The DA5 ANSWER has the following structure:

```
struct answer_combo {
   struct transaction_type
   UINT16_T segment_number_n  // Segment Number
   UINT8_T items_c  // Item
   CHAR filler_1_s  // Filler
   Array ITEM [max no: 100] {
      struct combo_series
      char[32] cbs_id_s  // Combo Series, Identity
      UINT8_T items_c  // Item
      char[3] filler_3_s  // Filler
      Array ITEM [max no: 4] {
         struct series  // Named struct no: 50000
         UINT16_T ratio_n  // Ratio
         CHAR op_if_buy_c  // Operation if Buy
         CHAR op_if_sell_c  // Operation if Sell
      }
   }
```

```
}
```

3.1.27.7 Answer, comments

For each Combo Series a record is received and they are prefaced with a Transaction Type (DA5) and an Item field, specifying the number of records.

3.1.28 DQ6 [Broker Signatures QUERY]

3.1.28.1 Fingerprint

QUERY properties	
transaction type	DQ6
calling sequence	omniapi_query_ex
struct name	query_broker
facility	EP0
partitioned	false
segmented	true
answers	DA6

ANSWER properties	
transaction type	DA6
struct name	answer_broker
segmented	true

3.1.28.2 Purpose

The identity of each single person authorized for trading is registered at the Exchange at the Instrument Type or Instrument Class level. It is then possible for the customer to request this information for his own staff.

3.1.28.3 Structure

The DQ6 QUERY has the following structure:

```
struct query_broker {
   struct transaction_type
   struct series  // Named struct no: 50000
   UINT16_T segment_number_n  // Segment Number
   char[2] country_id_s  // Name, Country
   char[5] ex_customer_s  // Customer, Identity
   char[3] filler_3_s  // Filler
}
```

3.1.28.4 Usage and Conditions

Series

Series may be zeroed (all markets) or completed as **Country Number** and **Market Code** or a complete **Instrument Type**.

3.1.28.5 Answer Structure

The DA6 ANSWER has the following structure:

```
struct answer_broker {
   struct transaction_type
   UINT16_T segment_number_n  // Segment Number
   char[2] country_id_s  // Name, Country
   char[5] ex_customer_s  // Customer, Identity
   CHAR filler_1_s  // Filler
   UINT16_T items_n  // Items
   Array ITEM [max no: 50] {
      char[5] user_id_s  // User
      UINT8_T program_trader_c  // Program Trader
      UINT16_T cst_id_n  // Customer Number
      UINT16_T usr_id_n  // User, Number
      UINT16_T items_n  // Items
      Array ITEM [max no: 100] {
         struct series  // Named struct no: 50000
      }
   }
}
```

3.1.28.6 Answer, comments

Series

Series in the answer can specify different levels of the instrument hierarchy. The user can be allowed to trade a number of both Instrument Types and Instrument Classes.

For an Instrument Type the Series structure is completed with Country, Market and Instrument Group.

For an Instrument Class the Series structure is completed with Country, Market, Instrument Group and Commodity.

For each broker at the customer, the broker ID and all legal instrument types it is authorized to trade in are returned. The response is prefaced with a Transaction Type (DA6) and an Item field specifying the number of records.

3.1.29 DQ7 [Market QUERY]

3.1.29.1 Fingerprint

QUERY properties	
transaction type	DQ7
calling sequence	omniapi_query_ex
struct name	query_market
facility	EP0
partitioned	false
segmented	true
answers	DA7

ANSWER properties	
transaction type	DA7
struct name	answer_market
segmented	true

3.1.29.2 Purpose

The purpose of this transaction is to retrieve markets for all tradable series in the system.

3.1.29.3 Structure

The DQ7 QUERY has the following structure:

```
struct query_market {
   struct transaction_type
   struct series  // Named struct no: 50000
   UINT16_T segment_number_n  // Segment Number
   char[2] filler_2_s  // Filler
}
```

3.1.29.4 Usage and Conditions

Series

may be zeroed (all markets) or completed as **Country Number** and **Market Code**.

3.1.29.5 Answer Structure

The DA7 ANSWER has the following structure:

```
struct answer_market {
   struct transaction_type
   UINT16_T segment_number_n  // Segment Number
   UINT16_T items_n  // Items
   Array ITEM [max no: 100] {
      UINT16_T normal_trading_days_n  // Normal Trading Days
      UINT16_T normal_settl_days_n  // Normal Settlement Days
      UINT16_T normal_clearing_days_n  // Normal Clearing Days
      UINT8_T country_c  // Country Number
      UINT8_T market_c  // Market Code
      char[32] name_s  // Name
      char[5] mar_id_s  // Market, Identity
      UINT8_T market_type_c  // Market, Type
      UINT8_T index_market_c  // Index Market
      char[15] bic_code_s  // BIC Code
      char[8] mic_code_s  // MIC Code
      char[2] filler_2_s  // Filler
   }
}
```

3.1.29.6 Answer, comments

The answer received contains a list of markets. Each response is prefaced with the transaction type (DA7) and an item field specifying the number of records contained in the response.

3.1.30 DQ8 [Instrument Group QUERY]

3.1.30.1 Fingerprint

QUERY properties	
transaction type	DQ8
calling sequence	omniapi_query_ex
struct name	query_instrument_group
facility	EP0
partitioned	false
segmented	true
answers	DA8

ANSWER properties	
transaction type	DA8
struct name	answer_instrument_group
segmented	true

3.1.30.2 Purpose

This transaction gets the valid instrument groups in binary format and their equivalent character representation.

3.1.30.3 Structure

The DQ8 QUERY has the following structure:

```
struct query_instrument_group {
   struct transaction_type
   struct series  // Named struct no: 50000
   UINT16_T segment_number_n  // Segment Number
   char[2] filler_2_s  // Filler
}
```

3.1.30.4 Usage and Conditions

Series

may be zeroed (all markets) or completed as Country Number and Market Code or a complete Instrument Type.

3.1.30.5 Answer Structure

The DA8 ANSWER has the following structure:

```
struct answer_instrument_group {
   struct transaction_type
   UINT16_T segment_number_n  // Segment Number
   UINT16_T items_n  // Items
   Array ITEM [max no: 100] {
      UINT16_T extended_info_n  // Extended Information
      UINT8_T instrument_group_c  // Instrument Group
      char[32] name_s  // Name
      char[3] ing_id_s  // Instrument Group Identity
      UINT8_T group_type_c  // Group, Type
      UINT8_T tailor_made_c  // Tailor Made
      UINT8_T option_type_c  // Option, Type
      UINT8_T option_style_c  // Option, Style
      UINT8_T warrant_c  // Warrant
      UINT8_T average_c  // Average
      UINT8_T average_period_c  // Average Period
      UINT8_T repo_type_c  // Repo Type
      UINT8_T buy_sell_back_c  // Buy Sell Back
      UINT8_T synthetic_type_c  // Type, Synthetic
      UINT8_T non_traded_ref_c  // Non Traded Reference
      UINT8_T future_styled_c  // Option, Future Styled
      UINT8_T when_issued_c  // When Issued
      UINT8_T is_exclusive_opening_sell_c  // Exclusive Open Sell
      UINT8_T knock_variant_c  // Knock Variant
      UINT8_T binary_variant_c  // Option, Binary Variant
      UINT8_T option_variant_c  // Option, Variant
      UINT8_T physical_delivery_c  // Physical Delivery
      UINT8_T forward_style_c  // Style, Forward
      UINT8_T swap_style_c  // Style, Swap
      UINT8_T maturity_c  // Maturity
```

```
        char[15] group_short_name_s  // Short Name, Instrument Group
        char[2] filler_2_s  // Filler
    }
}
```

3.1.30.6 Answer, comments

The answer received contains a list of instrument groups.

3.1.31 DQ9 [Series Backoffice QUERY]

3.1.31.1 Fingerprint

QUERY properties	
transaction type	DQ9
calling sequence	omniapi_query_ex
struct name	query_series
facility	EP0
partitioned	false
segmented	true
answers	DA9

ANSWER properties	
transaction type	DA9
struct name	answer_series_bo
segmented	true

3.1.31.2 Related Messages

BU9

3.1.31.3 Purpose

The purpose of this transaction is to retrieve all existing series in the system, including expired ones and other non-tradable series, for example, payment series.

Note that the same ASCII-name may be returned for different combinations, but with different binary codes and different last trading date.

Note: Preferably, the more modern DQ125 should be used instead of DQ9 (Delta Queries and Broadcasts concept).

3.1.31.4 Structure

The DQ9 QUERY has the following structure:

```
struct query_series {
   struct transaction_type
   struct series  // Named struct no: 50000
   UINT16_T segment_number_n  // Segment Number
   char[2] filler_2_s  // Filler
}
```

3.1.31.5 Usage and conditions

Series

Series may be zeroed (all markets) or completed as **Country Number** and **Market Code** or a complete **Instrument Type**.

3.1.31.6 Answer Structure

The DA9 ANSWER has the following structure:

```
struct answer_series_bo {
   struct transaction_type
   char[8] date_trading_s  // Date, Trading
   UINT16_T segment_number_n  // Segment Number
   UINT16_T items_n  // Items
   Array ITEM [max no: 330] {
      struct series  // Named struct no: 50000
      struct upper_level_series
      INT32_T contract_size_i  // Contract Size
      INT32_T price_quot_factor_i  // Price, Quotation Factor
      UINT16_T state_number_n  // Trading State Number
      char[32] ins_id_s  // Series, Identity
      char[12] isin_code_s  // ISIN Code
      UINT8_T stopped_by_issue_c  // Stopped By Issue
      char[12] isin_code_old_s  // ISIN Code, Old Series
      char[8] date_notation_s  // Date, Notation
      char[8] date_last_trading_s  // Date, Last Trading
      char[6] time_last_trading_s  // Time, Last Trading
      char[8] date_delivery_start_s  // Date, Delivery Start
      char[8] date_delivery_stop_s  // Date, Delivery Stop
      UINT8_T deliverable_c  // Deliverable
      UINT8_T suspended_c  // Suspended
      UINT8_T series_status_c  // Series, Status
      UINT8_T tm_template_c  // Template Series
      UINT8_T tm_series_c  // Tailor Made Series
      char[8] settlement_date_s  // Date, Settlement
      char[8] start_date_s  // Date, Start
      char[8] end_date_s  // Date, End
      UINT8_T accept_collateral_c  // Accepted as Collateral
      char[8] date_first_trading_s  // Date, First Trading
```

```
        char[6] time_first_trading_s  // Time, First Trading
        UINT8_T traded_in_click_c  // Traded in GENIUM
        UINT8_T traded_c  // Traded
        char[8] effective_exp_date_s  // Effective Expiration Date
        CHAR filler_1_s  // Filler
    }
}
```

3.1.31.7 Answer, comments

The answer received contains a list of series. Each response is prefaced with the transaction type (DA9) and an item field specifying the number of records contained in the response.

Series

is returned regardless of the setting of the field traded_in_click_c.

Contract Size

This is the calculated contract size for the new series after an adjustment. For normal series (no adjustment) the Contract Size is 0. To receive the normal contract size and number of decimals, use DQ20.

Price Quotation Factor

This is the calculated Price Quotation Factor for the new series after an adjustment. For normal series (no adjustment) the Price Quotation Factor is 0. To receive the normal Price Quotation Factor and number of decimals, use DQ20.

Trading State Number

will be 0 when sent in this answer. It will contain the immediate ISS only when distributing the instrument series in the broadcast BU9. To get the immediate ISS use the UQ15 query.

Stopped by Issue

is ‘Yes’ for the old series after adjustment.

3.1.32 DQ10 [Instrument Class QUERY]

3.1.32.1 Fingerprint

QUERY properties	
transaction type	DQ10
calling sequence	omniapi_query_ex
struct name	query_instrument_class
facility	EP0
partitioned	false
segmented	true
answers	DA10

ANSWER properties	
transaction type	DA10
struct name	answer_instrument_class
segmented	true

3.1.32.2 Related Messages

BU10

3.1.32.3 Purpose

The purpose of this transaction is to retrieve instrument classes for all tradable series in the system, including combinations if any.

> **Note:** Preferably, the more modern DQ122 should be used instead of DQ10 (Delta Queries and Broadcasts concept).

3.1.32.4 Structure

The DQ10 QUERY has the following structure:

```
struct query_instrument_class {
   struct transaction_type
   struct series  // Named struct no: 50000
   UINT16_T segment_number_n  // Segment Number
   char[2] filler_2_s  // Filler
}
```

3.1.32.5 Usage and conditions

Series

may be zeroed (all markets) or completed as **Country Number** and **Market Code** or a complete **Instrument Type**.

3.1.32.6 Answer Structure

The DA10 ANSWER has the following structure:

```
struct answer_instrument_class {
   struct transaction_type
   UINT16_T segment_number_n  // Segment Number
   UINT16_T items_n  // Items
   Array ITEM [max no: 145] {
      struct series  // Named struct no: 50000
      struct upper_level_series
      INT32_T price_quot_factor_i  // Price, Quotation Factor
      INT32_T contract_size_i  // Contract Size
```

```
            INT32_T exerc_limit_i  // Exercise, Limit
            INT32_T redemption_value_i  // Redemption Value
            INT32_T min_qty_increment_i  // Minimum Quantity Increment
            UINT16_T derivate_level_n  // Derivate Level
            UINT16_T dec_in_strike_price_n  // Decimals, Strike Price
            UINT16_T dec_in_contr_size_n  // Decimals, Contract Size
            UINT16_T rnt_id_n  // Ranking Type
            UINT16_T dec_in_premium_n  // Decimals, Premium
            UINT16_T items_n  // Items
            Array ITEM [max no: 12] {
               struct tick_size
            }
            UINT16_T dec_in_deliv_n  // Decimals, Delivery
            UINT16_T items_block_n  // Item, Block
            Array BLOCK_SIZE [max no: 4] {
               INT64_T maximum_size_u  // Block Size, Maximum Volume
               UINT32_T minimum_size_n  // Block Size, Minimum Volume
               UINT32_T block_n  // Block Size
               UINT8_T lot_type_c  // Lot, Type
               char[3] filler_3_s  // Filler
            }
            UINT16_T cleared_dec_in_qty_n  // Decimals, Quantity
            UINT16_T virt_commodity_n  // Virtual Underlying
            UINT16_T dec_in_fixing_n  // Decimals, Fixing
            char[3] base_cur_s  // Currency, Trading
            UINT8_T traded_c  // Traded
            UINT8_T exerc_limit_unit_c  // Exercise, Limit Unit
            char[14] inc_id_s  // Instrument Class, Identity
            char[10] trc_id_s  // Trade Report Class
            char[32] name_s  // Name
            CHAR is_fractions_c  // Fraction, Premium
            UINT8_T price_format_c  // Premium/Price Format
            UINT8_T strike_price_format_c  // Strike Price, Format
            UINT8_T cabinet_format_c  // Cabinet Format
            UINT8_T price_unit_premium_c  // Price Unit, Premium
            UINT8_T price_unit_strike_c  // Price Unit, Strike
            char[32] settl_cur_id_s  // Currency, Settlement
            char[3] credit_class_s  // Credit Class
            char[12] csd_id_s  // CSD, Identity
            UINT8_T trd_cur_unit_c  // Traded Currency Unit
            UINT8_T collateral_type_c  // Collateral types
            UINT8_T fixing_req_c  // FIXING_REQ_C
            CHAR[2] mbs_id_s  // Minimum Bid Schedule
            char[12] valuation_group_id_s  // VAG, Identity ; Of type: VAG_ID_S
            char[3] filler_3_s  // Filler
      }
   }
```

3.1.32.7 Answer, comments

The answer received contains a list of classes. Each response is prefaced with the transaction type (DA10) and an item field specifying the number of records contained in the response.

Decimals, Contract Size

applies to the fields **Contract Size** and **Price Quotation Factor**.

3.1.33 DQ12 [Account Type QUERY]

3.1.33.1 Fingerprint

QUERY properties	
transaction type	DQ12
calling sequence	omniapi_query_ex
struct name	query_account_type
facility	EP0
partitioned	false
segmented	true
answers	DA12

ANSWER properties	
transaction type	DA12
struct name	answer_account_type
segmented	true

3.1.33.2 Related Messages

BU12

3.1.33.3 Purpose

This query retrieves all existing account types in the system.

3.1.33.4 Structure

The DQ12 QUERY has the following structure:

```
struct query_account_type {
   struct transaction type
   struct series  // Named struct no: 50000
   UINT16_T segment_number_n  // Segment Number
   char[2] filler_2_s  // Filler
}
```

3.1.33.5 Answer Structure

The DA12 ANSWER has the following structure:

```
struct answer_account_type {
   struct transaction_type
   UINT16_T segment_number_n  // Segment Number
   UINT16_T items_n  // Items
   Array ITEM [max no: 100] {
      char[12] acc_type_s  // Account Type
      char[40] description_s  // Description
      UINT8_T open_close_c  // Open or Closed
      UINT8_T transitory_c  // Transitory
      UINT8_T market_maker_c  // Market Maker
      UINT8_T own_inventory_c  // Own Inventory
      UINT8_T exclusive_opening_sell_c  // Exclusive Opening Sell
      UINT8_T positions_allowed_c  // Positions, Allowed
      UINT8_T trades_allowed_c  // Trades, Allowed
      char[12] atr_id_s  // Account Type Rule
      CHAR origin_c  // Origin, Account Type
   }
}
```

3.1.34 DQ13 [Account Fee Type QUERY]

3.1.34.1 Fingerprint

QUERY properties	
transaction type	DQ13
calling sequence	omniapi_query_ex
struct name	query_account_fee_type
facility	EP0
partitioned	false
segmented	true
answers	DA13

ANSWER properties	
transaction type	DA13
struct name	answer_account_fee_type
segmented	true

3.1.34.2 Related Messages

BU13

3.1.34.3 Purpose

The purpose of this query is to get a description of all existing account fee types in the system.

3.1.34.4 Structure

The DQ13 QUERY has the following structure:

```
struct query_account_fee_type {
   struct transaction type
   struct series  // Named struct no: 50000
   UINT16_T segment_number_n  // Segment Number
   char[2] filler_2_s  // Filler
}
```

3.1.34.5 Answer Structure

The DA13 ANSWER has the following structure:

```
struct answer_account_fee_type {
   struct transaction type
   UINT16_T segment_number_n  // Segment Number
   UINT16_T items_n  // Items
   Array ITEM [max no: 100] {
      char[12] fee_type_s  // Account Fee Type
      char[40] description_s  // Description
   }
}
```

3.1.35 DQ14 [Underlying Adjustment QUERY]

3.1.35.1 Fingerprint

QUERY properties	
transaction type	DQ14
calling sequence	omniapi_query_ex
struct name	query_underlying_adjustment
facility	EP0
partitioned	false
segmented	true
answers	DA14

ANSWER properties	
transaction type	DA14
struct name	answer_underlying_adjustment
segmented	true

3.1.35.2 Purpose

The purpose of this query is to get information of underlying adjustments.

3.1.35.3 Structure

The DQ14 QUERY has the following structure:

```
struct query_underlying_adjustment {
   struct transaction type
   struct series  // Named struct no: 50000
   UINT16_T segment_number_n  // Segment Number
   char[8] date_adjust_s  // Date, Adjust
   char[2] filler_2_s  // Filler
}
```

3.1.35.4 Usage and Conditions

Date, Adjust

can be a historical date as well as the current date. However, only adjustments relevant for this date are returned in the answer.

3.1.35.5 Answer Structure

The DA14 ANSWER has the following structure:

```
struct answer_underlying_adjustment {
   struct transaction type
   UINT16_T segment_number_n  // Segment Number
   UINT16_T items_n  // Items
   Array ITEM [max no: 100] {
      UINT16_T adjust_ident_n  // Adjustment Identifier
      UINT16_T commodity_n  // Commodity Code
      char[8] date_adjust_s  // Date, Adjust
      char[8] date_conversion_s  // Date, Conversion
      UINT8_T deal_price_modifier_c  // Modifier, Deal Price
      UINT8_T contract_size_modifier_c  // Modifier, Contract Size
      UINT8_T strike_price_modifier_c  // Modifier, Strike Price
      UINT8_T contracts_modifier_c  // Modifier, Number of Contracts
      UINT8_T und_price_modifier_c  // Modifier, Underlying Price
      UINT8_T so_strike_price_modifier_c  // Modifier, Spin Off Strike Price
      UINT8_T so_contract_size_modifier_c  // Modifier, Contract Size
      UINT8_T so_deal_price_modifier_c  // Modifier, Spin Off Deal Price
      INT32_T deal_price_mod_factor_i  // Modifier Factor, Deal Price
      INT32_T contr_size_mod_factor_i  // Modifier Factor, Contract Size
      INT32_T strike_price_mod_factor_i  // Modifier Factor, Strike Price
      INT32_T contracts_mod_factor_i  // Modifier Factor, Number of Contracts
      INT32_T und_price_mod_factor_i  // Modifier Factor, Underlying Price
     INT32_T so_strike_price_mod_factor_i  // Modifier Factor, Spin Off Strike
 Price
```

```
      INT32_T so_contr_size_mod_factor_i  // Modifier Factor, Spin Off Contract
 Size
      INT32_T so_deal_price_mod_factor_i  // Modifier Factor, Spin Off Deal
Price
      INT32_T pqf_mod_factor_i  // Modifier Factor, Price Quotation Factor
     INT32_T so_pqf_mod_factor_i  // Modifier Factor, Spin Off Price Quotation
 Factor
      UINT16_T new_commodity_n  // Commodity Code, New
      UINT16_T so_commodity_n  // Commodity code, Spin Off
      UINT8_T pqf_modifier_c  // Modifier, Price Quotation Factor
     UINT8_T so_pqf_modifier_c  // Modifier, Spin Off Price Quotation Factor
      UINT8_T country_c  // Country Number
      UINT8_T market_c  // Market Code
      UINT8_T so_country_c  // Market, Spin Off
      UINT8_T so_market_c  // Market, Spin Off
      UINT8_T adjusted_c  // Adjusted Series
      UINT8_T spinoff_c  // Spinoff
      UINT16_T items_n  // Items
      char[2] filler_2_s  // Filler
      Array DELIVERY_CHANGE [max no: 20] {
         struct series  // Named struct no: 50000
         INT32_T contract_share_i  // Contract Share
      }
   }
}
```

3.1.35.6 Answer, comments

Adjustment identifier

is a unique number for each adjustment. If different conditions for different types of series exist for one underlying adjustment, several adjustment identifiers exist.

Series

means the new delivery underlying.

Contract Share

is the total contract size. The number of decimals in the contract share is defined in the Instrument Class.

3.1.36 DQ15 [Converted Series QUERY]

3.1.36.1 Fingerprint

QUERY properties	
transaction type	DQ15
calling sequence	omniapi_query_ex
struct name	query_converted_series
facility	EP0

QUERY properties	
partitioned	false
segmented	true
answers	DA15

ANSWER properties	
transaction type	DA15
struct name	answer_converted_series
segmented	true

3.1.36.2 Purpose

The purpose of this query is to get a conversion table between old and new series after an underlying adjustment. If the adjustment includes a spin off, an extra item for each spin off series is added in the answer.

3.1.36.3 Structure

The DQ15 QUERY has the following structure:

```
struct query_converted_series {
   struct transaction_type
   struct series  // Named struct no: 50000
   UINT16_T segment_number_n  // Segment Number
   UINT16_T adjust_ident_n  // Adjustment Identifier
}
```

3.1.36.4 Usage and Conditions

Adjustment Identifier

must be specified in the query. This is the unique identifier for the adjustrment retrieved in DQ14.

3.1.36.5 Answer Structure

The DA15 ANSWER has the following structure:

```
struct answer_converted_series {
   struct transaction_type
   UINT16_T segment_number_n  // Segment Number
   UINT16_T items_n  // Items
   Array ITEM [max no: 100] {
      UINT16_T adjust_ident_n  // Adjustment Identifier
      char[2] filler_2_s  // Filler
      INT32_T contract_size_i  // Contract Size
      INT32_T price_quot_factor_i  // Price, Quotation Factor
      struct old_series
      struct new_series
```

```
    }
}
```

3.1.36.6 Answer, comments

If the adjustment includes a spin off, an extra item for each spin off series is added in the answer:

- Item 1: Old Series 1 New Calculated Series 1
- Item 2: Old Series 1 Spin Off Series 1
- Item 3: Old Series 2 New Calculated Series 2
- Item 4: Old Series 2 Spin Off Series 2

Series, Old

is the series before adjustment.

Series, New

is the series after adjustment.

Contract Size

is the new contract size after adjustment. The number of decimals in the contract size is defined in the instrument class.

3.1.37 DQ18 [Non-Trading Days QUERY]

3.1.37.1 Fingerprint

QUERY properties	
transaction type	DQ18
calling sequence	omniapi_query_ex
struct name	query_non_trading_days
facility	EP0
partitioned	false
segmented	true
answers	DA18

ANSWER properties	
transaction type	DA18
struct name	answer_non_trading_days
segmented	true

3.1.37.2 Related Messages

BU18

3.1.37.3 Purpose

This query returns information about non-trading and/or settlement days.

3.1.37.4 Structure

The DQ18 QUERY has the following structure:

```
struct query_non_trading_days {
   struct transaction_type
   struct series  // Named struct no: 50000
   UINT16_T segment_number_n  // Segment Number
   char[2] filler_2_s  // Filler
}
```

3.1.37.5 Usage and Conditions

> **Note:**
> Weekends (normally Saturdays and Sundays) are not included in the list if they are always closed.
> The normal trading and settlement days are returned in the answer of DQ7 or DQ23.

Series

may be zeroed (all markets) or completed as **Country Number** and **Market Code**.

3.1.37.6 Answer Structure

The DA18 ANSWER has the following structure:

```
struct answer_non_trading_days {
   struct transaction_type
   UINT16_T segment_number_n  // Segment Number
   UINT16_T items_n  // Items
   Array ITEM [max no: 100] {
      UINT8_T country_c  // Country Number
      UINT8_T market_c  // Market Code
      char[8] date_non_trading_s  // Date, Non Trading
      UINT8_T closed_for_trading_c  // Closed, trading
      UINT8_T closed_for_settlement_c  // Closed, settlement
      UINT8_T closed_for_clearing_c  // Closed, clearing
      char[3] filler_3_s  // Filler
   }
}
```

3.1.38 DQ19 [Underlying Backoffice QUERY]

3.1.38.1 Fingerprint

QUERY properties	
transaction type	DQ19
calling sequence	omniapi_query_ex
struct name	query_underlying
facility	EP0
partitioned	false
segmented	true
answers	DA19

ANSWER properties	
transaction type	DA19
struct name	answer_underlying
segmented	true

3.1.38.2 Related Messages

BU19

3.1.38.3 Purpose

The purpose of this transaction is to retrieve underlyings for all series in the system.

Note: Preferably, the more modern DQ121 should be used instead of DQ19 (Delta Queries and Broadcasts concept).

3.1.38.4 Structure

The DQ19 QUERY has the following structure:

```
struct query_underlying {
   struct transaction_type
   struct series  // Named struct no: 50000
   UINT16_T segment_number_n  // Segment Number
   char[2] filler_2_s  // Filler
}
```

3.1.38.5 Usage and conditions

Series

may be zeroed (all markets) or completed as **Country Number** and **Market Code** or a complete **Instrument Type**.

3.1.38.6 Answer Structure

The DA19 ANSWER has the following structure:

```
struct answer_underlying {
   struct transaction_type
   UINT16_T segment_number_n  // Segment Number
   UINT16_T items_n  // Items
   Array ITEM [max no: 50] {
      INT32_T subscription_price_i  // Subscription, Price
      INT32_T interest_rate_i  // Interest Rate
      UINT16_T commodity_n  // Commodity Code
      char[6] com_id_s  // Underlying Identity
      char[12] isin_code_s  // ISIN Code
      UINT16_T dec_in_price_n  // Decimals, Price
      char[8] date_release_s  // Date, Issue
      char[8] date_termination_s  // Date, Maturity
      char[8] date_dated_s  // Date, Dated
      char[32] name_s  // Name
      char[3] base_cur_s  // Currency, Trading
      UINT8_T deliverable_c  // Deliverable
      UINT16_T coupon_frequency_n  // Coupon Frequency
      INT64_T nominal_value_q  // Nominal Value
      UINT16_T day_count_n  // Day Count
      UINT16_T days_in_interest_year_n  // Days In Interest Year
      UINT32_T coupon_interest_i  // Coupon Interest
      UINT16_T coupon_settlement_days_n  // Coupon Settlement Days
      UINT8_T underlying_type_c  // Type, Underlying
      UINT8_T price_unit_c  // Price Unit, Underlying
      UINT16_T dec_in_nominal_n  // Decimals, Nominal
      UINT16_T state_number_n  // Trading State Number
      UINT16_T linked_commodity_n  // Linked Commodity Code
      UINT8_T fixed_income_type_c  // Fixed Income Type
      UINT8_T underlying_status_c  // Underlying Status
      char[6] underlying_issuer_s  // Underlying Issuer
      char[6] time_delivery_start_s  // Time, Delivery Start
      char[6] time_delivery_stop_s  // Time, Delivery Stop
      char[4] sector_code_s  // Sector Code
      UINT16_T items_n  // Items
      Array COUPON [max no: 80] {
         char[8] date_coupdiv_s  // Coupon/Dividend Date
         UINT32_T dividend_i  // Dividend
      }
      UINT8_T virtual_c  // Virtual
      char[4] member_circ_numb_s  // Member, Circular Number
      CHAR inv_scheme_c  // Investment Scheme
```

```
        char[8] date_closing_s  // Date, Closing
        char[8] date_last_s  // Date, Last
        char[2] country_id_s  // Name, Country
        UINT8_T cur_unit_c  // Currency Unit
        char[3] filler_3_s  // Filler
    }
}
```

3.1.38.7 Answer, comments

For each underlying a record is received and they are prefaced with a transaction type (DA19) and an Item field, specifying the number of records.

Trading State Number

will be 0 (zero) in the answer of DQ19. When distributing the underlying in the broadcast BU19 the Trading State Number contains the immediate ISS only. To get the immediate ISS use the UQ15 query.

Decimals, Price

are used to interpret the Price Information for the Underlying.

3.1.39 DQ20 [Instrument Class Backoffice QUERY]

3.1.39.1 Fingerprint

QUERY properties	
transaction type	DQ20
calling sequence	omniapi_query_ex
struct name	query_instrument_class
facility	EP0
partitioned	false
segmented	true
answers	DA20

ANSWER properties	
transaction type	DA20
struct name	answer_instrument_class
segmented	true

3.1.39.2 Related Messages

BU20

3.1.39.3 Purpose

The purpose of this transaction is to retrieve instrument classes for all series in the system.

> **Note:** Preferably, the more modern DQ123 should be used instead of DQ20 (Delta Queries and Broadcasts concept).

3.1.39.4 Structure

The DQ20 QUERY has the following structure:

```
struct query_instrument_class {
   struct transaction_type
   struct series  // Named struct no: 50000
   UINT16_T segment_number_n  // Segment Number
   char[2] filler_2_s  // Filler
}
```

3.1.39.5 Usage and conditions

Series

may be zeroed (all markets) or completed as **Country Number** and **Market Code** or a complete **Instrument Type**.

3.1.39.6 Answer Structure

The DA20 ANSWER has the following structure:

```
struct answer_instrument_class {
   struct transaction_type
   UINT16_T segment_number_n  // Segment Number
   UINT16_T items_n  // Items
   Array ITEM [max no: 145] {
      struct series  // Named struct no: 50000
      struct upper_level_series
      INT32_T price_quot_factor_i  // Price, Quotation Factor
      INT32_T contract_size_i  // Contract Size
      INT32_T exerc_limit_i  // Exercise, Limit
      INT32_T redemption_value_i  // Redemption Value
      INT32_T min_qty_increment_i  // Minimum Quantity Increment
      UINT16_T derivate_level_n  // Derivate Level
      UINT16_T dec_in_strike_price_n  // Decimals, Strike Price
      UINT16_T dec_in_contr_size_n  // Decimals, Contract Size
      UINT16_T rnt_id_n  // Ranking Type
      UINT16_T dec_in_premium_n  // Decimals, Premium
      UINT16_T items_n  // Items
      Array ITEM [max no: 12] {
         struct tick_size
      }
      UINT16_T dec_in_deliv_n  // Decimals, Delivery
```

```
        UINT16_T items_block_n  // Item, Block
        Array BLOCK_SIZE [max no: 4] {
           INT64_T maximum_size_u  // Block Size, Maximum Volume
           UINT32_T minimum_size_n  // Block Size, Minimum Volume
           UINT32_T block_n  // Block Size
           UINT8_T lot_type_c  // Lot, Type
           char[3] filler_3_s  // Filler
        }
        UINT16_T cleared_dec_in_qty_n  // Decimals, Quantity
        UINT16_T virt_commodity_n  // Virtual Underlying
        UINT16_T dec_in_fixing_n  // Decimals, Fixing
        char[3] base_cur_s  // Currency, Trading
        UINT8_T traded_c  // Traded
        UINT8_T exerc_limit_unit_c  // Exercise, Limit Unit
        char[14] inc_id_s  // Instrument Class, Identity
        char[10] trc_id_s  // Trade Report Class
        char[32] name_s  // Name
        CHAR is_fractions_c  // Fraction, Premium
        UINT8_T price_format_c  // Premium/Price Format
        UINT8_T strike_price_format_c  // Strike Price, Format
        UINT8_T cabinet_format_c  // Cabinet Format
        UINT8_T price_unit_premium_c  // Price Unit, Premium
        UINT8_T price_unit_strike_c  // Price Unit, Strike
        char[32] settl_cur_id_s  // Currency, Settlement
        char[3] credit_class_s  // Credit Class
        char[12] csd_id_s  // CSD, Identity
        UINT8_T trd_cur_unit_c  // Traded Currency Unit
        UINT8_T collateral_type_c  // Collateral types
        UINT8_T fixing_req_c  // FIXING_REQ_C
        CHAR[2] mbs_id_s  // Minimum Bid Schedule
        char[12] valuation_group_id_s  // VAG, Identity ; Of type: VAG_ID_S
        char[3] filler_3_s  // Filler
    }
}
```

3.1.39.7 Answer, comments

The answer received contains a list of classes. Each response is prefaced with the transaction type (DA20) and an item field specifying the number of records contained in the response.

Decimals, Contract Size

applies to the fields **Contract Size** and **Price Quotation Factor**.

3.1.40 DQ22 [Instrument Type Backoffice QUERY]

3.1.40.1 Fingerprint

QUERY properties	
transaction type	DQ22
calling sequence	omniapi_query_ex

QUERY properties	
struct name	query_instrument
facility	EP0
partitioned	false
segmented	true
answers	DA22

ANSWER properties	
transaction type	DA22
struct name	answer_instrument
segmented	true

3.1.40.2 Purpose

The purpose of this transaction is to retrieve all instrument types in the system.

3.1.40.3 Structure

The DQ22 QUERY has the following structure:

```
struct query_instrument {
   struct transaction_type
   struct series  // Named struct no: 50000
   UINT16_T segment_number_n  // Segment Number
   char[2] filler_2_s  // Filler
}
```

3.1.40.4 Usage and conditions

Series

may be zeroed (all markets) or completed as **Country Number** and **Market Code** or a complete **Instrument Type**.

3.1.40.5 Answer Structure

The DA22 ANSWER has the following structure:

```
struct answer_instrument {
   struct transaction_type
   UINT16_T segment_number_n  // Segment Number
   UINT16_T items_n  // Items
   Array ITEM [max no: 100] {
      struct series  // Named struct no: 50000
      UINT32_T min_show_vol_u  // Order, Min Show Volume
      UINT16_T hidden_vol_meth_n  // Method, Hidden Volume
```

```
        UINT16_T pub_inf_id_n  // Public Order Info
        char[8] int_id_s  // Instrument, Identity
        char[32] name_s  // Name
        UINT8_T maintain_positions_c  // Maintain Positions
        UINT8_T traded_c  // Traded
        UINT8_T post_trade_proc_c  // Post Trade processed
        UINT8_T pos_handling_c  // Position handling
        UINT8_T directed_trade_information_c  // Directed Trade Information
        UINT8_T public_deal_information_c  // Public Deal Information
        char[2] filler_2_s  // Filler
    }
}
```

3.1.40.6 Answer, comments

The answer received contains a list of types. Each response is prefaced with the transaction type (DA22) and an item field specifying the number of records contained in the response.

3.1.41 DQ23 [Market Backoffice QUERY]

3.1.41.1 Fingerprint

QUERY properties	
transaction type	DQ23
calling sequence	omniapi_query_ex
struct name	query_market
facility	EP0
partitioned	false
segmented	true
answers	DA23

ANSWER properties	
transaction type	DA23
struct name	answer_market
segmented	true

3.1.41.2 Purpose

The purpose of this query is to retrieve markets for all series in the system.

3.1.41.3 Structure

The DQ23 QUERY has the following structure:

```
struct query_market {
   struct transaction_type
   struct series  // Named struct no: 50000
   UINT16_T segment_number_n  // Segment Number
   char[2] filler_2_s  // Filler
}
```

3.1.41.4 Usage and Conditions

Series

may be zeroed (all markets) or completed as **Country Number** and **Market Code**.

3.1.41.5 Answer Structure

The DA23 ANSWER has the following structure:

```
struct answer_market {
   struct transaction_type
   UINT16_T segment_number_n  // Segment Number
   UINT16_T items_n  // Items
   Array ITEM [max no: 100] {
      UINT16_T normal_trading_days_n  // Normal Trading Days
      UINT16_T normal_settl_days_n  // Normal Settlement Days
      UINT16_T normal_clearing_days_n  // Normal Clearing Days
      UINT8_T country_c  // Country Number
      UINT8_T market_c  // Market Code
      char[32] name_s  // Name
      char[5] mar_id_s  // Market, Identity
      UINT8_T market_type_c  // Market, Type
      UINT8_T index_market_c  // Index Market
      char[15] bic_code_s  // BIC Code
      char[8] mic_code_s  // MIC Code
      char[2] filler_2_s  // Filler
   }
}
```

3.1.41.6 Answer, comments

The answer received contains a list of markets. Each response is prefaced with the transaction type (DA23) and an item field specifying the number of records contained in the response.

3.1.42 DQ24 [Exchange QUERY]

3.1.42.1 Fingerprint

QUERY properties	
transaction type	DQ24
calling sequence	omniapi_query_ex

QUERY properties	
struct name	query_exchange_dq24
facility	EP0
partitioned	false
segmented	true
answers	DA24

ANSWER properties	
transaction type	DA24
struct name	answer_exchange_da24
segmented	true

3.1.42.2 Purpose

This query provides information on all exchanges in the system.

3.1.42.3 Structure

The DQ24 QUERY has the following structure:

```
struct query_exchange_dq24 {
   struct transaction_type
   struct series  // Named struct no: 50000
   UINT16_T segment_number_n  // Segment Number
   char[2] filler_2_s  // Filler
}
```

3.1.42.4 Usage and conditions

Series

must be zeroed.

3.1.42.5 Answer Structure

The DA24 ANSWER has the following structure:

```
struct answer_exchange_da24 {
   struct transaction_type
   UINT16_T segment_number_n  // Segment Number
   UINT16_T items_n  // Items
   Array ITEM [max no: 100] {
      struct da24 {
         UINT8_T country_c  // Country Number
         CHAR opra_indicator_c  // OPRA Indicator
         char[32] name_s  // Name
```

```
            char[4] exchange_short_s  // Exchange, Short Name
            char[2] country_id_s  // Name, Country
            char[40] tz_exchange_s  // Time Zone, Exchange
            char[12] master_clh_id_s  // Master CLH, Identity
            char[2] country_s  // Country
            char[8] date_implementation_s  // Date, Implementation
            char[2] filler_2_s  // Filler
        }
    }
}
```

3.1.42.6 Answer, comments

The answer received contains a list of exchanges. Each response is prefaced with the Transaction Type (DA24) and an Item field specifying the number of records included in the response.

3.1.43 DQ28 [Central Group QUERY]

3.1.43.1 Fingerprint

QUERY properties	
transaction type	DQ28
calling sequence	omniapi_query_ex
struct name	query_central_group
facility	EP0
partitioned	false
segmented	true
answers	DA28

ANSWER properties	
transaction type	DA28
struct name	answer_central_group
segmented	true

3.1.43.2 Related Messages

BU28

3.1.43.3 Purpose

The purpose of this transaction is to retrieve the centrally defined display groups. A group contains a list of series names grouped together.

3.1.43.4 Structure

The DQ28 QUERY has the following structure:

```
struct query_central_group {
   struct transaction type
   struct series  // Named struct no: 50000
   UINT16_T segment_number_n  // Segment Number
   char[2] filler_2_s  // Filler
}
```

3.1.43.5 Usage and Conditions

Series

May be zeroed (all markets) or completed as **Country Number** and **Market Code** or a complete **Instrument Type**.

3.1.43.6 Answer Structure

The DA28 ANSWER has the following structure:

```
struct answer_central_group {
   struct transaction type
   UINT16_T segment_number_n  // Segment Number
   UINT16_T items_n  // Items
   Array ITEM [max no: 1000] {
      char[12] central_group_s  // Central Group Name
      UINT16_T leg_number_n  // Leg Number
      UINT8_T sort_type_c  // Sort Criteria
      CHAR filler_1_s  // Filler
      char[32] long_ins_id_s  // Series Name, Long
   }
}
```

3.1.43.7 Answer, comments

Central Group Name

is repeated for every series contained in the group.

Series Name, Long

may contain wildcards:

- * for an optional number of characters
- ? for one character

Name

The display name is repeated for every series contained in the group.

The answer received contains a list of series and the central group the series is connected to.

Each response is prefaced with the Transaction Type (DA28) and an Item field specifying the number of records contained in the response.

3.1.44 DQ29 [Trading State QUERY]

3.1.44.1 Fingerprint

QUERY properties	
transaction type	DQ29
calling sequence	omniapi_query_ex
struct name	query_trading_state
facility	EP0
partitioned	false
segmented	true
answers	DA29

ANSWER properties	
transaction type	DA29
struct name	answer_trading_state
segmented	true

3.1.44.2 Purpose

The purpose of this transaction is to retrieve the definitions of existing Trading States.

3.1.44.3 Structure

The DQ29 QUERY has the following structure:

```
struct query_trading_state {
   struct transaction_type
   struct series  // Named struct no: 50000
   UINT16_T segment_number_n  // Segment Number
   char[2] filler_2_s  // Filler
}
```

3.1.44.4 Usage and Conditions

Series

All fields in the series must be set to 0 (zero).

3.1.44.5 Answer Structure

The DA29 ANSWER has the following structure:

```
struct answer_trading_state {
   struct transaction_type
   UINT16_T segment_number_n  // Segment Number
   UINT16_T items_n  // Items
   Array ITEM [max no: 100] {
      char[20] state_name_s  // Trading State Name
      UINT16_T state_number_n  // Trading State Number
      UINT16_T iss_def_warning_interval_n  // Warning Interval, Default for
ISS
      UINT16_T iss_def_num_of_warnings_n  // Number of Warnings, Default for
ISS
      UINT16_T state_type_number_n  // State Type Number
      UINT8_T continues_matching_c  // Matching, Open
      UINT8_T trading_end_c  // End of Trading
      UINT8_T price_quotation_required_c  // Price, Quotation Required
      UINT8_T market_orders_allowed_c  // Market Orders, Allowed
      UINT8_T fill_or_kill_allowed_c  // Fill or Kill Allowed
      UINT8_T fill_and_kill_allowed_c  // Fill and Kill Allowed
      UINT8_T edited_ob_changes_avail_c  // Edited Price Information Available
      UINT8_T ob_changes_avail_c  // Order Book Changes Available
      UINT8_T external_full_depth_c  // Full Depth, External
      UINT8_T internal_full_depth_c  // Full Depth, Internal
      UINT8_T end_of_clearing_day_c  // End of Clearing Day
      UINT8_T odd_lot_allwd_c  // Odd Lot, Allowed
      UINT8_T action_odd_lot_c  // Odd Lot, Action
      UINT8_T state_priority_c  // State Priority
      char[2] filler_2_s  // Filler
   }
}
```

3.1.44.6 Answer, comments

The answer received contains a list of existing trading states. Each response is prefaced with the Transaction Type (DA29) and an Item field specifying the number of records contained in the response.

3.1.45 DQ30 [User Type Info QUERY]

3.1.45.1 Fingerprint

QUERY properties	
transaction type	DQ30
calling sequence	omniapi_query_ex
struct name	query_user_type_info
facility	EP0

QUERY properties	
partitioned	false
segmented	true
answers	DA30

ANSWER properties	
transaction type	DA30
struct name	answer_user_type_info
segmented	true

3.1.45.2 Purpose

The Query User Type Info Transaction provides information on user type and legal transactions and broadcasts authorized for the querying user.

3.1.45.3 Structure

The DQ30 QUERY has the following structure:

```
struct query_user_type_info {
   struct transaction_type
   struct series  // Named struct no: 50000
   UINT16_T segment_number_n  // Segment Number
   char[2] filler_2_s  // Filler
}
```

3.1.45.4 Usage and Conditions

Series

All fields in the series must be set to 0 (zero).

3.1.45.5 Answer Structure

The DA30 ANSWER has the following structure:

```
struct answer_user_type_info {
   struct transaction_type
   UINT16_T segment_number_n  // Segment Number
   UINT16_T items_n  // Items
   char[5] ust_id_s  // User Type, Identity
   UINT8_T ext_or_int_c  // User Type
   UINT8_T is_trader_c  // Trader
   UINT8_T program_trader_c  // Program Trader
   UINT8_T trader_authorization_c  // Trader, Authorization
   char[3] filler_3_s  // Filler
   Array ITEM [max no: 100] {
```

```
        struct transaction_type
        UINT8_T trans_or_bdx_c  // Transaction or Broadcast
        char[3] filler_3_s  // Filler
    }
}
```

3.1.45.6 Answer, comments

The answer received contains a list of of legal transactions/broadcasts. Each response is prefaced with the Transaction Type (DA30) and an Item field specifying the number of records included in the response.

3.1.46 DQ33 [Currency QUERY]

3.1.46.1 Fingerprint

QUERY properties	
transaction type	DQ33
calling sequence	omniapi_query_ex
struct name	query_currency
facility	EP0
partitioned	false
segmented	true
answers	DA33

ANSWER properties	
transaction type	DA33
struct name	answer_currency
segmented	true

3.1.46.2 Purpose

The purpose of this transaction is to get valid currencies.

3.1.46.3 Structure

The DQ33 QUERY has the following structure:

```
struct query_currency {
   struct transaction_type
   struct series  // Named struct no: 50000
   UINT16_T segment_number_n  // Segment Number
   char[2] filler_2_s  // Filler
}
```

3.1.46.4 Usage and Conditions

Series

All fields in the series must be set to 0 (zero).

3.1.46.5 Answer Structure

The DA33 ANSWER has the following structure:

```
struct answer_currency {
   struct transaction_type
   UINT16_T segment_number_n  // Segment Number
   UINT16_T items_n  // Items
   Array ITEM [max no: 100] {
      UINT16_T sec_rel_primary_n  // Relation to Primary, Secondary
      UINT16_T third_rel_primary_n  // Relation to Primary, Tertiary
      char[3] base_cur_s  // Currency, Trading
      char[15] pri_unit_s  // Unit, Primary
      char[15] sec_unit_s  // Unit, Secondary
      char[15] third_unit_s  // Unit, Tertiary
      char[5] pri_not_s  // Notation, Primary
      char[5] sec_not_s  // Notation, Secondary
      char[5] third_not_s  // Notation, Tertiary
      UINT8_T acc_as_pay_c  // Accepted As Payment
      UINT8_T currency_format_c  // Currency Format
      char[3] filler_3_s  // Filler
   }
}
```

3.1.46.6 Answer, comments

The answer received contains a list of currencies. Each response is prefaced with the Transaction Type (DA33) and an Item field specifying the number of records contained in the response.

3.1.47 DQ34 [Account Type Rule QUERY]

3.1.47.1 Fingerprint

QUERY properties	
transaction type	DQ34
calling sequence	omniapi_query_ex
struct name	query_account_type_rule
facility	EP0
partitioned	false
segmented	true
answers	DA34

ANSWER properties	
transaction type	DA34
struct name	answer_account_type_rule
segmented	true

3.1.47.2 Purpose

The purpose of this transaction is to get account type rule for each account type.

3.1.47.3 Structure

The DQ34 QUERY has the following structure:

```
struct query_account_type_rule {
   struct transaction_type
   struct series  // Named struct no: 50000
   UINT16_T segment_number_n  // Segment Number
   char[2] filler_2_s  // Filler
}
```

3.1.47.4 Usage and conditions

Series

may be zeroed.

3.1.47.5 Answer Structure

The DA34 ANSWER has the following structure:

```
struct answer_account_type_rule {
   struct transaction_type
   UINT16_T segment_number_n  // Segment Number
   UINT16_T items_n  // Items
   Array ITEM [max no: 100] {
      char[12] atr_id_s  // Account Type Rule
      UINT8_T create_over_api_c  // Create Over API
      UINT8_T activate_at_reg_c  // Activate At Registration
      UINT16_T account_field_no_n  // Account Field Number
      UINT8_T attribute_rule_c  // Attribute Rule
      char[3] filler_3_s  // Filler
   }
}
```

3.1.47.6 Answer, comments

The answer received contains a list of rules. Each response is prefaced with the Transaction Type (DA34) and an Item field specifying the number of records contained in the response.

3.1.48 DQ35 [Participant QUERY]

3.1.48.1 Fingerprint

QUERY properties	
transaction type	DQ35
calling sequence	omniapi_query_ex
struct name	query_participant
facility	EP0
partitioned	false
segmented	true
answers	DA35

ANSWER properties	
transaction type	DA35
struct name	answer_participant
segmented	true

3.1.48.2 Purpose

The purpose of this query is to get all participants (members).

3.1.48.3 Structure

The DQ35 QUERY has the following structure:

```
struct query_participant {
   struct transaction type
   struct series  // Named struct no: 50000
   UINT16_T segment_number_n  // Segment Number
   char[2] filler_2_s  // Filler
}
```

3.1.48.4 Usage and conditions

Series

may be zeroed.

3.1.48.5 Answer Structure

The DA35 ANSWER has the following structure:

```
struct answer_participant {
   struct transaction_type
   UINT16_T segment_number_n  // Segment Number
   UINT16_T items_n  // Items
   Array ITEM [max no: 500] {
      char[2] country_id_s  // Name, Country
      char[5] ex_customer_s  // Customer, Identity
      char[15] bic_code_s  // BIC Code
      char[32] name_s  // Name
      UINT8_T swift_member_c  // SWIFT Member
      char[12] clh_id_s  // Clearinghouse
      UINT8_T trading_access_c  // Trading, Access
      CHAR cl_status_c  // CL, Status
      char[3] filler_3_s  // Filler
   }
}
```

3.1.48.6 Answer, comments

The answer received contains a list of all participants (members). Each response is prefaced with the transaction type (DA35) and an item field specifying the number of records contained in the response.

3.1.49 DQ44 [Legal Account Instrument QUERY]

3.1.49.1 Fingerprint

QUERY properties	
transaction type	DQ44
calling sequence	omniapi_query_ex
struct name	query_legal_account_instrument
facility	EP0
partitioned	false
segmented	true
answers	DA44

ANSWER properties	
transaction type	DA44
struct name	answer_legal_account_instrument
segmented	true

3.1.49.2 Purpose

This query returns a list of Account Types. Account Types are used to classify different accounts in GENIUM INET Clearing.

3.1.49.3 Structure

The DQ44 QUERY has the following structure:

```
struct query_legal_account_instrument {
   struct transaction type
   struct series  // Named struct no: 50000
   UINT16_T segment number n  // Segment Number
   char[2] filler_2_s  // Filler
}
```

3.1.49.4 Answer Structure

The DA44 ANSWER has the following structure:

```
struct answer_legal_account_instrument {
   struct transaction type
   UINT16_T segment number n  // Segment Number
   UINT16_T items n  // Items
   Array ITEM [max no: 500] {
      struct series  // Named struct no: 50000
      char[12] acc type s  // Account Type
   }
}
```

3.1.50 DQ45 [Trade Report Type QUERY]

3.1.50.1 Fingerprint

QUERY properties	
transaction type	DQ45
calling sequence	omniapi_query_ex
struct name	query_trade_report_types
facility	EP0
partitioned	false
segmented	true
answers	DA45

ANSWER properties	
transaction type	DA45
struct name	answer_trade_report_types
segmented	true

3.1.50.2 Purpose

This query is used to retrieve all trade report types.

3.1.50.3 Structure

The DQ45 QUERY has the following structure:

```
struct query_trade_report_types {
   struct transaction_type
   struct series  // Named struct no: 50000
   UINT16_T segment_number_n  // Segment Number
   char[2] filler_2_s  // Filler
}
```

3.1.50.4 Usage and conditions

Series

has no implication on the selection of items returned. All available trade report types are returned.

3.1.50.5 Answer Structure

The DA45 ANSWER has the following structure:

```
struct answer_trade_report_types {
   struct transaction_type
   UINT16_T segment_number_n  // Segment Number
   UINT16_T items_n  // Items
   Array ITEM [max no: 200] {
      INT64_T initial_trr_min_value_u  // Initial Trade Report, Minimum Order
 Value.
      char[10] trc_id_s  // Trade Report Class
      char[4] trr_id_s  // Trade Report, Identity
      char[32] condition_s  // Trade Report Description
      UINT8_T authorized_c  // Authorized
      UINT8_T ext_t_state_c  // Trade Report Type
      UINT8_T allow_interbank_c  // Allow interbank
      UINT8_T allow_within_participant_c  // Allow within participant
      UINT8_T cbo_trade_report_c  // Combo Trade Report
      UINT8_T allow_non_std_settlement_c  // Allow non standard settlement
      UINT8_T time_of_agree_req_c  // Time of agreement required
      UINT8_T time_of_agree_gran_c  // Time of agreement granularity
      char[2] filler_2_s  // Filler
   }
}
```

3.1.50.6 Answer, comments

After a successful DQ45, information about Trade Report Types is returned to the sender.

3.1.51 DQ46 [Deal Source QUERY]

3.1.51.1 Fingerprint

QUERY properties	
transaction type	DQ46
calling sequence	omniapi_query_ex
struct name	query_deal_source
facility	EP0
partitioned	false
segmented	true
answers	DA46

ANSWER properties	
transaction type	DA46
struct name	answer_deal_source
segmented	true

3.1.51.2 Purpose

The purpose of this transaction is to receive all available deal sources.

3.1.51.3 Structure

The DQ46 QUERY has the following structure:

```
struct query_deal_source {
   struct transaction type
   struct series  // Named struct no: 50000
   UINT16_T segment number n  // Segment Number
   char[2] filler 2 s  // Filler
}
```

3.1.51.4 Answer Structure

The DA46 ANSWER has the following structure:

```
struct answer_deal_source {
   struct transaction type
   UINT16_T segment number n  // Segment Number
   UINT16_T items n  // Items
   Array ITEM [max no: 100] {
      INT64_T ds attribute q  // Deal Source Attribute
      INT16_T deal source n  // Deal Source
```

```
        char[128] desc_long_s  // Description, Long
        char[32] desc_abbreviated_s  // Description, Abbreviated
        char[2] filler_2_s  // Filler
    }
}
```

3.1.51.5 Answer, comments

The answer received contains a list of all available deal sources. Each response is prefaced with the transaction type (DA46).

3.1.52 DQ50 [Non-Settlement Days QUERY]

3.1.52.1 Fingerprint

QUERY properties	
transaction type	DQ50
calling sequence	omniapi_query_ex
struct name	query_non_trad_settl_days
facility	EP0
partitioned	false
segmented	true
answers	DA50

ANSWER properties	
transaction type	DA50
struct name	answer_non_trad_settl_days
segmented	true

3.1.52.2 Related Messages

BU50

3.1.52.3 Purpose

The purpose of this query is to retrieve Non-settlement days for all Markets and Instrument Classes. Any settlement days defined on Instrument Class level overrides the days specified on Market level for that specific Instrument Class.

3.1.52.4 Structure

The DQ50 QUERY has the following structure:

```
struct query_non_trad_settl_days {
```

```
   struct transaction_type
   struct series  // Named struct no: 50000
   UINT16_T segment_number_n  // Segment Number
   char[2] filler_2_s  // Filler
}
```

3.1.52.5 Answer Structure

The DA50 ANSWER has the following structure:

```
struct answer_non_trad_settl_days {
   struct transaction_type
   UINT16_T segment_number_n  // Segment Number
   UINT16_T items_n  // Items
   Array ITEM [max no: 100] {
      struct series  // Named struct no: 50000
      char[8] date_non_trading_s  // Date, Non Trading
   }
}
```

3.1.52.6 Answer, comments

The answer received contains a list of non-settlement days for all markets and their connected instrument classes.

Series

- is specified with Country Number + Market Code - if specified on Market level.
- is specified with Country Number + Market Code + Instrument Group + Commodity Code - if specified on Instrument Class level.

3.1.53 DQ87 [Market Maker Protection QUERY]

3.1.53.1 Fingerprint

QUERY properties	
transaction type	DQ87
calling sequence	omniapi_query_ex
struct name	query_mm_protection
facility	EP0
partitioned	false
segmented	true
answers	DA87

ANSWER properties	
transaction type	DA87
struct name	answer_mm_protection
segmented	true

3.1.53.2 Related Messages

BU87, DC87

3.1.53.3 Purpose

The Query Market Maker Protection provides information of the market maker protection parameters defined for the participant and underlying.

3.1.53.4 Structure

The DQ87 QUERY has the following structure:

```
struct query_mm_protection {
   struct transaction_type
   struct series  // Named struct no: 50000
   UINT16_T segment_number_n  // Segment Number
   char[2] filler_2_s  // Filler
}
```

3.1.53.5 Usage and conditions

Series

Should be filled with 0 (zero)

3.1.53.6 Answer Structure

The DA87 ANSWER has the following structure:

```
struct answer_mm_protection {
   struct transaction_type
   UINT16_T segment_number_n  // Segment Number
   UINT16_T items_n  // Items
   Array ITEM [max no: 500] {
      INT64_T quantity_protection_q  // Quantity protection
      INT64_T delta_protection_q  // Delta protection
      INT32_T exposure_time_interval_i  // Exposure Time Interval
      INT32_T frozen_time_i  // Frozen Time
      UINT16_T commodity_n  // Commodity Code
      char[2] country_id_s  // Name, Country
      char[5] ex_customer_s  // Customer, Identity
      UINT8_T include_futures_c  // Include futures
      char[2] filler_2_s  // Filler
```

```
    }
}
```

3.1.54 DQ120 [Delta Underlying QUERY]

3.1.54.1 Fingerprint

QUERY properties	
transaction type	DQ120
calling sequence	omniapi_query_ex
struct name	query_delta
facility	EP0
partitioned	false
segmented	true
answers	DA120

VIA properties	
transaction type	DA120
struct name	The message complies with the VIM concept and has no top struct. The sequence of possible structs is described in the Structure section.
segmented	true

3.1.54.2 Related Messages

BU120

3.1.54.3 Purpose

The Delta Underlying Query is used to retrieve information about a new underlying or an underlying that has been changed.

3.1.54.4 Concept of Delta Queries and Broadcasts

The first time the user sends the delta query a full answer is needed, since the user does not have any stored instrument data. To receive a full answer, the Download Reference Number in the query is sent with NO_VALUE (equals to any negative integer, for example -1). The answer contains the latest Download Reference Number for the query.

The next time the user logs in, the previous delta sequence number is incremented by one and sent with the query (if only the delta is requested).

Each record in the answer is indicated with an operation that guides the client to Insert, Update, or Remove the item. A removal item for expired Option Instrument Series may contain a wildcard in Strike Price. The client application should remove all series that maps to the Instrument Class and Expiration Date.

Note: The operation is according to the back-end view of the data. Consequently, the client application should handle the following:

1. An Insertion can be received for an existing item. This should be treated as an Update.
2. An Update can be received for a non-existing item. This should be treated as an Insert.
3. A Removal can be received for a non-existing item. This should be ignored.

When sending the query, the client can choose to either query for a full answer or to receive only the delta since last login.

During certain circumstances, the back-end may enforce a full answer even though a delta was requested. This must be handled by the client.

In a full answer the operation will always be sent as Insert.

When querying for instrument data, only instruments defined in the allowed list for the user/participant are returned in the answer. If this setup of allowed instruments is changed, either by removing or adding new instruments, the central system cannot detect this easily from the sequence number.

Therefore when a delta query is received, the system checks if the setup has been changed since the last time the user logged in (this is detected from the Download Reference Number sent in the query). If that is the case, a full answer is returned together with a field in the answer header that indicates that a full answer is received.

The full answer is required to be returned to the user only the first time the user sends the query after a change of the instrument access. Therefore the full answer time-stamp in the query is compared to the actual time-stamp of latest change of allowed instruments. If the full answer time-stamp is after the latest change, a full answer is not distributed again.

Example

Assume the highest Download reference number both in the central system and the api client, is 10.

1. Legal Instrument is changed in the central system with implementation time = T1.
2. The front-end api client sends a delta query with Download Reference Number 11 (=10+1) and a time-stamp (T0) of latest received full answer.
3. The central system compares the time-stamp T0 with implementation time T1. Apparently, the legal instruments are changed since latest full answer (T1 > T0), and a full answer is returned with Download reference number =10 and a new Full answer Time-stamp (T2, with current UTC time).
4. The next day the user logs in again using Download Reference Number 11, but this time with the new time-stamp, T2.
5. Assume the central system has now on its side the highest Download Reference Number =13 since some records have changed (but assuming no changes in legal instrument, that is T1 is still the latest implementation time).
6. The central system compares the time-stamp T2 with implementation time T1. Since the time-stamp T2 is after the latest change in legal instrument, the delta answer returns the delta with Download Reference Number =13 and the previous time-stamp (T2).

3.1.54.5 Structure

The DQ120 QUERY has the following structure:

```
struct query_delta
```

3.1.54.6 Usage and Conditions

Full Answer Timestamp

The timestamp is mandatory in the query. If it is missing or does not have a valid format, a full answer is distributed.

Download Reference Number

is used for synchronisation of the information sent from the central system. The api client must keep track of the highest number for which delta information is received. This number is distributed both in answers to explicitly put delta queries, as well as distributed in delta broadcasts. When putting a delta query this number is incremented by one and included in the query.

3.1.54.7 Answer Structure

The DA120 VIA has the following structure:

```
struct answer_segment_hdr
struct item_hdr
struct sub_item_hdr
struct ns_delta_header  // Named struct no: 37001
Sequence {
   struct item_hdr
   Sequence {
      struct sub_item_hdr
      Choice {
         struct ns_remove  // Named struct no: 37002
         struct ns_underlying_basic  // Named struct no: 37201
         struct ns_fixed_income  // Named struct no: 37202
         struct ns_coupon_dates  // Named struct no: 37203
      }
   }
}
```

3.1.54.8 Answer, comments

Query DQ120 will return all underlyings regardless of Status (active or suspended).

This query and the related queries listed in “Related Messages” above support a delta concept where the client application keeps track of the latest received item (Download Reference Number) and uses this number incremented with one the next time the query is sent. This means that the answer of the next query only will contain any changes that have occurred since the previous query.

Full Answer Timestamp

will contain the time (UTC) when a full answer was sent the last time. Consequently, if the current answer is a full answer, this time is update as compared to the time sent in the query.

Download Reference Number

is used for synchronisation of the information sent from the central system. The api client must keep track of the highest number for which delta information is received. This number is distributed both in answers to delta queries, as well as in delta broadcasts.

The NS_DELTA_HEADER structure will be the first item of the variable items.

3.1.55 DQ121 [Delta Underlying for Back Office QUERY]

3.1.55.1 Fingerprint

QUERY properties	
transaction type	DQ121
calling sequence	omniapi_query_ex
struct name	query_delta
facility	EP0
partitioned	false
segmented	true
answers	DA121

VIA properties	
transaction type	DA121
struct name	The message complies with the VIM concept and has no top struct. The sequence of possible structs is described in the Structure section.
segmented	true

3.1.55.2 Related Messages

BU121

3.1.55.3 Purpose

The Delta Underlying for Back Office query is used to retrieve information about a new Delta Underlying or a Delta Underlying that has been changed.

3.1.55.4 Structure

The DQ121 QUERY has the following structure:

```
struct query_delta
```

3.1.55.5 Usage and Conditions

The Delta Underlying for Back Office query DQ121 returns all instrument classes regardless of Traded (Yes or No).

For a detailed description of how to use this query and a general information on the content of broadcasts and answers to queries, please see section **DQ120**.

3.1.55.6 Answer Structure

The DA121 VIA has the following structure:

```
struct answer_segment_hdr
struct item_hdr
struct sub_item_hdr
struct ns_delta_header  // Named struct no: 37001
Sequence {
   struct item_hdr
   Sequence {
      struct sub_item_hdr
      Choice {
         struct ns_remove  // Named struct no: 37002
         struct ns_underlying_basic  // Named struct no: 37201
         struct ns_fixed_income  // Named struct no: 37202
         struct ns_coupon_dates  // Named struct no: 37203
      }
   }
}
```

3.1.55.7 Answer, comments

The NS_DELTA_HEADER structure will be the first item of the variable items.

3.1.56 DQ122 [Delta Instrument Class QUERY]

3.1.56.1 Fingerprint

QUERY properties	
transaction type	DQ122
calling sequence	omniapi_query_ex
struct name	query_delta
facility	EP0
partitioned	false
segmented	true
answers	DA122

VIA properties	
transaction type	DA122
struct name	The message complies with the VIM concept and has no top struct. The sequence of possible structs is described in the Structure section.
segmented	true

3.1.56.2 Related Messages

BU122

3.1.56.3 Purpose

Instrument class query is used to retrieve information about a new Instrument Class or an Instrument Class that has been changed.

3.1.56.4 Structure

The DQ122 QUERY has the following structure:

```
struct query_delta
```

3.1.56.5 Usage and Conditions

Instrument class query DQ122 returns all instrument classes regardless of Traded (Yes or No) when a delta is returned. In the case of a full answer only classes denoted as Traded=yes are returned.

For a detailed description of how to use this query and a general information on the content of broadcasts and answers to queries, refer to section **DQ120**.

3.1.56.6 Answer Structure

The DA122 VIA has the following structure:

```
struct answer_segment_hdr
struct item_hdr
struct sub_item_hdr
struct ns_delta_header  // Named struct no: 37001
Sequence {
   struct item_hdr
   Sequence {
      struct sub_item_hdr
      Choice {
         struct ns_remove  // Named struct no: 37002
         struct ns_inst_class_basic  // Named struct no: 37101
         struct ns_price_tick  // Named struct no: 37102
         struct ns_block_size  // Named struct no: 37103
         struct ns_inst_class_secur  // Named struct no: 37105
      }
   }
}
```

3.1.56.7 Answer, comments

When there are multiple tick sizes for a class, the named structure no: 37102 (**NS Price Tick**) is repeated.

For **NS Price Tick**, the instrument is traded in price or yield. **NS Price Tick Corr** gives the corresponding price if the trade is in yield, or the corresponding yield if the trade is in price.

The NS_DELTA_HEADER structure will be the first item of the variable items.

3.1.57 DQ123 [Delta Instrument Class for Back Office QUERY]

3.1.57.1 Fingerprint

QUERY properties	
transaction type	DQ123
calling sequence	omniapi_query_ex
struct name	query_delta
facility	EP0
partitioned	false
segmented	true
answers	DA123

VIA properties	
transaction type	DA123
struct name	The message complies with the VIM concept and has no top struct. The sequence of possible structs is described in the Structure section.
segmented	true

3.1.57.2 Related Messages

BU123

3.1.57.3 Purpose

Instrument class query is used to retrieve information about a new Instrument Class or an Instrument Class that has been changed.

3.1.57.4 Structure

The DQ123 QUERY has the following structure:

```
struct query_delta
```

3.1.57.5 Usage and Conditions

Instrument class query DQ123 (Back Office variant) returns all instrument classes regardless of Traded (Yes or No).

For a detailed description of how to use this query and a general information on the content of broadcasts and answers to queries, refer to section **DQ120**.

3.1.57.6 Answer Structure

The DA123 VIA has the following structure:

```
struct answer_segment_hdr
struct item_hdr
struct sub_item_hdr
struct ns_delta_header  // Named struct no: 37001
Sequence {
   struct item_hdr
   Sequence {
      struct sub_item_hdr
      Choice {
         struct ns_remove  // Named struct no: 37002
         struct ns_inst_class_basic  // Named struct no: 37101
         struct ns_price_tick  // Named struct no: 37102
         struct ns_block_size  // Named struct no: 37103
         struct ns_inst_class_secur  // Named struct no: 37105
      }
   }
}
```

3.1.57.7 Answer, comments

For **NS Price Tick**, the instrument is traded in price or yield. **NS Price Tick Corr** gives the corresponding price if the trade is in yield, or the corresponding yield if the trade is in price.

The NS_DELTA_HEADER structure will be the first item of the variable items.

3.1.58 DQ124 [Delta Instrument Series QUERY]

3.1.58.1 Fingerprint

QUERY properties	
transaction type	DQ124
calling sequence	omniapi_query_ex
struct name	query_delta
facility	EP0
partitioned	false
segmented	true
answers	DA124

VIA properties	
transaction type	DA124
struct name	The message complies with the VIM concept and has no top struct. The sequence of possible structs is described in the Structure section.
segmented	true

3.1.58.2 Related Messages

BU124

3.1.58.3 Purpose

Instrument series query is used to retrieve information about a new Instrument Series or an Instrument Series that has been changed.

3.1.58.4 Structure

The DQ124 QUERY has the following structure:

```
struct query_delta
```

3.1.58.5 Usage and Conditions

Instrument series query DQ124 returns all instrument series regardless of Last Trade Date, Traded (Yes or No), and Status (Active or Suspended) when a delta is returned. In the case of a full answer only series denoted as Traded=yes and with Last Trading Date in the future are returned.

For a detailed description of how to use this query and a general information on the content of broadcasts and answers to queries, refer to section **DQ120**.

When querying for Instrument Series and the operation is Remove, the binary representation may contain wildcard. For single series the only possible field that may contain wildcard in the series binary code is Strike Price.

3.1.58.6 Answer Structure

The DA124 VIA has the following structure:

```
struct answer_segment_hdr
struct item_hdr
struct sub_item_hdr
struct ns_delta_header  // Named struct no: 37001
Sequence {
   struct item_hdr
   Sequence {
      struct sub_item_hdr
      Choice {
         struct ns_remove  // Named struct no: 37002
         struct ns_inst_series_basic  // Named struct no: 37301
         struct ns_inst_series_basic_single  // Named struct no: 37302
```

```
            struct ns_inst_series_id  // Named struct no: 37310
        }
    }
}
```

3.1.58.7 Answer, comments

The NS_DELTA_HEADER structure will be the first item of the variable items.

3.1.59 DQ125 [Delta Instrument Series for Back Office QUERY]

3.1.59.1 Fingerprint

QUERY properties	
transaction type	DQ125
calling sequence	omniapi_query_ex
struct name	query_delta
facility	EP0
partitioned	false
segmented	true
answers	DA125

VIA properties	
transaction type	DA125
struct name	The message complies with the VIM concept and has no top struct. The sequence of possible structs is described in the Structure section.
segmented	true

3.1.59.2 Related Messages

BU125

3.1.59.3 Purpose

Instrument series query is used to retrieve information about a new Instrument Series or an Instrument Series that has been changed.

3.1.59.4 Structure

The DQ125 QUERY has the following structure:

```
struct query_delta
```

3.1.59.5 Usage and Conditions

Instrument series query DQ125 (Back Office variant) will return all series regardless of Last Trade Date, Traded (Yes or No), and Status (Active or Suspended).

For a detailed description of how to use this query and a general information on the content of broadcasts and answers to queries, refer to section **DQ120**.

When querying for Instrument Series and the operation is Remove, the binary representation may contain wildcard. For single series the only possible field that may contain wildcard in the series binary code is Strike Price.

3.1.59.6 Answer Structure

The DA125 VIA has the following structure:

```
struct answer_segment_hdr
struct item_hdr
struct sub_item_hdr
struct ns_delta_header  // Named struct no: 37001
Sequence {
   struct item_hdr
   Sequence {
      struct sub_item_hdr
      Choice {
         struct ns_remove  // Named struct no: 37002
         struct ns_inst_series_basic  // Named struct no: 37301
         struct ns_inst_series_basic_single  // Named struct no: 37302
         struct ns_inst_series_bo  // Named struct no: 37306
         struct ns_inst_series_id  // Named struct no: 37310
      }
   }
}
```

3.1.59.7 Answer, comments

The NS_DELTA_HEADER structure will be the first item of the variable items.

3.1.60 DQ126 [Combo Series QUERY]

3.1.60.1 Fingerprint

QUERY properties	
transaction type	DQ126
calling sequence	omniapi_query_ex
struct name	query_combo
facility	EP0
partitioned	false
segmented	true

QUERY properties	
answers	DA126

VIA properties	
transaction type	DA126
struct name	The message complies with the VIM concept and has no top struct. The sequence of possible structs is described in the Structure section.
segmented	true

3.1.60.2 Related Messages

Related queries: DQ120, DQ122, DQ124 (and DQ121, DQ123, DQ125 which are Back Office related)

Related broadcasts: BU120, BU122, BU124, BU126 (and BU121, BU123, BU125 which are Back Office related)

3.1.60.3 Purpose

This query is used to retrieve information about a new Combination Series or a Combination Series that has been changed.

3.1.60.4 Structure

The DQ126 QUERY has the following structure:

```
struct query_combo {
   struct transaction_type
   struct series  // Named struct no: 50000
   UINT16_T segment_number_n  // Segment Number
   char[2] filler_2_s  // Filler
}
```

3.1.60.5 Usage and Conditions

Note that this query and the related BU126 do not support the delta concept that the querys and broadcasts listed in "Related Messages" above support.

Series

The Series may be zeroed (all markets) or completed as **Country Number** and **Market Code** or a complete **Instrument Type**.

3.1.60.6 Answer Structure

The DA126 VIA has the following structure:

```
   struct answer_segment_hdr
```

```
Sequence {
   struct item_hdr
   Sequence {
      struct sub_item_hdr
      Choice {
         struct ns_inst_series_basic  // Named struct no: 37301
         struct ns_combo_series_leg  // Named struct no: 37308
         struct ns_inst_series_id  // Named struct no: 37310
      }
   }
}
```

3.1.61 DQ136 [Combo Series for Back Office QUERY]

3.1.61.1 Fingerprint

QUERY properties	
transaction type	DQ136
calling sequence	omniapi_query_ex
struct name	query_combo
facility	EP0
partitioned	false
segmented	true
answers	DA136

VIA properties	
transaction type	DA136
struct name	The message complies with the VIM concept and has no top struct. The sequence of possible structs is described in the Structure section.
segmented	true

3.1.61.2 Related Messages

BU136

3.1.61.3 Purpose

This query is used to retrieve information about all existing Combination Series, also historical Combination Series where the last trading date has passed. In the message DQ126 only tradable Combination Series are returned.

3.1.61.4 Structure

The DQ136 QUERY has the following structure:

```
struct query_combo {
   struct transaction_type
   struct series  // Named struct no: 50000
   UINT16_T segment_number_n  // Segment Number
   char[2] filler_2_s  // Filler
}
```

3.1.61.5 Usage and Conditions

> **Note:**
> This query and the related BU136 do not support the delta concept.

Series

may be zeroed (all markets) or completed as **Country Number** and **Market Code** or a complete **Instrument Type**.

3.1.61.6 Answer Structure

The DA136 VIA has the following structure:

```
struct answer_segment_hdr
Sequence {
   struct item_hdr
   Sequence {
      struct sub_item_hdr
      Choice {
         struct ns_inst_series_basic  // Named struct no: 37301
         struct ns_combo_series_leg  // Named struct no: 37308
         struct ns_inst_series_id  // Named struct no: 37310
      }
   }
}
```

3.2 Trading and Market Information

3.2.1 BD2 [Edited Price Information VIB]

3.2.1.1 Fingerprint

VIB properties	
transaction type	BD2
calling sequence	omniapi_read_event_ext_ex or omniapi_read_event_block

VIB properties	
struct name	The message complies with the VIM concept and has no top struct. The sequence of possible structs is described in the Structure section.
info type	instrument class
virtual underlying	true

3.2.1.2 Purpose

The subscription to BD2 provides processed price information from the Central System. The data populated is based on trades executed during the trading day and could be subject to a holdback before distributed.

Note: Some products could be marked by the Exchange to have restricted information dissemination. Broadcasts will not be sent out for such products.

3.2.1.3 Structure

The BD2 VIB has the following structure:

```
struct broadcast_hdr
Sequence {
   struct item_hdr
   Sequence {
      struct sub_item_hdr
      Choice {
         struct market_info_series  // Named struct no: 33038
         struct market_info_reason  // Named struct no: 33043
         struct market_info_base  // Named struct no: 33034
         struct market_info_hke  // Named struct no: 33044
         struct ob_levels_closing  // Named struct no: 33031
      }
   }
}
```

3.2.1.4 Usage and Conditions

In order to maintain a real time database of the BD2 information the client application must use the IQ18/IQ19 queries to download a baseline of the information. Please refer to each section respectively for information on the sequence for this.

3.2.1.5 Structure Contents

The set of possible named structures cannot be changed intra day.

For some structured data additional explanations are provided in the following.

Market Info, Series

Fields usage in this structure:

All or None indicates if the given information relates to the ‘All or None’ deal history. Deals from the ‘All or None’ order book are calculated separately from other deals for the instrument. It could thus exist one set of high, low, last etc. that relates to the ‘All or None’ executed orders and one set that relates to ordinary orders executed. It should be noted that trading with ‘All or None’ orders are not available to all exchanges.

Market Info, Base

This structure is provided in the broadcast only if any of the included fields has a new value set.

Fields usage in this structure:

Price, Opening
Price, High
Price, Low
Price, Last When any price fields has bit 31 set (highest bit) while all other bits are zero, this indicates that no price is available. This differs from the value of zero (all bits zero) indicating a price of zero (allowed at some exchanges).

Turnover means the number of traded contracts during the day. If there are 100 contracts in a deal (100 bids and 100 asks) the turnover will increase with 100.

Number of deals gives the number of deals executed today.

Deal source contains the deal source of the last executed deal for the instrument.

Market Info, HKE

This structure is provided only if any of the included fields has a new value set and its distribution has been enabled by the exchange.

Order Book Levels, Closing

This structure is provided in the broadcast only if any of the included fields has a new value set.

Fields usage in this structure:

Price, Closing When the price field has bit 31 set (highest bit) while all other bits are zero, this indicates that no price is available. This differs from the value of zero (all bits zero) indicating a price of zero.

3.2.2 BD3 [Underlying Information BROADCAST]

3.2.2.1 Fingerprint

BROADCAST properties	
transaction type	BD3
calling sequence	omniapi_read_event_ext_ex or omniapi_read_event_block
struct name	underlying_info
info type	general

3.2.2.2 Purpose

This subscription returns information on the Underlying products. This information is normally produced outside the Exchange and distributed in the API.

3.2.2.3 Structure

The BD3 BROADCAST has the following structure:

```
struct underlying_info {
   struct broadcast_type
   INT32_T bid_premium_i  // Bid Premium
   INT32_T ask_premium_i  // Ask Premium
   INT32_T closing_price_i  // Price, Closing
   INT32_T opening_price_i  // Price, First
   INT32_T high_price_i  // Price, High
   INT32_T low_price_i  // Price, Low
   INT32_T last_price_i  // Price, Last
   INT32_T ref_price_i  // Price, Reference
   INT64_T turnover_u  // Turnover
   INT64_T best_bid_volume_u  // Best Bid Volume
   INT64_T best_ask_volume_u  // Best Ask Volume
   UINT8_T undisclosed_bid_volume_c  // Undisclosed Bid Volume
   UINT8_T undisclosed_ask_volume_c  // Undisclosed Ask Volume
   char[2] filler_2_s  // Filler
   UINT16_T commodity_n  // Commodity Code
   char[6] ext_time_s  // Time, External
}
```

3.2.2.4 Usage and conditions

Price, reference

is exchange specific where the exchange itself specifies the usage of it. The field is thus not always updated.

> **Note:** The data contained in this broadcast is normally produced outside the exchange.

3.2.3 BD70 [Trade Ticker VIB]

3.2.3.1 Fingerprint

VIB properties	
transaction type	BD70
calling sequence	omniapi_read_event_ext_ex or omniapi_read_event_block
struct name	The message complies with the VIM concept and has no top struct. The sequence of possible structs is described in the Structure section.
info type	instrument class

VIB properties	
segmented	true

3.2.3.2 Related Messages

TR70, BD71, TR71

3.2.3.3 Purpose

This broadcast is used to subscribe for deals executed in the market.

3.2.3.4 Structure

The BD70 VIB has the following structure:

```
struct broadcast_hdr
Sequence {
   struct sub_item_hdr
   Choice {
      struct basic_trade_ticker  // Named struct no: 34401
      struct extended_trade_ticker  // Named struct no: 34402
      struct trade_report_trade_ticker  // Named struct no: 34403
      struct half_trade_ticker  // Named struct no: 34405
   }
}
```

3.2.3.5 Usage and conditions

Segment Number

If segment number is non-zero it indicates that the total deal is split between several broadcasts. The last broadcast for one deal will have segment number equal to 0.

In the struct basic_trade_ticker, the **Match group number** field should not be used.

3.2.4 BD71 [Amended Trades VIB]

3.2.4.1 Fingerprint

VIB properties	
transaction type	BD71
calling sequence	omniapi_read_event_ext_ex or omniapi_read_event_block
struct name	The message complies with the VIM concept and has no top struct. The sequence of possible structs is described in the Structure section.
info type	instrument class
segmented	true

3.2.4.2 Related Messages

TR70, BD70, TR71

3.2.4.3 Purpose

This broadcast is used to subscribe for amended and canceled deals.

3.2.4.4 Structure

The BD71 VIB has the following structure:

```
struct broadcast_hdr
Sequence {
   struct sub_item_hdr
   Choice {
      struct trade_ticker_amend  // Named struct no: 34406
      struct basic_trade_ticker  // Named struct no: 34401
      struct half_trade_ticker  // Named struct no: 34405
   }
}
```

3.2.4.5 Usage and conditions

BD71 can be linked to the original Trade in BD70 using Match Group Number and Series.

In the struct basic_trade_ticker, the **Match group number** field should not be used.

3.2.5 BI1 [Resumption and Suspension of Trading BROADCAST]

3.2.5.1 Fingerprint

BROADCAST properties	
transaction type	BI1
calling sequence	omniapi_read_event_ext_ex or omniapi_read_event_block
struct name	suspend_resume_trading
info type	general

3.2.5.2 Purpose

This subscription returns information related to suspended trading for a certain commodity as well as information when trading will start.

3.2.5.3 Structure

The BI1 BROADCAST has the following structure:

```
struct suspend_resume_trading {
   struct broadcast_type
   UINT16_T commodity_n  // Commodity Code
   UINT8_T on_off_c  // On or Off
   CHAR filler_1_s  // Filler
}
```

3.2.6 BI5 [Indices Information BROADCAST]

3.2.6.1 Fingerprint

BROADCAST properties	
transaction type	BI5
calling sequence	omniapi_read_event_ext_ex or omniapi_read_event_block
struct name	index_info
info type	general

3.2.6.2 Purpose

This subscription returns information on the indices products. This information is normally produced outside the Exchange and is distributed in the API.

3.2.6.3 Structure

The BI5 BROADCAST has the following structure:

```
struct index_info {
   struct broadcast_type
   char[15] index_s  // Index, Identify
   char[8] last_index_s  // Index, Last Value
   char[8] high_index_s  // Index, Highest Value
   char[8] low_index_s  // Index, Lowest Value
   char[8] change_previous_s  // Change, Since Previous
   char[8] change_yesterday_s  // Change, Since Yesterday
   char[5] timestamp_dist_s  // Time, Distribution
   char[5] timestamp_comp_s  // Time, Computation
   char[3] filler_3_s  // Filler
}
```

3.2.6.4 Usage and conditions

Change, Since Previous
Change, Since Yesterday

are expressed as a change in percentage where current values are compared to the previous value or the last value from yesterday (or previous trading day). –10.35 means that the index has decreased 10,35 % since last day. 0.15 means that current index value is 0,15 % higher than the previous value.

Time, Distribution
Time, Computation

is given by the information supplier.

> **Note:** The data contained in this broadcast is normally produced outside the exchange.

3.2.7 BI7 [Signal Information Ready BROADCAST]

3.2.7.1 Fingerprint

BROADCAST properties	
transaction type	BI7
calling sequence	omniapi_read_event_ext_ex or omniapi_read_event_block
struct name	info_ready
info type	general

3.2.7.2 Purpose

This broadcast is used throughout the system to notify processes and applications that certain information is at hand, or that specific events have occurred. The nature of the message lies within the broadcast's information type and is interpreted according to the list given in the documentation of the Information Type field.

3.2.7.3 Structure

The BI7 BROADCAST has the following structure:

```
struct info_ready {
   struct broadcast_type
   INT32_T info_type_i  // Information Type
   struct series  // Named struct no: 50000
   char[8] business_date_s  // Date, Business
   char[8] sent_date_s  // Date, Sent
   char[6] sent_time_s  // Time, Sent
   char[8] clearing_date_s  // Clearing Date
   UINT16_T seq_num_srm_n  // Sequence number for SRM
}
```

3.2.7.4 Usage and Conditions

Information Type

In general, only a subset of the information types is of relevance to a specific exchange. The following information types are considered relevant in the context of this manual. Note that the descriptions below are to be regarded as complementary text to the descriptions in the **Detailed Field Information** chapter. Note also that the **Detailed Field Information** chapter lists all information types.

Information type	Interpretation	Comment
1	Binary information ready	When the signal is sent, all binary clearing data is ready for retrieval (per instrument type). Series contains in this case Country Number, Market Code and Instrument Group.
2	All reports ready	Not used in Genium INET.
3	Product in repair state	The signal BI7 type 3 is sent in the evening if new data is to be produced for the current business date and a BI7 type 1 has already been sent. Other BI7 or BI26 type signals might also have been sent, e.g. BI7, type 2. After the BI7 type 3 signal has been sent, new trades via Dedicated Trade Information Broadcast and new deliveries via BD18 is sent followed by a BI7 type 1 signal and possibly other BI7 or BI28 signals. This is used in case of an emergency situation. Series contains in this case Country Number and Market Code.
8	Margin information ready	Series contains in this case Country Number and Market Code.
9	Margin vector information ready	Series contains in this case Country Number and Market Code.
10	Margin information from margin call ready	This could be done intra-day. Series contains in this case Country Number and Market Code.
11	Sum margin information ready	Series contains in this case only zeroes.
12	New series generated	Series contains in this case; Country Number and Market, or Country Number, Market and Instrument Group, or Country Number, Market, Instrument Group and Commodity.
13	All securities closed	
16	Exercise/delivery information	Series contains in this case; Instrument type. Only used in linked clearing.
17	Open interest ready	Series contains in this case; Instrument type. Only used in linked clearing.

Information type	Interpretation	Comment
19	Signal fixing ready	Only sent on redemption. Series contains in this case Country Number and Market Code.
41	Margin Evening Prices and preliminary vector files ready	-
42	Intra Day Margin Calculation ready	This information is sent out when the intra day calculation has totally finished.
49	API data from Intra Day Margin Calculation ready	This information type is sent out when API data from intra day calculation is available, but reports still remain to be created.
50	Owl cycle ready	This information type is used instead of type 42 when dealing with owl cycle results.
51	API data from Owl cycle ready	This information type is used instead of type 49 when dealing with owl cycle results.
100	Daily trading statistics ready	This information type is use to declare that the daily trade statistics is available for current business day. Series contains in this case Country Number and Market Code.
101	Revised Daily Trade statistics information	This information type is use to declare that the daily trade statistics for a previous business day has been updated with a new revised open interest. Series contains in this case Country Number and Market Code.
256 and above	Report <no> ready	This information type is used to declare that a certain report is now available. The report can be retrieved using LQ1. A standard set of reports is described in the documentation of LQ1. Information Type identifies the report. Series contains in this case Country Number and Market Code. Signals sent to indicate when specific reports are available depend on Exchange policy.

Information type	Interpretation	Comment
		Note: All Instrument types within the market must be signalled before the query (LQ1) can be used.

3.2.8 BI9 [Price Information Heartbeat BROADCAST]

3.2.8.1 Fingerprint

BROADCAST properties	
transaction type	BI9
calling sequence	omniapi_read_event_ext_ex or omniapi_read_event_block
struct name	info_heartbeat
info type	general

3.2.8.2 Purpose

Price information heartbeat is a means for trader applications to detect if the price information flow is alive. It is implemented as a broadcast sent out regularly (for example every 20 seconds) from the Central System.

3.2.8.3 Structure

The BI9 BROADCAST has the following structure:

```
struct info_heartbeat {
   struct broadcast_type
   UINT8_T heartbeat_interval_c  // Heartbeat Interval
   UINT8_T instance_c  // Instance, Number
   UINT8_T tot_instances_c  // Total Instance
   char[40] description_s  // Description
   CHAR filler_1_s  // Filler
}
```

3.2.8.4 Usage and Conditions

Heartbeat Interval

gives the interval between two Price Information Heartbeat broadcasts. Within the interval, at least one broadcast will be sent by each Information Heartbeat sender.

Instance, Number

uniquely identifies an Information Heartbeat sender in the central system.

Total Instance

defines the total number of Information Heartbeat senders in the central system.

Description

is a textual description of this particular Information Heartbeat sender.

Example:

If Total Instance is 3, there are three Information Heartbeat senders in the central system. Each of these senders distributes the broadcast and the first uses Instance Number 1, the second uses Instance Number 2 etc.

3.2.9 BI41 [Instrument Status Information BROADCAST]

3.2.9.1 Fingerprint

BROADCAST properties	
transaction type	BI41
calling sequence	omniapi_read_event_ext_ex or omniapi_read_event_block
struct name	instrument_status_info
info type	general

3.2.9.2 Purpose

The Instrument Status Information broadcast consists of the status for a market, an instrument type, an instrument class, series or an underlying. It is sent at the actual change and as a warning before the state changes. The variable “State Change, Seconds” tells whether it is a warning or a state change. Value larger than zero means a warning.

3.2.9.3 Structure

The BI41 BROADCAST has the following structure:

```
struct instrument_status_info {
   struct broadcast_type
   UINT16_T items_n  // Items
   char[2] filler_2_s  // Filler
   Array ITEM [max no: 9] {
      struct series  // Named struct no: 50000
      UINT16_T seconds_to_state_change_n  // State Change, Seconds
      UINT16_T state_number_n  // Trading State Number
      char[80] warning_msg_s  // Warning Message
      UINT16_T state_level_e  // Level
      char[8] actual_start_date_s  // Actual Start Date
      char[6] actual_start_time_s  // Actual Start Time
      char[8] next_planned_start_date_s  // Planned Start Date, Next
```

```
        char[6] next_planned_start_time_s  // Planned Start Time, Next
        char[2] filler_2_s  // Filler
    }
}
```

3.2.9.4 Usage and Conditions

A **trading session state** is configurable on market level, instrument type level or instrument class level.

An **instrument session state** is configurable on instrument series level or underlying level.

The Query Instrument Status transaction is used as recovery for this broadcast, see UQ15 (Instrument Status Query).

Series

Series should be completed according to the table below to be able to identify a specific Market, Instrument Type, Instrument Class, Series or Underlying.

What to identify	Complete the following fields
Market	Country Number Market Code
Instrument Type	Country Number Market Code Instrument Group
Instrument Class	Country Number Market Code Instrument Group Commodity Code
Series	Country Number Market Code Instrument Group Commodity Code Expiration Date Price, Strike
Underlying	Commodity Code

Expiration Date
Strike Price

can in some cases be zero for a series.

Trading State Number

can have the value of zero, only for trading state changes on series and underlying level. The meaning of this is that the trading state is no longer set on series level, and the series level inherits the trading state from the level above.

Level

The Level field is supplied as a means to separate an instrument class from a series.

If, for example, the value 2 is sent in, only session states set on Instrument Type will be returned.

Seconds to State Change

may have a value other than zero, e.g. for trading state changes on series level or for warning messages.

3.2.10 BI63 [Preliminary Settlement Prices BROADCAST]

3.2.10.1 Fingerprint

BROADCAST properties	
transaction type	BI63
calling sequence	omniapi_read_event_ext_ex or omniapi_read_event_block
struct name	settle_price_update
info type	general

3.2.10.2 Purpose

This subscription returns intra day calculated settlement prices.

3.2.10.3 Structure

The BI63 BROADCAST has the following structure:

```
struct settle_price_update {
   struct broadcast_type
   UINT16_T items_n  // Items
   char[2] filler_2_s  // Filler
   Array ITEM [max no: 50] {
      struct series  // Named struct no: 50000
      INT32_T settle_price_i  // Price, Settlement
      char[8] settlement_date_s  // Date, Settlement
      UINT8_T settlement_price_type_c  // Settlement Price Type
      char[3] filler_3_s  // Filler
   }
}
```

3.2.10.4 Usage and conditions

The exchange might calculate settlement prices for all or a subset of all instrument series intra day. The calculation might be executed more than once for each instrument series. It is an exchange decision when,

how often and for which instrument series intra day settlement prices are calculated. It is furthermore an exchange decision how the intra day settlement prices relate to the settlement price published in the Trade Statistics Query.

To download current values for the preliminary settlement prices, the Preliminary Settlement Prices Query is used.

Settlement Price Type

is type of settlement price. It is an exchange decision which price types to use.

Price, Settlement

when the price field has bit 31 set (highest bit) while all other bits are zero, this indicates that no price is available. This differs from the value of zero (all bits zero) indicating a price of zero.

Note: This information might not be produced and published by the exchange. The exchange might also have rules for when, how often and for which instrument series the information is produced.

3.2.11 BI73 [Undo Signal Ready Info BROADCAST]

3.2.11.1 Fingerprint

BROADCAST properties	
transaction type	BI73
calling sequence	omniapi_read_event_ext_ex or omniapi_read_event_block
struct name	undo_info_ready
info type	general

3.2.11.2 Purpose

When the Undo Signal Ready is triggered for a certain information type, a broadcast called Undo Signal Ready Info (BI73) will be sent.

3.2.11.3 Structure

The BI73 BROADCAST has the following structure:

```
struct undo_info_ready {
   struct broadcast_type
   INT32_T info_type_i  // Information Type
   struct series  // Named struct no: 50000
   char[8] business_date_s  // Date, Business
   char[8] clearing_date_s  // Clearing Date
   char[8] sent_date_s  // Date, Sent
   char[6] sent_time_s  // Time, Sent
   UINT16_T seq_num_srm_n  // Sequence number for SRM
}
```

3.2.12 BI81 [Market Announcement Information VIB]

3.2.12.1 Fingerprint

VIB properties	
transaction type	BI81
calling sequence	omniapi_read_event_ext_ex or omniapi_read_event_block
struct name	The message complies with the VIM concept and has no top struct. The sequence of possible structs is described in the Structure section.
info type	general

3.2.12.2 Purpose

The Market Announcement Information broadcast sends information to all users. This information can be either a market message or a company announcement.

3.2.12.3 Structure

The BI81 VIB has the following structure:

```
struct broadcast_hdr
Sequence {
   struct sub_item_hdr
   Choice {
      struct message_core_info  // Named struct no: 35001
      struct message_information  // Named struct no: 35002
      struct destination_item  // Named struct no: 35003
      struct document_url  // Named struct no: 35004
   }
}
```

3.2.12.4 Usage and Condition

Market Control Message

A Market Control Message is sent when the Market Control staff wants to send a message. It is normally sent to a whole market, i.e. with Level set to Market (destination_level_c = 1) but it can sometimes be sent on Underlying or Series level.

This message will be sent when the message type is set to Market Message/Market Control (message_information_type_c = 2). It can be sent with three different priorities: normal, high and low.

A Market Control Message can be sent with destination to all markets. This is indicated by series in destination_item is set to null (no specific market is indicated), and destination_level_c = 1 (Market level).

Company Announcement

A Company Announcement is sent when individual companies want to send information to the market, this means that these messages are typically sent

with Level set to Underlying (destination_level_c = 2) or Series level (destination_level_c = 3).

This message will be sent when the message type is set to Company Announcement (message_information_type_c = 1). It can be sent with three different priorities: normal, high and low.

3.2.12.5 Structure Contents

message_core_info (35001)

Fields usage in this structure:

Sequence Number	A serial number defined by the central system. The number starts with 1 every day.
Date Time, External	Time stamps in UTC.

document_url (35004)

Fields usage in this structure:

Items	holds information about the actual length of the URL. Only the actual size of the message is sent in the broadcast. The maximum value is decided by the URL data type.
Link, URL	holds the URL link pointing to a document where e.g. a full announcement can be found.

3.2.13 II12 [Underlying and indices QUERY]

3.2.13.1 Fingerprint

QUERY properties	
transaction type	II12
calling sequence	omniapi_query_ex
struct name	query_underlying_indices
facility	EP0
partitioned	false
answers	IA12

ANSWER properties	
transaction type	IA12
struct name	answer_underlying_indices
segmented	true

3.2.13.2 Purpose

This query makes it possible to retrieve information about underlyings and indices. The information returned corresponds to the information in the broadcasts:

- Indices Information, BI5
- Underlying Information, BD3

3.2.13.3 Structure

The II12 QUERY has the following structure:

```
struct query_underlying_indices {
   struct transaction_type
   struct series  // Named struct no: 50000
   UINT16_T segment_number_n  // Segment Number
   char[8] date_s  // Date
   char[2] filler_2_s  // Filler
}
```

3.2.13.4 Usage and conditions

Which underlyings or indices that are updated during the day and possible to retrieve information about, are defined by the Exchange.

Series

should be zero filled with one exception, the **Commodity Code** field. If the value of the Commodity Code field is zero, all current values for all underlying/indices are returned in the answer. If the value of the Commodity Code field is non-zero, it should contain a valid code in the system.

Date

specifies for which day the values should be requested. A value of “00000000“ gives the latest values. If no value exists for the current day, the previous trading day’s value will be returned. If a specific day is requested, the latest values for that day will be returned. The Date is specified in the format YYYYMMDD. Information is only available for a limited number of historical dates, which is defined by the Exchange. Typically 10 days of information is available.

3.2.13.5 Return Codes

An II12 transaction may also be aborted by the Marketplace, in which case only the reason for the transaction being aborted is returned to the sender.

cstatus	txstat		rcvbuf
Successful	Normal		list of values - see Answer, structure
Transaction aborted	BADSEG	Segment number can not be zero in an input query.	-

3.2.13.6 Answer Structure

The IA12 ANSWER has the following structure:

```
struct answer_underlying_indices {
   struct transaction type
   UINT16_T items_n  // Items
   UINT16_T segment_number_n  // Segment Number
   Array ITEM [max no: 635] {
      struct series  // Named struct no: 50000
      INT32_T bid_premium_i  // Bid Premium
      INT32_T ask_premium_i  // Ask Premium
      INT32_T closing_price_i  // Price, Closing
      INT32_T opening_price_i  // Price, First
      INT32_T high_price_i  // Price, High
      INT32_T low_price_i  // Price, Low
      INT32_T last_price_i  // Price, Last
      INT32_T ref_price_i  // Price, Reference
      INT32_T change_previous_i  // Change, Since Previous
      INT32_T change_yesterday_i  // Change, Since Yesterday
      INT32_T points_of_movement_i  // Points, Movement
      INT64_T turnover_u  // Turnover
      INT64_T best_bid_volume_u  // Best Bid Volume
      INT64_T best_ask_volume_u  // Best Ask Volume
      char[8] date_s  // Date
      char[6] ext_time_s  // Time, External
      UINT8_T undisclosed_bid_volume_c  // Undisclosed Bid Volume
      UINT8_T undisclosed_ask_volume_c  // Undisclosed Ask Volume
      char[2] filler_2_s  // Filler
      char[2] reserved_2_s  // Reserved
   }
}
```

3.2.13.7 Answer, comments

Date

reflects the requested date as specified in the query.

> **Note:** This information might not be produced by the Exchange and the exact contents of this record is dependent on the incoming data.

3.2.14 II17 [Preliminary Settlement Prices QUERY]

3.2.14.1 Fingerprint

QUERY properties	
transaction type	II17
calling sequence	omniapi_query_ex

QUERY properties	
struct name	query_prel_settlement
facility	EP0
partitioned	false
answers	IA17

ANSWER properties	
transaction type	IA17
struct name	answer_prel_settlement
segmented	true

3.2.14.2 Purpose

This query makes it possible to retrieve information about preliminary settlement prices calculated by the exchange intra day.

The information returned corresponds to the information broadcasted in the Preliminary Settlement Price Update Broadcast, see **BI63**.

3.2.14.3 Structure

The II17 QUERY has the following structure:

```
struct query_prel_settlement {
   struct transaction type
   struct series  // Named struct no: 50000
   char[8] settlement date s  // Date, Settlement
   UINT16 T segment number n  // Segment Number
   UINT8 T settlement price type c  // Settlement Price Type
   CHAR filler 1 s  // Filler
}
```

3.2.14.4 Usage and conditions

The exchange might calculate settlement prices for all or a subset of all instrument series intra day. The calculation might be executed more than once for each instrument series. It is an exchange decision when, how often and for which instrument series intra day settlement prices are calculated. It is furthermore an exchange decision how the intra day settlement prices relates to the settlement price published in the Trade Statistics Query.

Series

is either zero filled or filled with **Country Code**, **Market Code** and **Commodity Code**.

If zero filled the query will return information on all instrument series where preliminary settlement prices has been calculated intra day. If Country Code, Market Code and Commodity Code is filled in the query will only return instrument series that matches the given combination of these fields.

Date, Settlement

should contain the date of interest.

Settlement Price Type

should contain the Price Type of interest.

3.2.14.5 Return Codes

cstatus	txstat	rcvbuf
Successful	Normal	list of values – see Answer, structure
Transaction aborted	BADSEG: Segment number can not be Zero in an input query.	-

After a successful II17 transaction, a list of preliminary settlement prices is returned to the sender.

An II17 transaction might also be aborted by the Market place, in which case only the reason for the transaction being aborted is returned to the sender.

3.2.14.6 Answer Structure

The IA17 ANSWER has the following structure:

```
struct answer_prel_settlement {
   struct transaction_type
   UINT16_T items_n  // Items
   UINT16_T segment_number_n  // Segment Number
   Array ITEM [max no: 1500] {
      struct series  // Named struct no: 50000
      INT32_T settl_price_i  // Settlement Price
      char[8] settlement_date_s  // Date, Settlement
      UINT8_T settlement_price_type_c  // Settlement Price Type
      char[6] hhmmss_s  // Time, External
      CHAR filler_1_s  // Filler
   }
}
```

3.2.14.7 Answer, comments

Price, Settlement

when the price field has bit 31 set (highest bit) while all other bits are zero, this indicates that no price is available. This differs from the value of zero (all bits zero) indicating a price of zero.

Note: This information may not be produced and published by the exchange. The exchange may also have rules for when, how often and for which instrument series the information is produced.

3.2.15 IQ12 [Total Equilibrium Prices QUERY]

3.2.15.1 Fingerprint

QUERY properties	
transaction type	IQ12
calling sequence	omniapi_query_ex
struct name	query_tot_equil_prices
facility	EP0
partitioned	true
answers	IB12

ANSWER properties	
transaction type	IB12
struct name	answer_tot_equil_prices
segmented	true

3.2.15.2 Purpose

This query is used to download the equilibrium price information from the central system.

3.2.15.3 Structure

The IQ12 QUERY has the following structure:

```
struct query_tot_equil_prices {
   struct transaction type
   struct series  // Named struct no: 50000
   struct filter_series {
      UINT8_T country_c  // Country Number
      UINT8_T market_c  // Market Code
      UINT8_T instrument_group_c  // Instrument Group
      UINT8_T modifier_c  // Modifier
      UINT16_T commodity_n  // Commodity Code
      UINT16_T expiration_date_n  // Date, Expiration
      INT32_T strike_price_i  // Strike Price
   }
   UINT16_T segment_number_n  // Segment Number
   char[2] filler_2_s  // Filler
```

```
}
```

3.2.15.4 Usage and conditions

Series

must be filled with any valid series.

Series, filter

When sending an IQ12 query, the application must fill in a filter series. This filter series is then used by the central system when building the answer. Only instruments matching the provided filter are replied back. With the filter, it is possible to request all instruments for a given market, a given instrument type, or a given instrument class. It is also possible to request all available instruments or a specific one. A binary value of zero, for any field in the filter, indicates a wildcard. Thus, to fill in an instrument type, filter the application should fill in country code, market code, and instrument type code. All other fields within the filter should be set to binary zero.

The usage of the IQ12 transaction is defined by the exchange.

3.2.15.5 Return Codes

An IQ12 transaction may also be aborted by the Marketplace. In that case only the reason for the transaction being aborted is returned to the sender.

cstatus	txstat	rcvbuf
Successful	Normal	list of equilibrium prices, see Answer, strucure
Transaction aborted	INFO_BADSEG	-
Transaction aborted	...	-

Please refer to the **Error Messages Reference Manual** for details about why transcations are aborted.

3.2.15.6 Answer Structure

The IB12 ANSWER has the following structure:

```
struct answer_tot_equil_prices {
   struct transaction_type
   UINT16_T segment_number_n  // Segment Number
   UINT8_T instance_c  // Instance, Number
   UINT8_T instance_next_c  // Next Instance Number
   struct series_next
   UINT16_T items_n  // Items
   char[2] filler_2_s  // Filler
   Array ITEM [max no: 1230] {
      struct series  // Named struct no: 50000
      INT64_T equilibrium_quantity_i  // Equilibrium Volume
      INT32_T equilibrium_price_i  // Price, Equilibrium
      INT32_T best_bid_premium_i  // Best Bid Price, Preopening
      INT32_T best_ask_premium_i  // Best Ask Price, Pre-opening
```

```
        INT64_T best_bid_quantity_i  // Best Bid Volume, Preopening
        INT64_T best_ask_quantity_i  // Best Ask Volume, Pre-opening
        UINT8_T matching_price_type_c  // Matching Price Type
        char[3] filler_3_s  // Filler
    }
}
```

3.2.15.7 Answer, comments

After a successfull IQ12 transaction, a list of equilibrium prices is returned to the sender.

Price fields

If any **Price** has bit 31 set (the highest bit) while all other bits are zero, this means that no price is available. Note the use of different bit patterns to distinguish a price that is not available from a price that is zero. For the value of zero, set all bits to zero.

Equilibrium Volume
Best Bid Volume, Pre- Opening
Best Ask Volume, Pre-Opening

These fields are only updated if enabled by the exchange.

Best Bid Price, Pre-Opening
Best Ask Price, Pre-Opening

These fields are only updated if enabled by the exchange.

The Client should confirm to the following logic in order to download data for all instrument series:

1. When the answer is recieved for the first query, the recieved Instance Number must be remembered.
2. From the answer structure, copy the Next Series to the subsequent query. If the Segment Number in the answer is greater than zero, the value should be incremented by one and copied to the Segment Number in the subsequent query, otherwise (received Segment Number is zero) the value of one should be copied.
3. Repeat step 2 until Next Instance Number in the answer is equal to the saved value from step 1 and the Segment Number in the answer is zero.

3.2.16 IQ18 [Total Volumes and Prices VIQ]

3.2.16.1 Fingerprint

VIQ properties	
transaction type	IQ18
calling sequence	omniapi_query_ex
struct name	The message complies with the VIM concept and has no top struct. The sequence of possible structs is described in the Structure section.
facility	EP0

VIQ properties	
partitioned	true
virtual underlying	true
answers	IA18

VIA properties	
transaction type	IA18
struct name	The message complies with the VIM concept and has no top struct. The sequence of possible structs is described in the Structure section.
segmented	true

3.2.16.2 Purpose

This query is used to download the Edited Price Information and Edited Order Book Information from the central system. In order to maintain a real time database of the information published in the transactions the user application must listen to BO14 and BD2 broadcasts.

3.2.16.3 Structure

The IQ18 VIQ has the following structure:

```
struct query_hdr
struct sub_item_hdr
struct ob_levels_query_data  // Named struct no: 33020
Sequence {
   struct sub_item_hdr
   Choice {
      struct ob_levels_id  // Named struct no: 33002
   }
}
```

3.2.16.4 Usage and Conditions

The logic is that the IQ18 provides the baseline of information for the BO14 and BD2 broadcasts.

IQ18 uses a technique that involves both segmented answers and instance numbers. The instance numbers represent answering processes in the central system. There might be one or several answering processes for IQ18. Applications using IQ18 must make sure that the transaction is sent to all answering processes in the central system. This implies that the application must send in a sequence of IQ18 transactions in order to download the data. The sequence is started by the application by specifying a randomly picked instrument series in the first IQ18 transaction. The answer to the first IQ18 transaction provides information on how to continue with the second IQ18 transaction. The second IQ18 answer provides information on how to continue with the third IQ18 transaction, and so on. How this is achieved is further described in the chapter “Structure, Contents” and chapter “Answer, Comments.”

The application may provide an optional filter in the transaction. If a filter is provided the central system only replies back instrument series matching the filter. If no filter is provided the central system replies back all instrument series traded this day. Regardless if a filter is provided or not the application must follow the

transaction rules as shortly described above and further described in chapter “Structure, Contents” and chapter “Answer, Comments.”

The following sequence of actions must be performed by the application in order to synchronize the query answer with BO14 and BD2 broadcasts.

1. Start subscribing for BO14 and BD2 broadcasts. Received broadcasts must not be processed until step 3. The user application must keep these broadcasts in an internal queue.
2. Send in the sequence of IQ18 queries (refer to “Answer, Comments” for more information).
3. When done with the IQ18, download of data the user application must handle the queued BO14 and BD2 broadcasts. They must be processed in the same order as they where received. The application has the correct information at the point when all queued broadcasts have been handled.
4. When all queued broadcasts have been processed the application can remove the usage of the internal queue. New broadcasts received should directly modify the (by the application) maintained database.

3.2.16.5 Structure Contents

query_hdr

Usage of fields in this structure:

Series	should in the first query be filled in with any valid series. In the consecutive queries, the series given in the previous answer shall be used. See Answer, Comments for more information.
Items	must be 1 if no filter is provided (Order Book Levels, Id), otherwise 2.
Size	must be the total transaction size in bytes.

Order Book Levels, Query Data

Usage of fields in this structure:

Segment Number	should in the first query be filled in with the value 1. In the consecutive queries, the Segment Number given in the previous answer shall be considered. See Answer, Comments for more information.

Order Book Levels, ID

This named structure is not mandatory in the query. If however provided, the series is used as a filter by the central system. Only instrument series matching the filter is returned in the answer.

> **Note:** There could only be zero or one occurrence of this structure in the query.

Usage of fields in this structure:

Series	is filled in with a valid filter series. The following filters are allowed: • Market (country and market code filled in. Other fields set to zero.) • Instrument type (country, market and group code filled in. Other fields set to zero.)

- Instrument class (country, market, group and commodity code filled in. Other fields set to zero.)
- Instrument series (a valid series is provided).

Block Size is not used and should be zero filled.

3.2.16.6 Return Codes

An IQ18 query may be aborted by the Marketplace. In this case only the reason for the query being aborted is returned to the sender.

cstatus	txstat	rcvbuf
Successful	Normal	list of price and order-book information – see above.
Transaction aborted	INFO_BADSEG	
Transacation aborted	...	

Please refer to *System Error Messages Reference* for details about why transcations are aborted.

3.2.16.7 Answer Structure

The IA18 VIA has the following structure:

```
struct answer_hdr
struct sub_item_hdr
struct ob_levels_next_query  // Named struct no: 33032
Sequence {
   struct sub_item_hdr
   struct ob_levels_id  // Named struct no: 33002
   Sequence {
      struct sub_item_hdr
      Choice {
         struct ob_levels_price_volumes  // Named struct no: 33003
         struct ob_levels_order_number  // Named struct no: 33004
         struct ob_levels_total_quantity  // Named struct no: 33005
         struct ob_levels_no_of_orders  // Named struct no: 33033
         struct ob_levels_price  // Named struct no: 33006
         struct market_info_base  // Named struct no: 33034
         struct market_info_hke  // Named struct no: 33044
         struct ob_levels_closing  // Named struct no: 33031
      }
   }
}
```

3.2.16.8 Answer, Comments

After a successful IQ18 transaction, a list of price and order-book information is returned to the sender.

The Client should confirm to the following logic in order to download the data:

1. When the answer is received for the first query, the received Instance Number must be remembered.
2. From the answer structure, copy the Series Next to the series in the query_hdr of the subsequent query. If the Segment Number in the answer is greater than zero the value should be incremented by one and copied to the Segment Number in the subsequent query, otherwise (received Sequence Number is zero) the value of one should be copied.
3. Repeat step 2 until Next Instance Number in the answer is equal to the saved value from step 1 and the Segment Number in the answer is zero.

The query answer will contain relevant information to the current market state. Information fields not applicable to the current market state will be excluded from the answer.

3.2.16.9 Answer, Structure Contents

Depending on exchange configuration, either **Order Book Levels, Price** or **Order Book Levels, Price and Volume** is distributed for a given instrument. The two of them are never distributed simultaneously for a given instrument. The exchange could however change the configuration intra day, causing a change of the distributed named structure. If for example the exchange decides to disable the volume distribution, the API client receives **Order Book Levels, Price and Volume** until this time and then directly a **Order Book Levels, Price**. The API client, in such a case, is responsible to clean up the local order book and remove volume figures as they are no longer being distributed by the exchange.

The interpretation of the various possible structures returned in the answer are the same as in BO14 and BD2 with some additions and exceptions described below.

Order Book Levels, Next Query

This structure is used by the application in order to perform a complete download of information as previously described.

Order Book Levels, ID

This structure defines the instrument series that succeeding structures relates to (up until the occurrence of a new Order Books Levels, ID structure).

For an example, please refer to BO14.

Order Book Levels, Price and Volumes

The Price masks are interpreted as bit fields where currently the low 10 bits are used. Bit 0 corresponds to the first ranked price, bit 1 to the second best ranked price, etc. For each bit set in the mask an array item is present. Bid items are placed before ask items in the array. Better rank prices are placed before lower ranked prices in the array. The field Items holds the total number of items within the array.

For examples, please refer to BO14.

Order Book Levels, Price

Each item in the array is of the structure type Order Book Levels, Price Item will be used the same way as Order Book Levels, Price and Volume when volume dissemination is not enabled. Then Order Book Levels, Price will be sent instead of Order Book Levels, Price and Volume.

Order Book Levels, Order Number

The order numbers provided in this structure are the order numbers for the first ranked bid and ask orders in the order book. Order number structures are only distributed if the exchange has enabled this distribution. If enabled, it applies to all instruments and is never changed intra day.

Order Book Levels, Total Quantity

are the total demand of all orders in the order book. Total quantity structures are only distributed if the exchange has enabled this distribution. If enabled, it applies to all instruments and is never changed intra day.

Order Book Levels, Closing

This structure is provided in the answer only if any of the included fields has a value set.

Usage of fields in this structure:

Price, Closing	The value of MIN_INT is used to indicate an undefined value while binary zero indicates a price of zero.

Order Book Levels, Number of Orders

The Number of Orders structure is only distributed if the exchange has enabled this distribution. If enabled, it applies to all instruments and is never changed intra day.

Usage of fields in this structure:

Number of orders	contains the number of orders on the price level that corresponds to this field's position in the array. For an example, please refer to BO14.

Market Info, Base

This structure is provided in the answer only if any of the included fields has a value set.

Usage of fields in this structure:

Price, Opening **Price, High** **Price, Low** **Price, Last**	The value of MIN_INT is used to indicate an undefined value while binary zero indicates a price of zero.

3.2.16.10 IQ18 Scenarios

The examples below illustrate the functionality of IA18 with respect to what information they may contain in different market situations.

Example

When the query is placed before the opening of the market - consequently no orders have been entered and no price or volume statistics are available - then the reply will consist only of the structures containing information, firstly the series the data relates to. Then for each series in the answer a possible closing price structure is sent. The reply also includes information about next query to send for more information.

In this case the answer will **not** contain any Order Book Levels, Price or Order Book Levels, Price and Volume structures, as the order book is empty. The answer will as well **not** include any Order Book Levels, Market Info structures as none of the included fields has a value set. The structure Order Book Levels, Closing will be included if the instrument series has a Closing price or Open balance set.

- Order Book Levels, Next Query
- Order Book Levels, ID
- Order Book Levels, Closing
- Order Book Levels, ID
- Order Book Levels, Closing
- Order Book Levels, ID (No Closing price or Open balance available)
- Order Book Levels, ID (No Closing price or Open balance available)
- Order Book Levels, ID
- Order Book Levels, Closing

Example

When the query is placed after the market has opened and there are orders in the market, and trades have been matched, then the sequence of named structures may look something like:

- Order Book Levels, Next Query
- Order Book Levels, ID
- Order Book Levels, Price and Volume (or Order Book Levels, Price)
- Order Book Levels, Market Info
- Order Book Levels, Closing
- Order Book Levels, Order Number (if enabled)
- Order Book Levels, Total Quantity (if enabled)
- Order Book Levels, Number of Orders
- Order Book Levels, ID
- Order Book Levels, Price and Volume (or Order Book Levels, Price)
- Order Book Levels, Market Info
- Order Book Levels, Closing
- Order Book Levels, Order Number (if enabled)
- Order Book Levels, Total Quantity(if enabled)
- Order Book Levels, Number of Orders
- Order Book Levels, ID

Example

When the query is placed after the market has opened and there are orders in the market but no trades have been matched, the sequence of named structures may look something like:

- Order Book Levels, Next Query
- Order Book Levels, ID
- Order Book Levels, Price and Volume (or Order Book Levels, Price)
- Order Book Levels, Closing (Closing price or Open balance set)
- Order Book Levels, Order Number (if enabled)
- Order Book Levels, Total Quantity (if enabled)

- Order Book Levels, Number of Orders
- Order Book Levels, ID
- Order Book Levels, Price and Volume (or Order Book Levels, Price)
- Order Book Levels, Closing (Closing price or Open balance set)
- Order Book Levels, Order Number (if enabled)
- Order Book Levels, Total Quantity (if enabled)
- Order Book Levels, Number of Orders
- Order Book Levels, ID

3.2.17 IQ19 [Total Volumes and Prices VIQ]

3.2.17.1 Fingerprint

VIQ properties	
transaction type	IQ19
calling sequence	omniapi_query_ex
struct name	The message complies with the VIM concept and has no top struct. The sequence of possible structs is described in the Structure section.
facility	EP0
partitioned	true
virtual underlying	true
answers	IA19

VIA properties	
transaction type	IA19
struct name	The message complies with the VIM concept and has no top struct. The sequence of possible structs is described in the Structure section.
segmented	true

3.2.17.2 Purpose

This transactions is used to download the Edited Price Information and Edited Order Book Information from the central system. In order to maintain a real time database of the information published in the transactions the user application must listen to BO15 and BD2 broadcasts.

3.2.17.3 Structure

The IQ19 VIQ has the following structure:

```
struct query_hdr
struct sub_item_hdr
struct ob_levels_query_data  // Named struct no: 33020
Sequence {
   struct sub_item_hdr
   Choice {
      struct ob_levels_id  // Named struct no: 33002
   }
}
```

3.2.17.4 Usage and conditions

The logic is that the IQ19 query provides the baseline of information for the BO15 and BD2 broadcasts.

IQ19 uses a technique involving both segmented answers and instance numbers. The instance numbers represent answering processes in the central system. There might be one or several answering processes for IQ19. Applications using IQ19 must make sure that the transaction is sent to all answering processes in the central system. This implies that the application must send in a sequence of IQ19 transactions in order to download the data. The sequence is started by the application by specifying a randomly picked instrument series in the first IQ19 transaction. The answer to the first IQ19 transaction provides information on how to continue with the second IQ19 transaction. The second IQ19 answer provides information on how to continue with the third IQ19 transaction etc. How this is achieved is further described in the chapter "Structure, Contents" and chapter "Answer, Comments".

The application may provide an optional filter in the transaction. If a filter is provided the central system only replies back instrument series matching the filter. If no filter is provided the central system replies back all instrument series traded this day. Regardless if a filter is provided or not the application must follow the transaction rules as shortly described above and further described in chapter Structure Contents and chapter Answer, Comments.

The following sequence of actions must be performed by the application in order to synchronize the query answer with BO15 and BD2 broadcasts.

1. Start subscribing for BO15 and BD2 broadcasts. Received broadcasts must not be processed until step 3. The user application must keep these broadcasts in an internal queue.
2. Send in the sequence of IQ19 queries (refer to "Answer, Comments" for more information).
3. When done with the IQ19 download of data the user application must handle the queued BO15 and BD2 broadcasts. They must be processed in the same order as they where received. The application has the correct information at the point when all queued broadcasts have been handled.
4. When all queued broadcasts have been processed the application can remove the usage of the internal queue. New broadcasts received should directly modify the (by the application) maintained database.

3.2.17.5 Structure Contents

query_hdr

Usage of fields in this structure:

Series should in the first query be filled in with any valid series. In the consecutive queries, the series given in the previous answer shall be used.

Items must be 1 if no filter is provided (Order Book Levels, Id), otherwise 2.

Size must be the total transaction size in bytes.

Order Book Levels, Query Data

Usage of fields in this structure:

Segment Number should in the first query be filled in with the value 1. In the consecutive queries, the Segment Number given in the previous answer shall be considered. See Answer, Comments for more information.

Order Book Levels, ID

This named structure is not mandatory in the query. If however provided, the series is used as a filter by the central system. Only instrument series matching the filter is returned in the answer.

> **Note:** There could only be zero or one occurrence of this structure in the query.

Usage of fields in this structure:

Series is filled in with a valid filter series. The following filters are allowed:

- Market (country and market code filled in. Other fields set to zero.)
- Instrument type (country, market and group code filled in. Other fields set to zero.)
- Instrument class (country, market, group and commodity code filled in. Other fields set to zero.)
- Instrument series (a valid series is provided.)

Block Size is not used and should be zero filled.

3.2.17.6 Return Codes

An IQ19 query may be aborted by the Marketplace. In this case only the reason for the query being aborted is returned to the sender.

cstatus	txstat	rcvbuf
Successful	Normal	list of price and order-book information – see above.
Transaction aborted	INFO_BADSEG	
Transacation aborted	...	

Please refer to the OMnet System Error Messages Reference for details about why transcations are aborted.

3.2.17.7 Answer Structure

The IA19 VIA has the following structure:

```
struct answer_hdr
struct sub_item_hdr
struct ob_levels_next_query  // Named struct no: 33032
Sequence {
   struct sub_item_hdr
   struct ob_levels_id  // Named struct no: 33002
   Sequence {
      struct sub_item_hdr
      Choice {
         struct ob_levels_price_volumes  // Named struct no: 33003
         struct ob_levels_order_number  // Named struct no: 33004
         struct ob_levels_total_quantity  // Named struct no: 33005
         struct ob_levels_no_of_orders  // Named struct no: 33033
         struct ob_levels_price  // Named struct no: 33006
         struct market_info_base  // Named struct no: 33034
         struct market_info_hke  // Named struct no: 33044
         struct ob_levels_closing  // Named struct no: 33031
      }
   }
}
```

3.2.17.8 Answer, Comments

After a successful IQ19 transaction, a list of price and order-book information is returned to the sender.

The Client should confirm to the following logic in order to download the data:

1. When the answer is received for the first query, the received Instance Number must be remembered.
2. From the answer structure, copy the Series Next to the series in the query_hdr of the subsequent query. If the Segment Number in the answer is greater than zero the value should be incremented by one and copied to the Segment Number in the subsequent query, otherwise (received Sequence Number is zero) the value of one should be copied.
3. Repeat step 2 until Next Instance Number in the answer is equal to the saved value from step 1 and the Segment Number in the answer is zero.

The query answer will contain relevant information to the current market state. Information fields not applicable to the current market state will be excluded from the answer.

3.2.17.9 Answer, Structure Contents

Depending on exchange configuration, either of **Order Book Levels, Price** or **Order Book Levels, Price and Volume** is distributed for a given instrument. The two of them are never distributed simultaneously for a given instrument. The exchange could however change the configuration intra day, causing a change of the distributed named structure. If for example the exchange decides to disable the volume distribution, the API client receives **Order Book Levels, Price and Volume** until this time and then directly a **Order Book Levels, Price**. The API client is in such case responsible to clean up the internal database and remove volume figures as these no longer are distributed by the exchange.

The interpretation of the various possible structures returned in the answer are the same as in BO15 and BD2 with some additions and exceptions described below.

Order Book Levels, Next Query

This structure is used by the application in order to perform a complete download of information as previously described.

Order Book Levels, ID

This structure defines the instrument series that succeeding structures relates to (up until the occurrence of a new Order Books Levels, ID structure).

For an example, please refer to BO15.

Order Book Levels, Price and Volumes

The Price masks are interpreted as bit fields where currently the low 10 bits are used. Bit 0 corresponds to the first ranked price, bit 1 to the second best ranked price, etc. For each bit set in the mask an array item is present. Bid items are placed before ask items in the array. Better rank prices are placed before lower ranked prices in the array. The field Items holds the total number of items within the array.

For examples, please refer to BO15.

Order Book Levels, Price

Each item in the array is of the structure type Order Book Levels, Price Item will be used the same way as Order Book Levels, Price and Volume when volume dissemination is not enabled. Then Order Book Levels, Price will be sent instead of Order Book Levels, Price and Volume.

Order Book Levels, Order Number

The order numbers provided in this structure are the order numbers for the first ranked bid and ask orders in the order book. Order number structures are only distributed if the exchange has enabled this distribution. If enabled, it applies to all instruments and is never changed intra day.

Order Book Levels, Total Quantity

are the total demand of all orders in the order book. Total quantity structures are only distributed if the exchange has enabled this distribution. If enabled, it applies to all instruments and is never changed intra day.

Order Book Levels, Closing

This structure is provided in the answer only if any of the included fields has a value set.

Usage of fields in this structure:

Price, Closing	The value of MIN_INT is used to indicate an undefined value while binary zero indicates a price of zero.

Order Book Levels, Number of Orders

The Number of Orders structure is only distributed if the exchange has enabled this distribution. If enabled, it applies to all instruments and is never changed intra day.

Usage of fields in this structure:

Number of orders	contains the number of orders on the price level that corresponds to this field's position in the array. For an example, please refer to BO15.

Market Info, Base

This structure is provided in the answer only if any of the included fields has a value set.

Usage of fields in this structure:

Price, Opening **Price, High** **Price, Low** **Price, Last**	The value of MIN_INT is used to indicate an undefined value while binary zero indicates a price of zero.

3.2.17.10 IQ19 Scenarios

The examples below illustrate the functionality of IA19 with respect to what information they may contain in different market situations.

Example

When the query is placed before the opening of the market - consequently no orders have been entered and no price or volume statistics are available - then the reply will consist only of the structures containing information, firstly the series the data relates to. Then for each series in the answer a possible closing price structure is sent. The reply also includes information about next query to send for more information.

In this case the answer will **not** contain any Order Book Levels, Price or Order Book Levels, Price and Volume structures, as the order book is empty. The answer will as well **not** include any Order Book Levels, Market Info structures as none of the included fields has a value set. The structure Order Book Levels, Closing will be included if the instrument series has a Closing price or Open balance set.

- Order Book Levels, Next Query
- Order Book Levels, ID
- Order Book Levels, Closing
- Order Book Levels, ID
- Order Book Levels, Closing
- Order Book Levels, ID (No Closing price or Open balance available)
- Order Book Levels, ID (No Closing price or Open balance available)
- Order Book Levels, ID
- Order Book Levels, Closing

Example

When the query is placed after the market has opened and there are orders in the market, and trades have been matched, then the sequence of named structures may look something like:

- Order Book Levels, Next Query
- Order Book Levels, ID
- Order Book Levels, Price and Volume (or Order Book Levels, Price)
- Order Book Levels, Market Info
- Order Book Levels, Closing
- Order Book Levels, Order Number (if enabled)
- Order Book Levels, Total Quantity (if enabled)
- Order Book Levels, Number of Orders

- Order Book Levels, ID
- Order Book Levels, Price and Volume (or Order Book Levels, Price)
- Order Book Levels, Market Info
- Order Book Levels, Closing
- Order Book Levels, Order Number (if enabled)
- Order Book Levels, Total Quantity(if enabled)
- Order Book Levels, Number of Orders
- Order Book Levels, ID

Example

When the query is placed after the market has opened and there are orders in the market but no trades have been matched, the sequence of named structures may look something like:

- Order Book Levels, Next Query
- Order Book Levels, ID
- Order Book Levels, Price and Volume (or Order Book Levels, Price)
- Order Book Levels, Closing (Closing price or Open balance set)
- Order Book Levels, Order Number (if enabled)
- Order Book Levels, Total Quantity (if enabled)
- Order Book Levels, Number of Orders
- Order Book Levels, ID
- Order Book Levels, Price and Volume (or Order Book Levels, Price)
- Order Book Levels, Closing (Closing price or Open balance set)
- Order Book Levels, Order Number (if enabled)
- Order Book Levels, Total Quantity (if enabled)
- Order Book Levels, Number of Orders
- Order Book Levels, ID

3.2.18 IQ42 [Trade Statistics QUERY]

3.2.18.1 Fingerprint

QUERY properties	
transaction type	IQ42
calling sequence	omniapi_query_ex
struct name	query_trade_statistics
facility	EP4
partitioned	false
answers	IA42

ANSWER properties	
transaction type	IA42
struct name	answer_trade_statistics
segmented	true

3.2.18.2 Purpose

This query is used to retrieve price and volume information for a business day.

3.2.18.3 Structure

The IQ42 QUERY has the following structure:

```
struct query_trade_statistics {
   struct transaction_type
   struct series  // Named struct no: 50000
   UINT16_T segment_number_n  // Segment Number
   char[8] date_s  // Date
   char[2] filler_2_s  // Filler
}
```

3.2.18.4 Usage and Conditions

In order to query the trade statistics for the current business day, a BI7 must have been received.

- BI7 with information type = 90. Signals that the daily prices statistics (high, low, last, …) are ready.
- BI7 with information type = 91. Signals that the settlement prices are ready.
- BI7 with information type = 100. Signals that all the end-of-day statistics are ready.

Historical dates can always be queried.

Series

is completed with **Country Number** and **Market Code**.

3.2.18.5 Return Codes

An IQ42 transaction may also be aborted by the Marketplace, in which case only the reason for the transaction being aborted is returned to the sender.

cstatus	txstat
Successful	INFO_SUCCESS
Successful	INFO_NOINFO
Successful	INFO_TODAYNOTAVAIL
Transaction aborted	INFO_BADSEG
Transaction aborted	...

Please refer to *System Error Messages Reference* for details about why transcations are aborted.

3.2.18.6 Answer Structure

The IA42 ANSWER has the following structure:

```
struct answer_trade_statistics {
   struct transaction_type
   UINT16_T segment_number_n  // Segment Number
   UINT16_T items_n  // Items
   Array ITEM [max no: 500] {
      struct series  // Named struct no: 50000
      INT32_T bid_premium_i  // Bid Premium
      INT32_T ask_premium_i  // Ask Premium
      INT32_T opening_price_i  // Price, First
      INT32_T settle_price_i  // Price, Settlement
      INT32_T last_price_i  // Price, Last
      INT32_T high_price_i  // Price, High
      INT32_T low_price_i  // Price, Low
      INT64_T volume_today_i  // Volume, Today
      INT64_T volume_yesterday_i  // Volume, Yesterday
      INT64_T turnaround_yesterday_u  // Turnover, Yesterday
      INT64_T turnaround_today_u  // Turnover, Today
      INT64_T open_balance_u  // Open Interest
      INT64_T revised_open_balance_u  // Revised Open Interest
      INT32_T volatility_i  // volatility
      INT32_T underlying_price_i  // Price, Underlying
      INT32_T corr_opening_price_i  // Price, Corresponding First
      INT32_T corr_high_price_i  // Price, Corresponding High
      INT32_T corr_low_price_i  // Price, Corresponding Low
      INT32_T corr_last_price_i  // Price, Corresponding Last
      UINT8_T bid_theo_c  // Bid, Theoretical Mark
      UINT8_T ask_theo_c  // Ask, Theoretical Mark
      char[2] filler_2_s  // Filler
   }
}
```

3.2.18.7 Answer, comments

Settle Price
Volatility

If the daily settlement price (Settle Price) and the Volatility is filled in or not, depends on the Exchange policy.

Revised Open Interest

The usage of this field depends on the Exchange policy. If the field is used, the prerequisite for it to be filled in is that a BI7 with Information Type 101 has been received, otherwise it will be empty.

The response is a list of series with Trade Information.

The information is available some time after the market has closed and the information reflects the status at the time of closing (after BI7 has been sent). Yesterday's volume and turnover are the real totals for the

previous day, including corrections and trades that have been made at the marketplace after the market has closed.

3.2.19 LQ3 [List with Version QUERY]

3.2.19.1 Fingerprint

QUERY properties	
transaction type	LQ3
calling sequence	omniapi_query_ex
struct name	query_list_ver
facility	EP4
partitioned	false
answers	LA3

ANSWER properties	
transaction type	LA3
struct name	answer_list_ver
segmented	true

3.2.19.2 Purpose

This transaction is used for transferring report files of a specific version.

3.2.19.3 Structure

The LQ3 QUERY has the following structure:

```
struct query_list_ver {
   struct transaction_type
   struct series  // Named struct no: 50000
   UINT16_T segment_number_n  // Segment Number
   char[8] date_s  // Date
   char[3] report_version_s  // Report Version
   char[3] filler_3_s  // Filler
   INT32_T info_type_i  // Information Type
}
```

3.2.19.4 Usage and conditions

Series

can be completed with Country Number and Market Code, but accepts blank and will then reply with reports for all markets.

3.2.19.5 Answer Structure

The LA3 ANSWER has the following structure:

```
struct answer_list_ver {
   struct transaction_type
   struct series  // Named struct no: 50000
   INT32_T info_type_i  // Information Type
   UINT16_T segment_number_n  // Segment Number
   char[40] list_name_s  // Name, List
   char[3] report_version_s  // Report Version
   CHAR filler_1_s  // Filler
   char[8] file_type_s  // File Type
   UINT16_T items_n  // Items
   char[50000] text_buffer_s  // Text, Buffer
}
```

3.2.19.6 Answer, comments

Item

the number of lines in the text buffer. Each line starts with a two-byte length word. The length word is word aligned.

Report Version

identifies the report version. The field has the format of a three character zero padded numeric value. It will be filled with space for report types with a date switch not equal to 3.

File Type

contains the suffix of the report file.

The response is received as a list of text lines.

3.2.20 LQ4 [Available Reports with Version QUERY]

3.2.20.1 Fingerprint

QUERY properties	
transaction type	LQ4
calling sequence	omniapi_query_ex
struct name	query_report_ver
facility	EP4
partitioned	false

QUERY properties	
answers	LA4

ANSWER properties	
transaction type	LA4
struct name	answer_report_ver
segmented	true

3.2.20.2 Purpose

This transaction is used for querying for available report versions.

3.2.20.3 Structure

The LQ4 QUERY has the following structure:

```
struct query_report_ver {
   struct transaction_type
   struct series  // Named struct no: 50000
   UINT16_T segment_number_n  // Segment Number
   char[8] date_s  // Date
   char[2] filler_2_s  // Filler
   INT32_T info_type_i  // Information Type
}
```

3.2.20.4 Usage and conditions

Series

can be completed with Country Number and Market Code, but accepts blank and will then reply with reports for all markets.

Information Type

- Information Type = 0 (returns all available reports for specified business date)
- Information Type = 256 (returns all possible reports for specified business date)
- Information Type = <specific report type number> (returns all available reports for specified business date and chosen report)

Note the difference between 'available' = already created and accessible via LQ3 and 'possible' = description about reports that can be created in the system.

A query about 'available' reports will return ALL versions if there are multiple reports for selected business date.

A query about 'possible' reports will return one item per possible type including a short description.

3.2.20.5 Answer Structure

The LA4 ANSWER has the following structure:

```
struct answer_report_ver {
   struct transaction_type
   UINT16_T segment_number_n  // Segment Number
   UINT16_T items_n  // Items
   Array ITEM [max no: 450] {
      struct series  // Named struct no: 50000
      INT32_T info_type_i  // Information Type
      char[8] date_s  // Date
      char[2] country_id_s  // Name, Country
      char[12] report_owner_s  // Report owner
      char[3] report_version_s  // Report Version
      char[32] name_s  // Name
      char[8] file_type_s  // File Type
      char[40] description_s  // Description
      UINT8_T ascii_bin_c  // ASCII or Binary
      char[8] created_date_s  // Date, Created
      char[6] created_time_s  // Time, Created
   }
}
```

3.2.20.6 Answer, comments

Report Version

identifies the report version. The field has the format of a three character zero padded numeric value. It will be filled with space for report types with a date switch not equal to 3. This field can be used to fill the sequence number field in a LQ3 transaction (and LQ259 or LQ2051 if applicable).

File Type

contains the suffix of the report file.

The response is received as a list of text lines.

3.2.21 MC4 [Quote Request with Volume TRANSACTION]

3.2.21.1 Fingerprint

TRANSACTION properties	
transaction type	MC4
calling sequence	omniapi_tx_ex
struct name	quote_request_vol
facility	EP0

TRANSACTION properties	
partitioned	true

3.2.21.2 Purpose

Normally a market maker responsibility does not include quotation responsibility for illiquid Series. But if someone wants to start trading in such a Series this function can be used. This quote request is sent to the Central System, and depending on the configuration, the Central System may broadcast this information..

3.2.21.3 Structure

The MC4 TRANSACTION has the following structure:

```
struct quote_request_vol {
   struct transaction_type
   struct series  // Named struct no: 50000
   UINT32_T block_n  // Block Size
   UINT8_T bid_or_ask_c  // Bid or Ask
   char[3] filler_3_s  // Filler
   INT64_T mp_quantity_i  // Quantity
}
```

3.2.21.4 Usage and conditions

Bid or Ask

When Bid or Ask is set to bid, it means that someone wants bid orders to be sent to the system. When set to 0, this means Bid **and** Ask.

Quantity

If Quantity is set to zero (0) the MC4 transaction should be interpreted like a quotation is requested with any volume.

Block Size

The MC4 may have either 0 as block size (all available block sizes will be taken into account), or a valid block size for the applicable instrument series.

3.2.21.5 Return Codes

After a successful MC4 transaction, the quote request is sent to connected applications through the MI4 broadcast.

cstatus	txstat		ordidt
Successful		Normal	Order ID for transaction

cstatus	txstat	ordidt
Transaction aborted	LM_MMSUP_NOT_LEGITIMATE Quote request not legitimate. Price exists in given Series.	-
Transaction aborted	... ...	-

An MC4 transaction may also be aborted by the Marketplace, in which case only the reason for the transaction being aborted is returned to the sender and the quote request is not broadcast.

Please refer to the **Error Messages Reference Manual** for details about why transactions are aborted.

3.2.22 MI4 [Quote Request with Volume Information BROADCAST]

3.2.22.1 Fingerprint

BROADCAST properties	
transaction type	MI4
calling sequence	omniapi_read_event_ext_ex or omniapi_read_event_block
struct name	quote_request_vol_info
info type	derivative

3.2.22.2 Related Messages

MC4

3.2.22.3 Purpose

The Quote Request with Volume Info is sent after a valid Quote Request with Volume. The broadcast is sent when the Quote Request is supposed to be sent to the entire market.

3.2.22.4 Structure

The MI4 BROADCAST has the following structure:

```
struct quote_request_vol_info {
   struct broadcast_type
   struct series  // Named struct no: 50000
   struct user_code
   UINT32_T block_n  // Block Size
   UINT8_T bid_or_ask_c  // Bid or Ask
   char[3] filler_3_s  // Filler
   INT64_T mp_quantity_i  // Quantity
}
```

3.2.22.5 Usage and conditions

The responsible market maker as well as other users may respond to this by sending in orders.

User

User in Quote Request broadcasts is the signature of the broker that sends a quote request transaction to the system. Depending on the configuration in CDB, on instrument type level, this field may be:

- Without counterpart: All user code fields are empty.
- With counterpart: Country and customer fields are filled.
- With counterpart and user: All user code fields are filled.

3.2.23 TR70 [Trade Ticker QUERY]

3.2.23.1 Fingerprint

QUERY properties	
transaction type	TR70
calling sequence	omniapi_query_ex
struct name	query_trade_ticker
facility	EP0
partitioned	true
answers	TA70

VIA properties	
transaction type	TA70
struct name	The message complies with the VIM concept and has no top struct. The sequence of possible structs is described in the Structure section.
segmented	true

3.2.23.2 Related Messages

BD70, BD71, TR71

3.2.23.3 Purpose

This query is used for recovering BD70.

3.2.23.4 Structure

The TR70 QUERY has the following structure:

```
struct query_trade_ticker {
   struct transaction_type
   struct series  // Named struct no: 50000
   struct search_series  // Of type: SERIES ; Named struct no: 50000
   struct timestamp  // Of type: TIME_SPEC
   UINT16_T segment_number_n  // Segment Number
   char[2] filler_2_s  // Filler
}
```

3.2.23.5 Usage and conditions

TR70 is a query corresponding to the BD70 broadcast that can be used for recovery purpose using publication timestamp. It is possible to download BD70 messages that have been distributed the current business day; previous days messages (trades) are not available. The query allows the following search criteria:

Time stamp: Download BD70 messages with a Publication timestamp equal or greater than the specified Time stamp.

Search Series: Download BD70 messages for a specific instrument series or according to a wildcard filter.

Start by sending a TR70 message with both Series fields set to all zeroes and the segment number field set to 1. This will return an TA70 with a set of BD70s back (if BD70 has been generated during the current trading day). If more TA70 segments exist to be returned, the segment number in the answer is larger than zero. If the segment number in the answer is zero, the next series field can be used as input for the TR70 series field. The segment number has to be set to 1 again and the procedure must be updated until both the series field and the segment number are zero.

Series

is used for routing.

3.2.23.6 Answer Structure

The TA70 VIA has the following structure:

```
struct answer_next_series_hdr {
   struct transaction_type
   struct next_series  // Of type: SERIES ; Named struct no: 50000
   UINT16_T segment_number_n  // Segment Number
   UINT16_T items_n  // Items
   UINT16_T size_n  // Size
   char[2] filler_2_s  // Filler
}
Sequence {
   struct item_hdr
   Sequence {
      struct sub_item_hdr
      Choice {
         struct basic_trade_ticker  // Named struct no: 34401
         struct extended_trade_ticker  // Named struct no: 34402
         struct trade_report_trade_ticker  // Named struct no: 34403
         struct half_trade_ticker  // Named struct no: 34405
      }
   }
```

```
}
```

3.2.23.7 Answer, comments

Deals previously distributed in BD70 and later canceled will not be included in the answer.

Deals previously distributed in BD70 and later amended will only be distributed with information relating to the period after the amendment.

In the struct basic_trade_ticker, the **Match group number** field should not be used.

3.2.24 TR71 [Amended Trades QUERY]

3.2.24.1 Fingerprint

QUERY properties	
transaction type	TR71
calling sequence	omniapi_query_ex
struct name	query_amended_trades
facility	EP0
partitioned	true
answers	TA71

VIA properties	
transaction type	TA71
struct name	The message complies with the VIM concept and has no top struct. The sequence of possible structs is described in the Structure section.
segmented	true

3.2.24.2 Related Messages

BD70, BD71, TR70

3.2.24.3 Purpose

This query is used for recovering BD71.

3.2.24.4 Structure

The TR71 QUERY has the following structure:

```
struct query_amended_trades {
   struct transaction_type
   struct series  // Named struct no: 50000
   UINT16_T segment_number_n  // Segment Number
```

```
    char[2] filler_2_s  // Filler
}
```

3.2.24.5 Answer Structure

The TA71 VIA has the following structure:

```
struct answer_next_series_hdr {
   struct transaction_type
   struct next_series  // Of type: SERIES ; Named struct no: 50000
   UINT16_T segment_number_n  // Segment Number
   UINT16_T items_n  // Items
   UINT16_T size_n  // Size
   char[2] filler_2_s  // Filler
}
Sequence {
   struct item_hdr
   Sequence {
      struct sub_item_hdr
      Choice {
         struct trade_ticker_amend  // Named struct no: 34406
         struct basic_trade_ticker  // Named struct no: 34401
         struct half_trade_ticker  // Named struct no: 34405
      }
   }
}
```

3.2.24.6 Answer, comments

In the struct basic_trade_ticker, the **Match group number** field should not be used.

3.2.25 UI1 [Application Status TRANSACTION]

3.2.25.1 Fingerprint

TRANSACTION properties	
transaction type	UI1
calling sequence	omniapi_tx_ex
struct name	application_status
facility	EP0
partitioned	false

3.2.25.2 Purpose

The Application Status Transaction is used to inform the central Marketplace that the user's trading application is ready to trade. A trading application is ready to trade when it has logged on and all necessary initializations are executed.

3.2.25.3 Structure

The UI1 TRANSACTION has the following structure:

```
struct application_status {
   struct transaction_type
   struct series  // Named struct no: 50000
   INT32_T application_status_i  // Status, Application
}
```

3.2.25.4 Usage and Conditions

Series

is a reserved field and is not in use.

Status

must be equal to one.

After a successful UI1, the market place is aware of that the client is ready to trade. There are no return codes.

3.2.26 UQ1 [Partition QUERY]

3.2.26.1 Fingerprint

QUERY properties	
transaction type	UQ1
calling sequence	omniapi_query_ex
struct name	query_partition
facility	EP0
partitioned	false
answers	UA1

ANSWER properties	
transaction type	UA1
struct name	answer_partition
segmented	true

3.2.26.2 Purpose

This query will return all partition information.

3.2.26.3 Structure

The UQ1 QUERY has the following structure:

```
struct query_partition {
   struct transaction_type
   struct series  // Named struct no: 50000
   UINT16_T segment_number_n  // Segment Number
   char[2] filler_2_s  // Filler
}
```

3.2.26.4 Answer Structure

The UA1 ANSWER has the following structure:

```
struct answer_partition {
   struct transaction_type
   UINT16_T segment_number_n  // Segment Number
   UINT16_T items_n  // Items
   Array ITEM [max no: 100] {
      struct server_partition {
         char[20] server_name_s  // Server Name
         struct transaction_type_low {
            struct transaction_type
         }
         struct transaction_type_high {
            struct transaction_type
         }
         struct series_fields_used {
            UINT8_T country_c  // Country Number
            UINT8_T market_c  // Market Code
            UINT8_T instrument_group_c  // Instrument Group
            UINT8_T modifier_c  // Modifier
            UINT16_T commodity_n  // Commodity Code
            UINT16_T expiration_date_n  // Date, Expiration
            INT32_T strike_price_i  // Strike Price
         }
         struct partition_low
         struct partition_high
         INT32_T event_type_i  // Stimuli Event
      }
   }
}
```

3.2.26.5 Answer, comments

Transaction Type, Low
Transaction Type, High

defines a range of transactions in one partition.

Series Field Used

shows all fields that are used, both in the **Partition Low, Binary Series** field and in the **Partition High, Binary Series**field. Value 1 in a field means that the field is used, value 0 means that the field is not used in the partition.

Partition, Low Binary Series
Partition High, Binary Series

defines a range of consecutive series in one partition.

Partition Low may be used to fill in the **Series** field in corresponding query.

If only country_c is set in **Series Field Used**, then the value in country_c in **Partition, Low Binary Series** is to fill instance_c in corresponding query.

OMnet Event Type

is used as facility number in the call to omniapi_query.

3.2.27 UQ9 [BI7 Signals Sent QUERY]

3.2.27.1 Fingerprint

QUERY properties	
transaction type	UQ9
calling sequence	omniapi_query_ex
struct name	query_bi7_signals_sent
facility	EP0
partitioned	false
answers	UA9

ANSWER properties	
transaction type	UA9
struct name	answer_bi7_signals_sent
segmented	true

3.2.27.2 Purpose

The purpose of this query is to retrieve all Signal Binary Information (BI7) signals sent for the date given in the query.

3.2.27.3 Structure

The UQ9 QUERY has the following structure:

```
struct query_bi7_signals_sent {
   struct transaction type
   struct search series
```

```
   UINT16_T segment_number_n  // Segment Number
   char[8] business_date_s  // Date, Business
   UINT16_T seq_num_srm_n  // Sequence number for SRM
}
```

3.2.27.4 Answer Structure

The UA9 ANSWER has the following structure:

```
struct answer_bi7_signals_sent {
   struct transaction_type
   UINT16_T segment_number_n  // Segment Number
   UINT16_T items_n  // Items
   Array ITEM [max no: 1000] {
      struct series  // Named struct no: 50000
      INT32_T info_type_i  // Information Type
      char[8] business_date_s  // Date, Business
      char[8] clearing_date_s  // Clearing Date
      char[8] sent_date_s  // Date, Sent
      char[6] sent_time_s  // Time, Sent
      UINT16_T seq_num_srm_n  // Sequence number for SRM
   }
}
```

3.2.28 UQ12 [Business Date QUERY]

3.2.28.1 Fingerprint

QUERY properties	
transaction type	UQ12
calling sequence	omniapi_query_ex
struct name	query_business_date
facility	EP1
partitioned	false
answers	UA12

ANSWER properties	
transaction type	UA12
struct name	answer_business_date
segmented	false

3.2.28.2 Purpose

The purpose of this query is to get the current business date, the UTC date and time.

3.2.28.3 Structure

The UQ12 QUERY has the following structure:

```
struct query_business_date {
   struct transaction type
}
```

3.2.28.4 Usage and Conditions

Note that the retrieved information is not for time synchronization purposes. For synchronization purposes use NTP (Network Time Protocol).) The answer also contains the exchanges TZ-variable and the current offset between UTC and the local time specified in the TZ-variable. The answer also consists of the current system version.

3.2.28.5 Answer Structure

The UA12 ANSWER has the following structure:

```
struct answer_business_date {
   struct transaction type
   char[16] omex version s  // OMEX Version
   char[8] business date s  // Date, Business
   char[8] utc date s  // UTC, Date
   char[6] utc time s  // UTC, Time
   char[40] tz variable s  // TZ-Variable
   char[2] filler 2 s  // Filler
   INT32 T utc offset i  // UTC, Offset
}
```

3.2.28.6 Answer, comments

The response received is the current business date and the current system version.

3.2.29 UQ13 [BI27 Broadcasts Sent QUERY]

3.2.29.1 Fingerprint

QUERY properties	
transaction type	UQ13
calling sequence	omniapi_query_ex
struct name	query_bi27_broadcasts_sent
facility	EP1
partitioned	false
answers	UA13

ANSWER properties	
transaction type	UA13
struct name	answer_bi27_broadcasts_sent
segmented	true

3.2.29.2 Purpose

The purpose of this query is to retrieve all Clearing Message (BI27) broadcasts that have been sent on the current business date.

3.2.29.3 Structure

The UQ13 QUERY has the following structure:

```
struct query_bi27_broadcasts_sent {
   struct transaction_type
   UINT16_T segment_number_n  // Segment Number
   char[2] filler_2_s  // Filler
}
```

3.2.29.4 Answer Structure

The UA13 ANSWER has the following structure:

```
struct answer_bi27_broadcasts_sent {
   struct transaction_type
   UINT16_T segment_number_n  // Segment Number
   CHAR filler_1_s  // Filler
   UINT8_T items_c  // Item
   Array ITEM1 [max no: 50] {
      UINT16_T broadcast_number_n  // Broadcast Number
      UINT8_T country_c  // Country Number
      UINT8_T market_c  // Market Code
      UINT16_T items_n  // Items
      char[2] filler_2_s  // Filler
      Array ITEM2 [max no: 10] {
         char[80] free_text_80_s  // Text , Free
      }
   }
}
```

3.2.29.5 Answer, comments

The text buffer contains 80 character lines, completed with trailing spaces, but no carriage return or other control characters.

3.2.30 UQ14 [BI81 Broadcasts Sent QUERY]

3.2.30.1 Fingerprint

QUERY properties	
transaction type	UQ14
calling sequence	omniapi_query_ex
struct name	query_bi81_broadcasts_sent
facility	EP0
partitioned	false
answers	UA14

VIA properties	
transaction type	UA14
struct name	The message complies with the VIM concept and has no top struct. The sequence of possible structs is described in the Structure section.
segmented	true

3.2.30.2 Purpose

The purpose of this transaction is to retrieve sent BI81 broadcasts.

3.2.30.3 Structure

The UQ14 QUERY has the following structure:

```
struct query_bi81_broadcasts_sent {
   struct transaction_type
   struct series  // Named struct no: 50000
   UINT16_T segment_number_n  // Segment Number
   UINT8_T message_information_type_c  // Message Information, Type
   UINT8_T message_priority_c  // Message, Priority
   char[8] date_s  // Date
   UINT32_T from_sequence_number_u  // From Sequence Number
   UINT32_T to_sequence_number_u  // To Sequence Number
   struct search_series
   UINT8_T update_status_note_c  // Status Note, Update
   char[3] filler_3_s  // Filler
}
```

3.2.30.4 Usage and Conditions

Message Information Type

should state the message type of interest. If filled with a zero, all message types are returned.

Series, search

Series can either be zero-filled, by which means a wildcard search on all series and markets, or point to a specific series or market.

Message Priority

should state the priority of the messages of interest. For example, if you only want to retrieve messages with high priority, state 3 for Message Priority. If filled with a zero, messages with all priorities are returned.

From Sequence Number

From Sequence Number should contain the first message number of interest. From Sequence Number must be filled in with a value greater than 0, since the first message is always one.

To Sequence Number

To Sequence Number should contain the last message number of interest. If To Sequence Number is filled with a zero, all remaining messages for the specified date are returned.

3.2.30.5 Answer Structure

The UA14 VIA has the following structure:

```
struct answer_segment_hdr
Sequence {
   struct item_hdr
   Sequence {
      struct sub_item_hdr
      Choice {
         struct message_core_info  // Named struct no: 35001
         struct message_information  // Named struct no: 35002
         struct destination_item  // Named struct no: 35003
         struct document_url  // Named struct no: 35004
      }
   }
}
```

3.2.30.6 Answer, Structure Contents

Message Meta-Data (35001)

Fields usage in this structure:

Sequence Number A serial number defined by the central system. The number starts with 1 every day.

Date Time, External	Time stamps in UTC.

3.2.31 UQ15 [Instrument Status QUERY]

3.2.31.1 Fingerprint

QUERY properties	
transaction type	UQ15
calling sequence	omniapi_query_ex
struct name	query_instrument_status
facility	EP1
partitioned	false
answers	UA15

ANSWER properties	
transaction type	UA15
struct name	answer_instrument_status
segmented	true

3.2.31.2 Purpose

The query returns the status for a Market, Instrument Type, Instrument Class, Series and Underlying or for all instrument levels.

3.2.31.3 Structure

The UQ15 QUERY has the following structure:

```
struct query_instrument_status {
   struct transaction_type
   struct series  // Named struct no: 50000
   UINT16_T segment_number_n  // Segment Number
   UINT16_T state_level_e  // Level
}
```

3.2.31.4 Usage and Conditions

The query search the parameters set in the Series and the Level parameters.

The instrument status is updated by the BI41 broadcast.

More information about the trading session handling is found in section “Trading Session” in *OMnet Message Reference, Introduction.*

Series

Series should be completed according to the table below to be able to identify a specific Market, Instrument Type, Instrument Class, Series or Underlying.

Any of the fields filled with binary zero, is regarded as wildcard for that field. If all fields in the series are filled with binary zeroes, the complete instrument status for all markets, instrument types, instrument classes, series and underlyings will be returned. Expiration date and Strike price can in some cases be zero for a series.

What to identify	Complete the following fields
Market	Country Number Market Code
Instrument Type	Country Number Market Code Instrument Group
Instrument Class	Country Number Market Code Instrument Group Commodity Code
Series	Country Number Market Code Instrument Group Commodity Code Expiration Date Price, Strike
Underlying	Commodity Code

Level

The Level field is supplied as a means to separate an instrument class from a series.

If, for example, the value 2 is sent in, only session states set on instrument type will be returned.

3.2.31.5 Return Codes

After a successful UQ15 query, a list of instrument status is returned to the sender.

A UQ15 transaction may also be aborted. In that case, only the reason for the transaction being aborted is returned to the sender.

Cstatus	txstat	Ordidt	rcvbuf
Successful	Normal	-	list of parameters - see below

Cstatus	txstat	Ordidt	rcvbuf
Transaction aborted	Error number that is translated by the OMnet routine get_error_message	-	-

Please refer to *System Error Messages Reference* for details about why transcations are aborted.

3.2.31.6 Answer Structure

The UA15 ANSWER has the following structure:

```
struct answer_instrument_status {
   struct transaction type
   UINT16_T segment_number_n  // Segment Number
   UINT16_T items_n  // Items
   Array ITEM [max no: 1000] {
      struct series  // Named struct no: 50000
      UINT16_T state_number_n  // Trading State Number
      UINT16_T state_level_e  // Level
   }
}
```

3.2.31.7 Answer, comments

Series

Series, completed with one of the following:

Market	Country Number Market Code
Instrument Type	Country Number Market Code Instrument Group
Instrument Class	Country Number Market Code Instrument Group Commodity Code
Series	Country Number Market Code Instrument Group Commodity Code Expiration Date Price, Strike
Underlying	Commodity Code

Segment Number

To get the next segments increase the segment number by one. The Segment Number is set to zero in the answer if there is no more to fetch.

3.2.32 UQ20 [BI73 Signals Sent QUERY]

3.2.32.1 Fingerprint

QUERY properties	
transaction type	UQ20
calling sequence	omniapi_query_ex
struct name	query_bi73_signals_sent
facility	EP0
partitioned	false
answers	UA20

ANSWER properties	
transaction type	UA20
struct name	answer_bi73_signals_sent
segmented	true

3.2.32.2 Purpose

This transaction is used to query which BI73 broadcasts have been sent on a certain day.

3.2.32.3 Structure

The UQ20 QUERY has the following structure:

```
struct query_bi73_signals_sent {
   struct transaction_type
   struct search_series
   UINT16_T segment_number_n  // Segment Number
   char[8] business_date_s  // Date, Business
   char[8] clearing_date_s  // Clearing Date
   UINT16_T seq_num_srm_n  // Sequence number for SRM
}
```

3.2.32.4 Usage and Conditions

The query is sent to the Supervision subsystem.

3.2.32.5 Answer Structure

The UA20 ANSWER has the following structure:

```
struct answer_bi73_signals_sent {
   struct transaction_type
   UINT16_T segment_number_n  // Segment Number
   UINT16_T items_n  // Items
   Array ITEM [max no: 1000] {
      struct series  // Named struct no: 50000
      INT32_T info_type_i  // Information Type
      char[8] business_date_s  // Date, Business
      char[8] clearing_date_s  // Clearing Date
      char[8] sent_date_s  // Date, Sent
      char[6] sent_time_s  // Time, Sent
      UINT16_T seq_num_srm_n  // Sequence number for SRM
   }
}
```

3.2.32.6 Answer, comments

Series

In the Series field the market and country must be filled whereas the rest of the Series should be filled with zeroes.

3.2.33 UQ21 [BI7 Signals Sent CL QUERY]

3.2.33.1 Fingerprint

QUERY properties	
transaction type	UQ21
calling sequence	omniapi_query_ex
struct name	query_bi7_signals_sent_cl
facility	EP0
partitioned	false
answers	UA21

ANSWER properties	
transaction type	UA21
struct name	answer_bi7_signals_sent
segmented	true

3.2.33.2 Purpose

The purpose of this query is to request BI7s for a particular clearing date.

3.2.33.3 Structure

The UQ21 QUERY has the following structure:

```
struct query_bi7_signals_sent_cl {
   struct transaction type
   struct search series
   UINT16_T segment_number_n  // Segment Number
   char[8] clearing_date_s  // Clearing Date
   UINT16_T seq_num_srm_n  // Sequence number for SRM
}
```

3.2.33.4 Answer Structure

The UA21 ANSWER has the following structure:

```
struct answer_bi7_signals_sent {
   struct transaction type
   UINT16_T segment_number_n  // Segment Number
   UINT16_T items_n  // Items
   Array ITEM [max no: 1000] {
      struct series  // Named struct no: 50000
      INT32_T info_type_i  // Information Type
      char[8] business_date_s  // Date, Business
      char[8] clearing_date_s  // Clearing Date
      char[8] sent_date_s  // Date, Sent
      char[6] sent_time_s  // Time, Sent
      UINT16_T seq_num_srm_n  // Sequence number for SRM
   }
}
```

3.3 Order Management

3.3.1 BD1 [Deals in the Market BROADCAST]

3.3.1.1 Fingerprint

BROADCAST properties	
transaction type	BD1
calling sequence	omniapi_read_event_ext_ex or omniapi_read_event_block
struct name	deal_user
info type	instrument class
segmented	true
virtual underlying	true

3.3.1.2 Purpose

This subscription returns information on deals closed in the market.

3.3.1.3 Structure

The BD1 BROADCAST has the following structure:

```
struct deal_user  // Named struct no: 34251
```

3.3.1.4 Usage and Conditions

Order Number

By checking the order number, the remote application knows if its “own” order pertains to a deal.

Sequence Number

is a non-consecutive (non-strictly) increasing number per series (thus two consecutive BD1s can have the same sequence number). If the Firm Order Book broadcast BO5 is not used, this can be used to synchronize the answer to MQ8. The BD1 message with sequence number not exceeding sequence number for any order in MQ8 answer should be discarded. Since MQ8 are segmented queries, different orders in the series can be marked with different sequence numbers.

One item is returned for each deal in the broadcast.

Ticker applicability

Event	Action
Reception of BD1	Use it in ticker as usual.

3.3.2 BO5 [Firm Order Book VIB]

3.3.2.1 Fingerprint

VIB properties	
transaction type	BO5
calling sequence	omniapi_read_event_ext_ex or omniapi_read_event_block
struct name	The message complies with the VIM concept and has no top struct. The sequence of possible structs is described in the Structure section.
info type	instrument dedicated
segmented	true

3.3.2.2 Purpose

All order-related activities for a firm are disseminated via this directed broadcast, for example, when a user enters or changes an order or an order being matched by another order. Thereby it is possible for each user to keep an internal order book for the firm.

3.3.2.3 Structure

The BO5 VIB has the following structure:

```
struct broadcast_hdr
Sequence {
   struct sub_item_hdr
   Choice {
      struct alter_trans  // Named struct no: 34009
      struct block_order_trans  // Named struct no: 34006
      struct block_order_trans_p  // Named struct no: 34106
      struct block_price_trans  // Named struct no: 34007
      struct block_price_trans_p  // Named struct no: 34107
      struct broker_trans  // Named struct no: 34013
      struct broker_trans_p  // Named struct no: 34113
      struct hv_alter_trans  // Named struct no: 34010
      struct hv_alter_trans_p  // Named struct no: 34110
      struct hv_order_trans  // Named struct no: 34005
      struct hv_order_trans_p  // Named struct no: 34105
      struct hv_price_2_trans  // Named struct no: 34001
      struct hv_price_2_trans_p  // Named struct no: 34101
      struct multi_order_response  // Named struct no: 34906
      struct combo_trans_part  // Named struct no: 34907
      struct combo_trans_part_p  // Named struct no: 34908
      struct order_change_combined  // Named struct no: 34902
      struct order_change_separate  // Named struct no: 34903
      struct order_chg_sep_trans_ack  // Named struct no: 34919
      struct bb_change_separate  // Named struct no: 34909
      struct order_price_change  // Named struct no: 34905
      struct order_return_info  // Named struct no: 34904
      struct order_trans  // Named struct no: 34004
      struct price_trans  // Named struct no: 34003
      struct price_trans_p  // Named struct no: 34103
      struct price_2_trans  // Named struct no: 34002
      struct segment_instance_number  // Named struct no: 34901
      struct stop_order_trans  // Named struct no: 34017
      struct stop_order_trans_p  // Named struct no: 34117
      struct long_stop_order_trans  // Named struct no: 34024
      struct long_stop_order_trans_p  // Named struct no: 34124
      struct tm_trade_rpt_trans  // Named struct no: 34014
      struct trade_report_trans  // Named struct no: 34018
      struct order_status  // Named struct no: 34910
      struct order_state  // Named struct no: 34913
      struct delete_trans  // Named struct no: 34011
      struct delete_trans_p  // Named struct no: 34111
      struct trade_report_1_trans  // Named struct no: 34021
      struct trade_report_1_trans_p  // Named struct no: 34119
      struct trade_report_2_trans  // Named struct no: 34022
      struct prio_crossing_trans  // Named struct no: 34020
```

```
            struct prio_crossing_trans_p  // Named struct no: 34118
            struct order_info  // Named struct no: 34917
            struct combo_acc_trans  // Named struct no: 34016
            struct combo_acc_trans_p  // Named struct no: 34116
            struct mp_trade_price  // Named struct no: 34918
            struct cppx_initiation_trans  // Named struct no: 34023
            struct cppx_initiation_trans_p  // Named struct no: 34123
            struct cppx_confirmation_trans  // Named struct no: 34028
            struct cppx_confirmation_trans_p  // Named struct no: 34125
            struct order_trade_info  // Named struct no: 34920
            struct order_leg_trade_info  // Named struct no: 34921
            struct time_in_force  // Named struct no: 34807
            struct exchange_info  // Named struct no: 50004
            struct free_text  // Named struct no: 34801
            struct clearing_info  // Named struct no: 34802
            struct linked_order_leg  // Named struct no: 34803
            struct linked_order_leg_number  // Named struct no: 34809
            struct order_owner  // Named struct no: 34804
            struct indicative_quote  // Named struct no: 34025
            struct multi_leg_order_insert  // Named struct no: 34817
            struct multi_leg_order_leg_number  // Named struct no: 34818
            struct multi_leg_order_insert_p  // Named struct no: 34819
        }
    }
```

3.3.2.4 Usage and Conditions

In order to maintain the real-time order book from the BO5 information, the user application must use MQ8 to download a baseline of the order book. The sequence for this is described in the MQ8 section of this document.

The broadcast structure contains a variable number of substructures. The broadcast thus contains one broadcast header structure followed by one or more variable structures.

The basic concept of this broadcast is to disseminate exactly the same information as sent in one order transaction with corresponding transaction status and order number. These broadcasts should therefore be processed in the same way as if the application itself had entered the order transaction.

In other words, the different order structures contained in this broadcast are simply copies of the corresponding structures sent to the central system, holding all information about the order.

Note, however, that for transactions that can submit either an absolute or a delta quantity, such as MO33 or MO36, BO5 will always return the resulting absolute quantity and the delta quantity (enum) field will always state that it is an absolute quantity.

Several BO5 broadcasts may belong together. The segment number is set to 1 for the first segment, 2 for the second segment, etc. The last segment is always set to zero. Thus, for single segment broadcasts, the segment number is 0 (zero.)

Note: Multi-item orders (such as MO36 and MO30) will be split up in separate order items in the resulting BO5 broadcast.

Note:

BO5 broadcasts may be duplicated. Applications should therefore make use of the sequence number to discard duplicates when receiving BO5 broadcasts.

Since there is one series of sequence number per partition, this has to be done on a per partition basis.

Sequence number and partition fields are available in the segment_instance_number substructure.

3.3.2.5 Structure Contents

Segment Instance Number

The **Instance** field denotes the matching engine partition that the broadcast originates from. It is set to 0 (zero) if only one instance exists.

Order Change Combined

When an order entered into the system is modified (such as traded) in any way before being added to the order book, a struct is sent in the same broadcast. Two consecutive Order Change Combined items are generated in case of a Fill and Kill order with residual quantity. The first part states the remaining quantity after matching while the second part indicates that the rest of the quantity is deleted.

Order Change Separate

The Order Change Separate structure is sent out due to changes in quantity of orders residing in the order book. As with Order Change Combined, the size and total size fields describe the remaining volumes of the order.

Order Change Price

The Order Change Price structure is sent out for orders for which the price has been changed (combo box orders.)

Order Return Info

The Order Return Info structure is limited to one per broadcast.

Multi Order Response

The Multi Order Response structure is sent in a BO5 originating from a received block order MO36. It contains information about failed orders of the block order. Successful items are sent in other structures.

3.3.3 BO10 [Equilibrium Price Update BROADCAST]

3.3.3.1 Fingerprint

BROADCAST properties	
transaction type	BO10
calling sequence	omniapi_read_event_ext_ex or omniapi_read_event_block
struct name	equil_price_update
info type	instrument class

BROADCAST properties	
virtual underlying	true

3.3.3.2 Purpose

This subscription provides information on changes in the equilibrium prices. Each broadcast includes a list of updated series where all series belongs to the same Instrument Class.

3.3.3.3 Structure

The BO10 BROADCAST has the following structure:

```
struct equil_price_update {
   struct broadcast_type
   UINT16_T items_n  // Items
   char[2] filler_2_s  // Filler
   Array ITEM [max no: 27] {
      struct series  // Named struct no: 50000
      INT64_T equilibrium_quantity_i  // Equilibrium Volume
      INT32_T equilibrium_price_i  // Price, Equilibrium
      INT32_T best_bid_premium_i  // Best Bid Price, Preopening
      INT32_T best_ask_premium_i  // Best Ask Price, Pre-opening
      INT64_T best_bid_quantity_i  // Best Bid Volume, Preopening
      INT64_T best_ask_quantity_i  // Best Ask Volume, Pre-opening
      UINT8_T matching_price_type_c  // Matching Price Type
      char[3] filler_3_s  // Filler
   }
}
```

3.3.3.4 Usage and conditions

Price fields

If any Price field has bit 31 set (the highest bit, MIN_INT) while all other bits are zero, this means that no price is available. Note the use of different bit patterns to distinguish a price that is not available from a price that is zero. For the value of zero, set all bits to zero.

Equilibrium Volume
Best Bid Volume, Pre- Opening
Best Ask Volume, Pre-Opening

These fields are only updated if enabled by the exchange.

Best Bid Price, Pre-Opening
Best Ask Price, Pre-Opening

These fields are only updated if enabled by the exchange.

The usage of the BO10 subscription is defined by the exchange.

3.3.4 BO14 [Order Book Levels VIB]

3.3.4.1 Fingerprint

VIB properties	
transaction type	BO14
calling sequence	omniapi_read_event_ext_ex or omniapi_read_event_block
struct name	The message complies with the VIM concept and has no top struct. The sequence of possible structs is described in the Structure section.
info type	instrument class
virtual underlying	true

3.3.4.2 Related Messages

IQ18

3.3.4.3 Purpose

The subscriptions for BO14 provides information on changes in the order book, but the data has been further processed by the central system before it is broadcasted.

3.3.4.4 Structure

The BO14 VIB has the following structure:

```
struct broadcast_hdr
Sequence {
   struct sub_item_hdr
   Choice {
      struct ob_levels_order_number  // Named struct no: 33004
      struct ob_levels_sequence_number  // Named struct no: 33001
      struct ob_levels_total_quantity  // Named struct no: 33005
      struct ob_levels_no_of_orders  // Named struct no: 33033
      struct ob_levels_undisclosed_quantity  // Named struct no: 33041
      struct ob_levels_price_volumes  // Named struct no: 33003
      struct ob_levels_id  // Named struct no: 33002
      struct ob_levels_hidden_quantity  // Named struct no: 33007
      struct ob_levels_price  // Named struct no: 33006
   }
}
```

3.3.4.5 Usage and Conditions

Only the total volume for each Premium is given, or only the Premium and no order related information is included. The information could also be subject to a holdback in case multiple order-book changes could be sent in a single broadcast. The exchange can also configure whether volumes will be present in the

broadcasts or not. If volumes are enabled it may be disseminated according to a dissemination step table configured by the exchange.

With respect to functionality, BO14 and BO15 are interchangeable broadcasts, but with separate configurations. Depending on how the exchange has configured the broadcasts they will differ in content and holdback.

Some data within the broadcasts is only provided if the exchange has enabled the distribution of it.

It is for example possible to specify the BO14 broadcast with a price depth of 5 and the BO15 broadcast with a depth of 1 and thereby provide two different subscription alternatives depending of bandwidth utilization.

In order to maintain a real time database of the BO14 information the user application can use IQ18 to download a baseline of the information.

In order to maintain a real time database of the BO15 information the user application must use IQ19 to download a baseline of the information. The sequence for this is described in the IQ18/IQ19 section of this document.

3.3.4.6 Structure Contents

Depending on exchange configuration, either of **Order Book Levels, Price** or **Order Book Levels, Price and Volumes** is distributed for a given instrument. The two of them are never distributed simultaneously for a given instrument. The exchange could however change the configuration intra day, causing a change of the distributed named structure. If for example the exchange decides to disable the volume distribution, the API client receives Order Book Levels, Price and Volumes up until this time and then directly an Order Book Levels, Price. The API client is in this case responsible to clean up the internal database and remove volume figures as these no longer are distributed by the exchange.

Order Book Levels, Sequence Number (OB_LEVELS_SEQUENCE_NUMBER)

This structure is always present as the first variable structure in a BO14 / BO15 broadcast. It occurs exactly once in a BO14 / BO15 broadcast. It should not be processed by the application.

Order Book Levels, ID (OB_LEVELS_ID)

This structure defines the instrument series that succeeding variable structures relates to (up until the occurrence of a new Order Book Levels, ID variable structure.)

The following example describes the relations between ID and succeeding structures:

Example

...	(previous series)
OB Levels, Id	Sets series A
OB Levels, Price and Volumes	Prices and volumes for series A
OB Levels, Order Number	Order numbers for series A
OB Levels, Id	Sets series B
OB Levels, Order Number	Order numbers for series B
OB Levels, Id	Sets series C
OB Levels, Price and Volumes	Prices and volumes for series C

...	(succeeding series)

Fields usage in this structure:

Block Size defines the block size of the Series. Block size 0 indicates the All or None order book. The distribution of All or None orders is enabled by the exchange.

Order Book Levels, Price and Volumes (OB_LEVELS_PRICE_VOLUMES)

Fields usage in this structure:

Premium Levels propagates the currently distributed order-book depth for this instrument series. Possible values are currently in the range from 0 to 5. A value of 0 means that the exchange doesn't distribute any prices at all. A value of 1 means that the exchange distributes the first ranked price level. A value of 2 means that the exchange distributes the 2 best prices levels, etc. The Premium Levels could be changed during the day for a given instrument series. In the case where the Premium Level is decreased the application must itself clear all price levels beyond the current level.

Demands Populated indicates if the distribution of volumes are enabled or disabled for the different price levels.

Premium If set to bit 31 (highest bit), while all other bits are zero, (MIN_INT) indicates that no Premium is available. This differs from the value of zero (all bits zero) indicating a Premium price of zero. Some exchanges allow orders to be placed with a price of zero. The use of different bit patterns for No-Premium and Zero Price-Premium makes it possible to distinguish them from each other. Non-Premium is distributed either because there are no orders in the order book, or because orders that have not been priced to a fix value exist (i.e. they were entered as market/auction orders).

Price mask, bid
Price mask, ask are interpreted as bit fields where currently the low 10 bits are used. Bit 0 corresponds to the first ranked price, bit 1 to the second best ranked price, etc. For each bit set in the mask an array item is present. All Bid items are placed before any ask items in the array. Better rank prices are placed before lower ranked prices in the array. The Items field holds the total number of items within the array.

Example

If the bid price mask has the value 3 (bit 0 and 1 set) and the ask price mask has the value 4 (bit 2 set), the array consists of the following items:

- Array[0] : Premium and demand for bid level 1
- Array[1] : Premium and demand for bid level 2
- Array[2] : Premium and demand for ask level 3

Example

If the bid price mask has the value 0 and the ask price mask has the value 31 (bit 0 to 4 set), the array consists of the following items:

- Array[0] : Premium and demand for ask level 1
- Array[1] : Premium and demand for ask level 2
- Array[2] : Premium and demand for ask level 3
- Array[3] : Premium and demand for ask level 4
- Array[4] : Premium and demand for ask level 5

Example

If the bid price mask has the value 16 (bit 4 set) and the ask price mask has the value 24 (bit 3 and 4 set), the array consists of the following items:

- Array[0] : Premium and demand for bid level 5
- Array[1] : Premium and demand for ask level 4
- Array[2] : Premium and demand for ask level 5

Order Book Levels, Price (OB_LEVELS_PRICE)

will be used in the same way as, but instead of, as Order Book Levels, Price and Volumes when volume dissemination is not enabled.

Order Book Levels, Order Number (OB_LEVELS_ORDER_NUMBER)

Order number variable structure is only distributed if the exchange has enabled this distribution. If enabled, it applies to all instruments and is never changed intra day.

Fields usage in this structure:

Order Number, bid **Order Number, ask**	are the order numbers for the first ranked bid and ask orders in the order book.

Order Book Levels, Total Quantity (OB_LEVELS_TOTAL_QUANTITY)

The Total Quantity variable structure is only distributed if the exchange has enabled this distribution. If enabled, it applies to all instruments and is never changed intra day.

Fields usage in this structure:

Total Bid Quantity **Total Ask Quantity**	are the total demand of all orders in the order book.

Order Book Levels, Hidden Quantity (OB_LEVELS_HIDDEN_QUANTITY)

The information in this structure shows if there are any hidden quantities for the bid or ask prices distributed.

This structure may neither be distributed for all instrument nor for all exchanges.

Order Book Levels, Number of Orders (OB_LEVELS_NO_OF_ORDERS)

The Number of Orders variable structure is only distributed if the exchange has enabled this distribution. If enabled, it applies to all instruments and is never changed intra day.

The information in this structure hold the number of individual orders at each bid and ask level.

Fields usage in this structure:

Premium Levels propagates the currently distributed order book depth for this instrument series.

Bid Orders, Total Number is the total number of individual bid orders in the order book for this instrument series.

Ask Orders, Total Number is the total number of individual ask orders in the order book for this instrument series.

Mask, Bid
Mask, Ask are interpreted as bit fields where currently the low 10 bits are used. Bit 0 corresponds to the first ranked price, bit 1 to the second best ranked price, etc. For each bit set in the mask an array item is present. All Bid items are placed before any ask items in the array. Better rank prices are placed before lower ranked prices in the array. The Items field holds the total number of items within the array.

Example

If the bid price mask has the value 3 (bit 0 and 1 set) and the ask price mask has the value 4 (bit 2 set), the array consists of the following items:

- Array[0] : Number of individual orders for bid level 1
- Array[1] : Number of individual orders for bid level 2
- Array[2] : Number of individual orders for ask level 3

Example

If the bid price mask has the value 0 and the ask price mask has the value 31 (bit 0 to 4 set), the array consists of the following items:

- Array[0] : Number of individual orders for ask level 1
- Array[1] : Number of individual orders for ask level 2
- Array[2] : Number of individual orders for ask level 3
- Array[3] : Number of individual orders for ask level 4
- Array[4] : Number of individual orders for ask level 5

Example

If the bid price mask has the value 16 (bit 4 set) and the ask price mask has the value 24 (bit 3 and 4 set), the array consists of the following items:

- Array[0] : Number of individual orders for bid level 5
- Array[1] : Number of individual orders for ask level 4

- Array[2] : Number of individual orders for ask level 5

3.3.5 BO15 [Order Book Levels VIB]

3.3.5.1 Fingerprint

VIB properties	
transaction type	BO15
calling sequence	omniapi_read_event_ext_ex or omniapi_read_event_block
struct name	The message complies with the VIM concept and has no top struct. The sequence of possible structs is described in the Structure section.
info type	instrument class
virtual underlying	true

3.3.5.2 Related Messages

IQ19

3.3.5.3 Purpose

The subscriptions for BO15 provides information on changes in the order book, but the data has been further processed by the central system before it is broadcasted.

3.3.5.4 Structure

The BO15 VIB has the following structure:

```
struct broadcast_hdr
Sequence {
   struct sub_item_hdr
   Choice {
      struct ob_levels_order_number  // Named struct no: 33004
      struct ob_levels_sequence_number  // Named struct no: 33001
      struct ob_levels_total_quantity  // Named struct no: 33005
      struct ob_levels_no_of_orders  // Named struct no: 33033
      struct ob_levels_undisclosed_quantity  // Named struct no: 33041
      struct ob_levels_price_volumes  // Named struct no: 33003
      struct ob_levels_id  // Named struct no: 33002
      struct ob_levels_price  // Named struct no: 33006
      struct ob_levels_hidden_quantity  // Named struct no: 33007
   }
}
```

3.3.5.5 Usage and conditions

Only the total volume for each Premium is given, or only the Premium and no order related information is included. The information could also be subject to a holdback in case multiple order-book changes could be sent in a single broadcast. The exchange can also configure whether volumes will be present in the broadcasts or not. If volumes are enabled it may be disseminated according to a dissemination step table configured by the exchange.

Volume dissemination step is a concept to reduce the need for new broadcasts if the available volume is only changed slightly while the price remains the same. For consecutive volume intervals, individual dissemination steps are defined. When a volume is broadcasted, it will be rounded down to the nearest value that is an integer times the step. If an order-book update results in the same price and rounded volume, there will be no broadcast sent.

With respect to functionality, BO14 and BO15 are interchangeable broadcasts, but with separate configurations. Depending on how the exchange has configured the broadcasts they will differ in content and holdback.

Some data within the broadcasts is only provided if the exchange has enabled the distribution of it.

It is for example possible to specify the BO14 broadcast with a price depth of 5 and the BO15 broadcast with a depth of 1 and thereby provide two different subscription alternatives depending of bandwidth utilization.

In order to maintain a real time database of the BO14 information the user application can use IQ18 to download a baseline of the information.

In order to maintain a real time database of the BO15 information the user application must use IQ19 to download a baseline of the information. The sequence for this is described in the IQ18/IQ19 section of this document.

3.3.5.6 Structure contents

Depending on exchange configuration, either of **Order Book Levels, Price** or **Order Book Levels, Price and Volumes** is distributed for a given instrument. The two of them are never distributed simultaneously for a given instrument. The exchange could however change the configuration intra day, causing a change of the distributed named structure. If for example the exchange decides to disable the volume distribution, the API client receives Order Book Levels, Price and Volumes up until this time and then directly an Order Book Levels, Price. The API client is in this case responsible to clean up the internal database and remove volume figures as these no longer are distributed by the exchange.

Order Book Levels, Sequence Number (OB_LEVELS_SEQUENCE_NUMBER)

This structure is always present as the first variable structure in a BO14 / BO15 broadcast. It occurs exactly once in a BO14 / BO15 broadcast. It should not be processed by the application.

Order Book Levels, ID (OB_LEVELS_ID)

This structure defines the instrument series that succeeding variable structures relates to (up until the occurrence of a new Order Book Levels, ID variable structure.)

The following example describes the relations between ID and succeeding structures:

Example

...	(previous series)
OB Levels, Id	Sets series A
OB Levels, Price and Volumes	Prices and volumes for series A
OB Levels, Order Number	Order numbers for series A
OB Levels, Id	Sets series B
OB Levels, Order Number	Order numbers for series B
OB Levels, Id	Sets series C
OB Levels, Price and Volumes	Prices and volumes for series C
...	(succeeding series)

Fields usage in this structure:

Block Size defines the block size of the Series. Block size 0 indicates the All or None order book. The distribution of All or None orders is enabled by the exchange.

Order Book Levels, Price and Volumes (OB_LEVELS_PRICE_VOLUMES)

Fields usage in this structure:

Premium Levels propagates the currently distributed order-book depth for this instrument series. Possible values are currently in the range from 0 to 5. A value of 0 means that the exchange doesn't distribute any prices at all. A value of 1 means that the exchange distributes the first ranked price level. A value of 2 means that the exchange distributes the 2 best prices levels, etc. The Premium Levels could be changed during the day for a given instrument series. In the case where the Premium Level is decreased the application must itself clear all price levels beyond the current level.

Demands Populated indicates if the distribution of volumes are enabled or disabled for the different price levels.

Premium If set to bit 31 (highest bit), while all other bits are zero, (MIN_INT) indicates that no Premium is available. This differs from the value of zero (all bits zero) indicating a Premium price of zero. Some exchanges allow orders to be placed with a price of zero. The use of different bit patterns for No-Premium and Zero Price-Premium makes it possible to distinguish them from each other. Non-Premium is distributed either because there are no orders in the order book, or because orders that have not been priced to a fix value exist (i.e. they were entered as market orders).

Price mask, bid
Price mask, ask are interpreted as bit fields where currently the low 10 bits are used. Bit 0 corresponds to the first ranked price, bit 1 to the second best ranked price, etc. For each bit set in the mask an array item is present. All Bid items are placed before any ask items in the array. Better rank prices are placed before lower ranked prices in the array. The Items field holds the total number of items within the array.

Example

If the bid price mask has the value 3 (bit 0 and 1 set) and the ask price mask has the value 4 (bit 2 set), the array consists of the following items:

- Array[0] : Premium and demand for bid level 1
- Array[1] : Premium and demand for bid level 2
- Array[2] : Premium and demand for ask level 3

Example

If the bid price mask has the value 0 and the ask price mask has the value 31 (bit 0 to 4 set), the array consists of the following items:

- Array[0] : Premium and demand for ask level 1
- Array[1] : Premium and demand for ask level 2
- Array[2] : Premium and demand for ask level 3
- Array[3] : Premium and demand for ask level 4
- Array[4] : Premium and demand for ask level 5

Example

If the bid price mask has the value 16 (bit 4 set) and the ask price mask has the value 24 (bit 3 and 4 set), the array consists of the following items:

- Array[0] : Premium and demand for bid level 5
- Array[1] : Premium and demand for ask level 4
- Array[2] : Premium and demand for ask level 5

Order Book Levels, Price (OB_LEVELS_PRICE)

will be used in the same way as, but instead of, as Order Book Levels, Price and Volumes when volume dissemination is not enabled.

Order Book Levels, Order Number (OB_LEVELS_ORDER_NUMBER)

Order number variable structure is only distributed if the exchange has enabled this distribution. If enabled, it applies to all instruments and is never changed intra day.

Fields usage in this structure:

Order Number, bid **Order Number, ask**	are the order numbers for the first ranked bid and ask orders in the order book.

Order Book Levels, Total Quantity (OB_LEVELS_TOTAL_QUANTITY)

The Total Quantity variable structure is only distributed if the exchange has enabled this distribution. If enabled, it applies to all instruments and is never changed intra day.

Fields usage in this structure:

Total Bid Quantity
Total Ask Quantity

are the total demand of all orders in the order book.

Order Book Levels, Hidden Quantity (OB_LEVELS_HIDDEN_QUANTITY)

The information in this structure shows if there are any hidden quantities for the bid or ask prices distributed.

This structure may neither be distributed for all instrument nor for all exchanges.

Order Book Levels, Number of Orders (OB_LEVELS_NO_OF_ORDERS)

The Number of Orders variable structure is only distributed if the exchange has enabled this distribution. If enabled, it applies to all instruments and is never changed intra day.

The information in this structure hold the number of individual orders at each bid and ask level.

Fields usage in this structure:

Premium Levels

propagates the currently distributed order book depth for this instrument series.

Bid Orders, Total Number

is the total number of individual bid orders in the order book for this instrument series.

Ask Orders, Total Number

is the total number of individual ask orders in the order book for this instrument series.

Mask, Bid
Mask, Ask

are interpreted as bit fields where currently the low 10 bits are used. Bit 0 corresponds to the first ranked price, bit 1 to the second best ranked price, etc. For each bit set in the mask an array item is present. All Bid items are placed before any ask items in the array. Better rank prices are placed before lower ranked prices in the array. The Items field holds the total number of items within the array.

Example

If the bid price mask has the value 3 (bit 0 and 1 set) and the ask price mask has the value 4 (bit 2 set), the array consists of the following items:

- Array[0] : Number of individual orders for bid level 1
- Array[1] : Number of individual orders for bid level 2
- Array[2] : Number of individual orders for ask level 3

Example

If the bid price mask has the value 0 and the ask price mask has the value 31 (bit 0 to 4 set), the array consists of the following items:

- Array[0] : Number of individual orders for ask level 1
- Array[1] : Number of individual orders for ask level 2
- Array[2] : Number of individual orders for ask level 3
- Array[3] : Number of individual orders for ask level 4
- Array[4] : Number of individual orders for ask level 5

Example

If the bid price mask has the value 16 (bit 4 set) and the ask price mask has the value 24 (bit 3 and 4 set), the array consists of the following items:

- Array[0] : Number of individual orders for bid level 5
- Array[1] : Number of individual orders for ask level 4
- Array[2] : Number of individual orders for ask level 5

3.3.6 BO38 [Market Maker Protection Settings Information BROADCAST]

3.3.6.1 Fingerprint

BROADCAST properties	
transaction type	BO38
calling sequence	omniapi_read_event_ext_ex or omniapi_read_event_block
struct name	market_maker_protection_info
info type	dedicated

3.3.6.2 Purpose

When the market maker protection settings change or there is a protection trigger, the Market Maker will be informed about the new protecting settings in a BO38 broadcast.

3.3.6.3 Structure

The BO38 BROADCAST has the following structure:

```
struct market_maker_protection_info {
   struct broadcast_type
   struct trading_code
   struct series  // Named struct no: 50000
   INT64_T calc_quantity_protection_q  // Calculated Quantity Protection
   INT64_T calc_delta_protection_q  // Calculated Delta Protection quantity
}
```

3.3.6.4 Usage and Conditions

Actual Volume Protection quantity

Will be zero when parameters are set.

Actual Delta Protection quantity

Will be zero when parameters are set.

3.3.7 BO55 [Trade Report Notification VIB]

3.3.7.1 Fingerprint

VIB properties	
transaction type	BO55
calling sequence	omniapi_read_event_ext_ex or omniapi_read_event_block
struct name	The message complies with the VIM concept and has no top struct. The sequence of possible structs is described in the Structure section.
info type	dedicated

3.3.7.2 Purpose

When the first part of a trade report is received by the system, this broadcast is used to notify the participant specified as counterparty in the trade report about this.

3.3.7.3 Structure

The BO55 VIB has the following structure:

```
struct broadcast_hdr
Sequence {
   struct sub_item_hdr
   Choice {
      struct trade_report_base  // Named struct no: 34808
      struct exchange_info  // Named struct no: 50004
   }
}
```

3.3.7.4 Usage and Conditions

For two-party trade reports, no notification is disseminated.

The application receiving this notification can use the information to fill in the fields in a corresponding trade report.

Order number

is the order number of the first part of the trade report.

Counterparty

is the participant entering the first side of the trade report.

3.3.8 BO99 [Block Transaction Response BROADCAST]

3.3.8.1 Fingerprint

BROADCAST properties	
transaction type	BO99
calling sequence	omniapi_read_event_ext_ex or omniapi_read_event_block
struct name	block_order_response
info type	dedicated

3.3.8.2 Purpose

This broadcast is sent when a block order or block quote is only partly executed. The response holds detailed information on the part that was not executed.

> **Note:** If all orders in the block are rejected, the BO99 is not sent.

For more detailed information see MO36.

3.3.8.3 Structure

The BO99 BROADCAST has the following structure:

```
struct block_order_response {
   struct broadcast_type
   QUAD_WORD order_number_u  // Order Number
   UINT8_T items_c  // Item
   char[3] filler_3_s  // Filler
   Array ITEM [max no: 100] {
      INT32_T transaction_status_i  // Transaction, Status
      INT32_T trans_ack_i  // Transaction, Acknowledgement
      UINT8_T item_number_c  // Item Number
      char[3] filler_3_s  // Filler
   }
}
```

3.3.8.4 Usage and Conditions

The BO99 is similar to the answer response used in MO30/MO414. However, note the BO99 is only sent for failed MO36 items not for MO30 items.

The Transaction Status will be 1 (true) if the order was successful, otherwise it will be zero. In the Order acknowledge, information regarding the state of the order will be sent.

cstatus	txstat
Successful	Bit 9 set in any combination with Bit 5, Bit 6 and Bit 7 – circuit breaker started

cstatus	txstat
Transaction aborted	GEN_CDC_INT_CLOSED – Instrument Type is not open for this Transaction Type
Transaction aborted	...

3.3.9 MO4 [Order Deletion TRANSACTION]

3.3.9.1 Fingerprint

TRANSACTION properties	
transaction type	MO4
calling sequence	omniapi_tx_ex
struct name	delete_trans
facility	EP0
partitioned	true

3.3.9.2 Purpose

The delete transaction is used to remove one or more orders from the Order Book. In contrast to the alter transaction, this transaction can affect several orders at once - a group of orders to be deleted can be specified.

3.3.9.3 Structure

The MO4 TRANSACTION has the following structure:

```
struct delete_trans  // Named struct no: 34011
```

3.3.9.4 Usage and Conditions

If **one** specific order is to be deleted, the following fields must be specified:

- **Series** (must be fully completed)
- **Order Number**
- **Bid or Ask**

When a **group** of orders is to be deleted the group is defined by the following fields:

- **Series**
- **Whose**
- **Bid or Ask**

Series

can be completed either as Underlying (**Country Number** plus **Market Code** plus **Commodity Code**) or as **Instrument Class**.

Client

Character "*" and "%" are allowed in the Client field. This is only valid for this transaction.

Whose

is used to specify My, Our, My Client's or Our Client's Order. In this way all combinations of Whose order can be obtained, i.e. My or Our Order, and in addition the Combination Client. Fields to be omitted should be filled with NUL characters.

Note: In MO4 (and MO44) the Client field may contain the wildcard characters * (substitutes zero or more characters) or % (substitutes a single character).

My Orders indicates that I, a broker from Company XX, wish to delete my orders specifically. The expression Our Orders indicates that I remove all Company XX orders regardless of who has placed the order, including orders placed by Exchange's staff on Company XX's account.

In addition, it is possible to remove a particular client's order. In this instance either the client for whom I have placed the order is specified, or the client of Company XX regardless of who placed the order is specified.

Type of order	Fields to be completed
All my orders	Customer and User
All our orders	Customer
All my orders for a specific client	Customer, User and Client
All our orders for a specific client	Customer and Client

Note:

All character fields must be space padded up to the total length of the field.

Bid or Ask

is set to either Bid, Ask or both Bid and Ask.

Example

- Series is completed with Country Number = 1 , Market Code = 1 and Commodity Code = 1 .
- Whose is completed with the Customer and User field.
- Bid or Ask is completed with bid.

The result will be that all my bids referring to that underlying are removed from the Order Book.

Example

- Series is completed inclusive Instrument Class with 6 4 3 1001.
- Whose is completed with the Customer and Client field.

- Bid or Ask is set to zero.

The result will be that all Company “Customer’s” bid orders and ask requests for client “client” concerning some currency forwards in that instrument class will be removed.

3.3.9.5 Return Codes

An MO4 transaction may also be aborted by the Marketplace, in which case only the reason for the transaction being aborted is returned to the sender.

Cstatus	Txstat	Ordidt
Successful	For multi order delete: The two least significant bytes in the field specify the number of orders deleted, or zero if no order exists. The two most significant bytes in the field specify the number of orders that should have been deleted but still remain in the order book due to market constraints.	-
Successful	For single order delete: n – number of contracts before deletion (for specific order deletion only, the whole Series, the order number and whether the order is a Bid or Ask order must be specified).	-
Transaction aborted	GEN_CDC_INT_CLOSED - Instrument Type is not open for this Transaction Type.	-
Transaction Aborted	...	-

Please refer to *System Error Messages Reference* for details about why transcations are aborted.

3.3.10 MO31 [Order Entry TRANSACTION]

3.3.10.1 Fingerprint

TRANSACTION properties	
transaction type	MO31
calling sequence	omniapi_tx_ex
struct name	hv_order_trans
facility	EP0
partitioned	true

3.3.10.2 Purpose

This transaction is used for placing orders in the Order book.

3.3.10.3 Structure

The MO31 TRANSACTION has the following structure:

```
struct hv_order_trans  // Named struct no: 34005
```

3.3.10.4 Usage and Conditions

A Fill or Kill order is indicated by having Block Size and Validity Time set to zero. The Central System will interpret this type of order as if the whole of ”Size” is to be closed immediately - if this does not occur, the whole order is discarded.

A Fill and Kill order is indicated by having Block Size set to a valid block size and Validity Time set to zero.

Series

must be completed for MO31 transactions.

Block Size

is the minimum closing unit accepted. The system can handle three block sizes except for block size zero. Valid block sizes can be retrieved from the system.

Client

is not validated before entered in the Order Book. However, for matched trades, the field Client is interpreted as the account identity in the clearing system.

Character "*" and "%" are **not** allowed in the Client field.

Quantity
Total Size Volume

Total Size Volume is the total volume of the order, that is, both the hidden and the shown volumes.

When the Quantity and the Total Size Volume are different, the value entered for the Quantity will be the shown size in the Order book and the Total Size Volume will show the total number of contracts for the order.

By setting Total Size Volume equal to Quantity, the order is sent as a hidden volume order with the whole quantity shown.

By setting Total Size Volume equal to zero, the order is sent as a normal order with Quantity number of contracts, that is, without hidden volume. A hidden size order cannot be converted to an order without hidden volume and vice versa.

When the shown contracts are all traded, the number of Quantity new contracts will be displayed in the Order book and the corresponding number will be decreased from the Total Size Volume amount. The possibility to have a hidden size is controlled on an instrument type level from the CDB.

Orders placed using MO31 cannot be managed using MO36/MO37.

For instruments with a commodity_type set to interest rate, the Quantity and Total Size Volume must be stated in multiples of nominal value.

Quantity and Total Size Volume must be stated in multiples of valid Block Size.

The maximum value for Quantity and Total Size Volume is set on instrument level in the CDB.

Validity Time

can be set to a certain value or to zero. Please see the Detailed Field Information chapter for details.

The Exchange defines a minimum Validity Time for an order.

Open Close Request

If the highest bit in the requested position field (open_close_req_c) is set, the open_close_req_c field will be interpreted as a bit map instead of an enum.

Bit 0 represents leg 1 in the combination, bit 1 legs 2, and so on.

It will only be possible to specify “open” or “close” for each leg. “0” for “open” and “1” for “close.”

Example

Let us assume a scenario with Open_close_req_c (8 bit):

Bit 7					...	Bit 1	Bit 0
1	0	0	0	0	1	0	1

If the highest bit is set, the field will be interpreted as a bitmap.

In the example above, the 1st and 3rd leg in the combination shall be close (1) and the rest of the legs shall be open (0).

Note:

The highest bit is only allowed for orders that are entered into combo series. If an order is entered in a series that is not a combo series, the normal **open_close_req_c** values must be specified.

Example 2:

Volume	10
Quantity	10
Block Size	10
Premium	7
Validity Time	rest of the day
Order Type	1

As Validity Time is not zero and the Order Type is 1, the order will be placed in the Order book or a deal is made immediately. A deal will only be accepted for blocks of ten, i.e. the whole Volume in this example.

Example 3:

Volume	10
Quantity	10

Block Size	1
Premium	7
Validity Time	rest of the day
Order Type	1

When the order is matched, some parts of the order may be closed and the remainder is placed in the Order book. As the block is one, the order can result in up to ten different deals.

Example 4:

Volume	100
Quantity	10
Block Size	1
Premium	7
Validity Time	rest of the day
Order Type	1

Closing will only be accepted for Block Sizes of one. The part of the order which is not closed will be placed in the Order book for the duration of the order with a displayed size of 10 (if at least ten contracts remain, otherwise the remaining size is displayed).

Example 5:

Volume	10
Quantity	10
Block Size	1
Premium	7
Validity Time	0
Order Type	1

Parts of the order (as much as possible) will be closed and the remainder will be discarded.

3.3.10.5 Return Codes

After a successful MO31 transaction, an order number and information regarding the state of the order will be returned to the sender. For a Standard Combination Order, each leg will get the same order number.

Cstatus	Txstat	ordidt
Successful	0 – fail-over in progress transaction status unknown	order number
Successful	1 – no part of the order placed in the Order book and no part closed	order number

Cstatus	Txstat	ordidt
Successful	2 – the whole order closed	order number
Successful	3 – the order partially closed and nothing placed in the Order book	order number
Successful	4 – the whole order placed in the Order book	order number
Successful	6 – the order partially placed in the Order book and partially closed	order number
Successful	17 – circuit breaker started, no part of the order placed in the Order book and no part closed	order number
Successful	19 – circuit breaker started, the order partially closed and nothing placed in the Order book	order number
Transaction aborted	GEN_CDC_INT_CLOSED – Instrument Type is not open for this Transaction Type	-
Transaction aborted	...	-

An MO31 transaction may also be aborted by the Marketplace, in which case only the reason for the transaction being aborted is returned to the sender.

Please refer to *System Error Messages Reference* for details about why transcations are aborted.

3.3.11 MO33 [Alteration TRANSACTION]

3.3.11.1 Fingerprint

TRANSACTION properties	
transaction type	MO33
calling sequence	omniapi_tx_ex
struct name	hv_alter_trans
facility	EP0
partitioned	true

3.3.11.2 Purpose

MO33 is used to alter an order in the order book.

3.3.11.3 Structure

The MO33 TRANSACTION has the following structure:

```
struct hv_alter_trans  // Named struct no: 34010
```

3.3.11.4 Usage and Conditions

Only one existing order, which is referred to by a unique order number, can be altered at a time.

> **Note:** The exchange itself specifies the usage and restriction of MO33.

Order Number, Series, and Bid or Ask must be filled in in order to identify the order in the order book.

The other fields must be completed only if they should be altered. The alteration is stated as the new value required for the specified order in the Order book. The remaining fields, which should not be altered, are set to zero. Fields with ASCII designations are completed with NULL characters (= binary zero) if the field should be ignored. Note that only the first character is checked for the NULL character. If this is NULL, the field is considered not to be altered.

The Bid or Ask field can be used to specify the Bid or Ask side if orders with the same Order Number exist on both sides.

> **Note:** This means that the Premium of an order can never be changed to a market price that is zero. For the same reason, the Validity Time of an order can never be changed to zero. A zero setting indicates that a field is to be left unchanged in the Order book.

It is possible to carry out several alterations at the same time.

Although the transaction superficially resembles a transaction that places an order and an Order Number, this does not imply that all the fields in an order placed in the order book can be altered.

The following fields may be altered:

- **Quantity**
- **Total Volume**
- **Validity Time**
- **Client**
- **Customer Information**
- **Open or Close, requested**
- **Give up member**
- **Exchange Info**
- **Premium**

Total Volume is used when changing hidden size orders. Then **Total Volume** specifies the total size of the order while **Quantity** specifies the shown size. **Total Volume** is always zero if hidden size/iceberg orders are not used at the exchange. Refer to the examples below.

An original order with no hidden size cannot be altered to become hidden size order and vice versa. When altering the time validity of an order, the system will take the new time relative to when the alteration was received by the central system. For example, if an order is placed on day 1 with a time validity of 5:22 (indicating it is valid for 22 days), and then altered on day 3 to 5:2 (indicating that is valid for only 2 days), then it will be set to expire before the market starts on day 5 (2 days after the alteration transaction).

The **Exchange Info** field may be overlaid with an exchange-specific struct, but it still follows the rules for ASCII fields here. Thus, if the first character of the exchange_info field is set to NULL (binary zero), the exchange_info from the existing order is used.

Changes to Quantity/Total Volume

Any change to the premium of an order, or increasing quantities if allowed by the market will result in the order losing its priority in the market.

When changing quantities there are two options: delta and absolute. Delta changes amend the quantity/total volume of an order by the given amount, positive to increase the quantity, negative to reduce the quantity. Absolute changes means that the quantity/total volume should be set to the value in the quantity/total volume field.

This is selected by using the field delta_quantity_c field. Setting this field to “1” indicates that absolute quantities should be used, setting to “2” indicates that quantities should be amended by the given delta amount.

If the delta_quantity_c is set to "2" and the resulting quantity of the order will be zero or less, the order is deleted from the order book.

Note:

The delta_quantity_c field must be filled in with either “1” or “2” in order for the transaction to be accepted.

Example

Original Order	Amendment	Result
mp_quantity_i =1000 total volume_i = 0	delta_quantity_c =1 mp_quantity_i = 600 total volume_i = 0	mp_quantity_i = 600 total volume_i = 0
mp_quantity_i = 1000 total volume_i = 0	delta_quantity_c = 2 mp_quantity_i = 600 total volume_i = 0	mp_quantity_i = 1600 total volume_i = 0
mp_quantity_i =1000 total volume_i = 0	delta_quantity_c = 2 mp_quantity_i = -600 total volume_i = 0	mp_quantity_i = 400 total volume_i = 0
mp_quantity_i = 2000 total volume_i = 10000	delta_quantity_c = 2 mp_quantity_i = -600 total volume_i = 10000	mp_quantity_i =1400 total volume_i = 20000
mp_quantity_i = 2000 total volume_i = 10000	delta_quantity_c = 2 mp_quantity_i = -2000 total volume_i = -10000	Order deleted
mp_quantity_i = 2000 total volume_i = 10000	delta_quantity_c = 2 mp_quantity_i = -2000 total volume_i = 0	Order deleted
mp_quantity_i = 2000 total volume_i = 10000	delta_quantity_c = 2 mp_quantity_i = -2000 total volume_i = 3000	Order deleted

Balance Quantity

If the field balance_quantity_i is provided the system checks this quantity against the existing total volume of the order prior to applying the amendment. If the two match then the amendment is applied, if not, an error is returned.

When altering the time validity of an order, the system will take the new time relative to when the alter transaction was received by the Central System.

Example

An order is placed with time validity 5:22 on day 1. On day 3 it is altered to time validity 5:2. This causes the order to expire before the market starts on day 5. Validity time is defined in **Detailed Field Descriptions.**

3.3.11.5 Return Codes

An MO33 transaction may also be aborted by the Marketplace, in which case only the reason for the transaction being aborted is returned to the sender.

cstatus	txstat	orderidt
Successful	n Number of contracts before the order was changed, or zero if no order exists.	-
Transaction aborted	GEN_CDC_INT_CLOSED Instrument Type is not open for this Transaction Type.	-
Transaction aborted	MP_MATCH_INV_ALTER Alter is not allowed with retained priority.	-
Transaction aborted	...	-

After a successful MO33 transaction the number of contracts before the order is changed, zero if no order exists, is returned to the sender. If no order from your own participant is found with the keys specified (Order Number, Series, Bid or Ask), the alter operation is still considered successful but will return txstat=0. In this case no order is altered.

Please refer to *System Error Messages Reference* for details about why transcations are aborted.

Note: Not changing anything at all as well as attempting to change fields that are not allowed to alter might be considered a successful operation from the return codes point of view. Consequently, return values as pointed out in this section, or alternatively an error code will be returned. In either case the order is unchanged. A successful MO33 does not change the order, an order alteration broadcast may be sent out.

3.3.12 MO36 [Two-Sided Price Quotation Block TRANSACTION]

3.3.12.1 Fingerprint

TRANSACTION properties	
transaction type	MO36
calling sequence	omniapi_tx_ex
struct name	block_price_trans
facility	EP0
partitioned	true

3.3.12.2 Purpose

This transaction is used for placing up to configurable maximum number of two-sided quotations in the Order book.

3.3.12.3 Structure

The MO36 TRANSACTION has the following structure:

```
struct block_price_trans  // Named struct no: 34007
```

3.3.12.4 Usage and Conditions

The maximum number of orders that can be placed in one transaction is retrieved from the system by using the Query Maximum Block Order Sizes (MQ99) query. The transaction is rejected, if the maximum limit is exceeded. The range of consecutive series allowed to be sent in one MO36 can be received using the UQ1 transaction.

> **Note:** The MO36 transaction does not handle combinations.

Previous quotes are replaced by new quotes if they exist.

Series

The Series must be completed for MO36 transactions. It is mandatory to fill in the Series and it has to be set to anyone of the series contained in the quotation block structure. The orders in a block transaction may be on different series as long as those series are traded in the same partition.

Order Number, Bid
Order Number, Ask

It is not possible to have more than one bid order and one ask order per series in the transaction.

The bid order to be replaced from the Order book is specified by Order Number, Bid and **Series**. The ask order to be replaced from the Order book is specified by Order Number, Ask and Series. To replace the whole two-sided quote, specify Order Number, Bid and Order Number, Ask together with Series.

Bid Quantity
Ask Quantity
Bid Total Volume
Ask Total volume

By setting Bid/Ask Total Volume to zero or equal to Bid/Ask Quantity, the order is sent as a normal order without hidden size.

When the Bid/Ask Quantity and the Bid/Ask Total Volume are different, the value entered for the Bid/Ask Quantity will be the shown size in the order book and the Bid/Ask Total Volume will show the total number of contracts for the order.

When the displayed contracts are all traded, the number of Bid/Ask Quantity new contracts will be displayed in the order book and the corresponding number will be decreased from the Bid/Ask Total Volume amount. The possibility to have a hidden size is controlled on an instrument type level from the CDB.

By setting both the Bid/Ask Quantity and Bid/Ask Total Volume to zero, the previous order in the block is deleted and not replaced by a new one.

Bid/Ask Quantity and Bid/Ask Total Volume must be stated in multiples of valid block sizes.

Block Size

is the minimum closing unit accepted. The system can handle two block sizes, except for block size zero. Valid block sizes can be retrieved from the system.

Validity Time

can be set to a certain value or to zero. The latter indicates that, after matching, no parts of the order will remain in the Order book, i.e. the size that can be closed is closed in a deal, and the rest is discarded.

When the Validity Time is set to a value other than zero, this value is to be stated in the following form:

- Number of Days
- the Rest of the Day
- as Long as the Series is Valid

The Exchange defines a minimum Validity Time for an order.

Client

is not validated before entered in the Order Book. However, for matched trades, the field Client is interpreted as the account identity in the clearing system.

Delta quantity

can have the value1 or 2 and specifies how the bid and ask quantity will be interpreted. A Delta quantity of 1 means that quantity is treated as an absolute quantity. For example, a quote 100@20 (quantity is 100) and Delta quantity of 1 will become a quote in orderbook of 100@20. A Delta quantity of 2 means that quantity is treated as delta quantity. The delta quantity will be added to the existing quantity of the quote it replaces. For example, there is a quote in the orderbook 100@20. A new quote with 30@20 (quantity is 30) and a Delta quantity of 2 will replace the existing quote with (100+30)@20, which becomes a quote of 130@20.

A trader that just wants to change price uses 2 in the Delta Quantity and zero in the Bid Quantity and Ask Quantity.

If the block transaction is sent with less than the maximum number of items allowed, then the size of the transaction must be calculated so it corresponds to the number of items used, instead of the total size of the structure. (The size of the transaction is calculated as (int)& rec.item[rec.items_c] – (int)&rec.)

Total volume cannot be changed from 0 to hidden quantity or from hidden quantity to 0.

3.3.12.5 Return Codes

After a successful MO36 transaction, an order number and the number of entered two-sided quotations, are returned to the sender. The order number is the same for all two-sided quotations in a block. If at least one side (bid/ask) of a two-sided quotation in the block is rejected, the Dedicated Block Transaction Response Broadcast (BO99) is returned and informs of which orders failed and their corresponding error code(s).

> **Note:** If all orders in the block are rejected, the BO99 is not sent.

An MO36 transaction may also be aborted by the Marketplace, in which case only the reason for the transaction being aborted is returned to the sender.

cstatus	txstat		ordidt
Successful	n	Number of two-sided quotations successfully entered and/or matched	Order number
Transaction aborted	n	Error number that is translated by the OMnet routine get_error_message	-

Please refer to *System Error Messages Reference* for details about why transcations are aborted.

3.3.13 MO37 [Two-Sided Price Quotation TRANSACTION]

3.3.13.1 Fingerprint

TRANSACTION properties	
transaction type	MO37
calling sequence	omniapi_tx_ex
struct name	hv_price_2_trans
facility	EP0
partitioned	true

3.3.13.2 Purpose

This transaction is used for placing a two-sided quotation with or without hidden size in the Order book. Previous quote is replaced by the new quote if it exists.

3.3.13.3 Structure

The MO37 TRANSACTION has the following structure:

```
struct hv_price_2_trans  // Named struct no: 34001
```

3.3.13.4 Usage and Conditions

All orders placed in the order book by the MO37 will be removed when using the order number of MO37 in this transaction.

Bid Quantity
Ask Quantity
Bid Volume
Ask Volume

Bid/Ask Quantity display the showsize in the Order book while the Bid/Ask Volume is the actual total size for the quote.

By setting Bid/Ask Volume to zero or equal to Bid/Ask Quantity, the order is sent as a normal order without hidden size.

By setting both the **Quantity** and **Bid**/**Ask Total Volume** to zero, the previous order is deleted and not replaced by a new one.

Order Number, Bid
Order Number, Ask

The order to be replaced in the Order book is specified by the Order Number, Bid and Order Number, Ask. The Central System will only look for the specified order number in the same series as the new order and the order will only be deleted if it exists. No error code is returned if the order does not exist.

Example

The Order Book contains two orders:

Order	Ask Size	Bid Size	Premium	Ask Total Size	Bid Total Size
Order one (from the same partici-pant)	5	-	12	-	-
Order two (from another participant)	-	5	10	-	-

An incoming Order with the same order number as the existing order has the following data:

Order	Ask Size	Bid Size	Premium	Ask Total Size	Bid Total Size
Order three (Order Type 1)	10	-	10	10	-
Order four (Order Type 1)	-	10	8	-	10

These orders will result in a deal of 5@10 and the following Order book:

Order	Ask Size	Bid Size	Premium	Ask Total Size	Bid Total Size
Order five (from the same partici-pant)	5	-	10	-	-
Order six (from the same partici-pant)	-	10	8	-	-

3.3.13.5 Return Codes

An MO37 transaction may also be aborted by the Marketplace, in which case only the reason for the transaction being aborted is returned to the sender.

cstatus	txstat		ordidt
Successful	No Bit set		order number
Successful	Bit 0 set	no part of the Ask order placed in the Order book and no part closed	order number
Successful	Bit 1 set	the whole Ask order closed	order number
Successful	Bit 0 and Bit 1 set	the Ask order partially closed and nothing placed in the Order book	order number
Successful	Bit 2 set	the whole Ask order placed in the Order book	order number

cstatus	txstat		ordidt
Successful	Bit 2 and Bit 1 set	the Ask order partially placed in the Order book and partially closed	order number
Successful	Bit 4 set	Circuit Breaker has started for the Ask order	order number
Successful	Bit 5 set	no part of the Bid order placed in the Order book and no part closed	order number
Successful	Bit 6 set	the whole Bid order closed	order number
Successful	Bit 5 and Bit 6 set	the Bid order partially closed and nothing placed in the Order book	order number
Successful	Bit 7 set	the whole Bid order placed in the Order book	order number
Successful	Bit 6 and Bit 7 set	the Bid order partially placed in the Order book and partially closed	order number
Successful	Bit 9 set	Circuit Breaker has started for the Bid order	order number
Transaction aborted	**GEN_CDC_INT_CLOSED** Instrument Type is not open for this Transaction Type.		-
Transaction aborted	**MP_MATCH_LOW_VOLUME** Fill or Kill order could not be filled because of low Order book size.		-
Transaction aborted	...	...	-

Please refer to *System Error Messages Reference* for details about why transcations are aborted.

3.3.14 MO40 [Inactive Deletion TRANSACTION]

3.3.14.1 Fingerprint

TRANSACTION properties	
transaction type	MO40
calling sequence	omniapi_tx_ex
struct name	delete_trans
facility	EP0
partitioned	true

3.3.14.2 Purpose

The delete inactive transaction is used to remove one or more (by the matching engine) inactivated orders from the Order Book. This transaction can affect several orders at once - a group of orders to be deleted can be specified.

This transaction is similar to MO4 but deletes inactive orders instead.

3.3.14.3 Structure

The MO40 TRANSACTION has the following structure:

```
struct delete_trans  // Named struct no: 34011
```

3.3.14.4 Usage and Conditions

Series

can be completed either as Underlying (**Country Number** plus **Market Code** plus **Commodity Code**) or as **Instrument Class**.

Whose

is used to specify My, Our, My Client’s or Our Client's Order. In this way all combinations of whose order can be obtained, i.e. My or Our Order, and in addition the Combination Client. Fields to be omitted should be filled with NUL characters.

My Orders indicates that I, a broker from Company XX, wish to delete my orders specifically. The expression Our Orders indicates that I remove all Company XX orders regardless of who has placed the order, including orders placed by Exchange’s staff on Company XX’s account.

In addition, it is possible to remove a particular client’s order. In this instance either the client for whom I have placed the order is specified, or the client of Company XX regardless of who placed the order is specified.

Type of order	Fields to be completed
All my orders	Customer and User

Type of order	Fields to be completed
All our orders	Customer
All my orders for a specific client	Customer, User and Client
All our orders for a specific client	Customer and Client

Note:
All character fields must be space padded up to the total length of the field.

Bid or Ask

Order is set to either Bid, Ask or both Bid and Ask.

It is not necessary to complete the whole transaction header as Series can be partially completed.

If one specific order is to be deleted, the whole Series, the order number and whether the order is a Bid or Ask order must be specified.

When a group of orders is to be deleted the group is defined by the following:

- **Series**
- **Whose**
- **Bid or Ask**

Example

- Series is completed with Country Number = 1, Market Code = 1 and Commodity Code = 1.
- Whose is completed with the Customer and User field.
- Bid or Ask is completed with bid.

The result will be that all my bids referring to Swedish Index Call Options are removed from the Order Book.

Example

- Series is completed inclusive Instrument Class with 6 4 3 1001.
- Whose is completed with the Customer and Client field.
- Bid or Ask is set to zero.

The result will be that all Company “Customer’s” bid orders and ask requests for client “client” concerning some currency forwards in the UK (OMLX) will be removed.

3.3.14.5 Return Codes

An MO40 transaction may also be aborted by the Marketplace, in which case only the reason for the transaction being aborted is returned to the sender.

cstatus	txstat	ordidt
Successful	n	-

cstatus	txstat	ordidt
	Number of orders deleted, or zero if no order exists.	
Successful	n Number of contracts before deletion (for specific order number deletion only.)	-
Transaction aborted	GEN_CDC_INT_CLOSED Instrument Type is not open for this Transaction Type.	-
Transaction aborted	...	-

After a successful MO40 transaction, the number of orders deleted, or zero if no order exists, is returned to the sender. Not finding an order to delete is considered a successful operation. For specific order number deletion, number of contracts before deletion, or zero if no order exists, is returned to the sender.

Please refer to *System Error Messages Reference* for details about why transcations are aborted.

3.3.15 MO74 [Trade Report Deletion, Unmatched TRANSACTION]

3.3.15.1 Fingerprint

TRANSACTION properties	
transaction type	MO74
calling sequence	omniapi_tx_ex
struct name	delete_trans
facility	EP0
partitioned	false

3.3.15.2 Purpose

This transaction is used to remove one or more unmatched trade reports from the trade report order book. The transaction can be used for the own participant and also for proxy usage (i.e. Trader ID).

3.3.15.3 Structure

The MO74 TRANSACTION has the following structure:

```
struct delete_trans  // Named struct no: 34011
```

3.3.15.4 Usage and conditions

Series

May contain wildcards.

Order Number

May be blank to indicate wildcard.

Whose, trading code

Must contain the member code of the participant, to which the user submitting the transaction belongs. May also be specified further.

Bid or Ask

May be blank to indicate wildcard.

Example: Assume a user belonging to a certain participant wishes to delete all trade reports submitted by a user within the same participant. To achieve this, the fields **Series**, **Order Number**and**Bid** or**Ask** are left blank in the transaction structure, while the field **Whose**, **Trading Code** is filled with the trading code of the user, for which trade reports are to be deleted.

3.3.15.5 Return Codes

After a successful MO74 transaction, the number of trade reports deleted will be returned to the sender.

An MO74 transaction may also be aborted by the Marketplace, in which case only the reason for the transaction being aborted is returned to the sender.

Please refer to the *System Error Messages Reference* for details about why transcations are aborted.

3.3.16 MO75 [Trade Report TRANSACTION]

3.3.16.1 Fingerprint

TRANSACTION properties	
transaction type	MO75
calling sequence	omniapi_tx_ex
struct name	trade_report_1_trans
facility	EP0
partitioned	true

3.3.16.2 Related Messages

MO76 is the two-sided version.

DQ45

3.3.16.3 Purpose

This transaction is used to send orders that have already led to closings outside the Exchange.

3.3.16.4 Structure

The MO75 TRANSACTION has the following structure:

```
struct trade_report_1_trans  // Named struct no: 34021
```

3.3.16.5 Usage and conditions

The trade report entered in the transaction can only be matched with a trade report entered by the participant specified in the **Counterparty** field.

The following fields are mandatory in a single-sided trade report:

- Transaction Type
- Trade Report Type
- Order Type (has to be a limit order)
- Series
- Bid or Ask (has to be either bid or ask)
- Quantity
- Premium
- Counterparty

Single-sided Trade Reporting

Two participants, A and B, have executed a trade outside the exchange. Each party reports its own side of the trade. The following sequence of events takes place:

1. Participant A submits a one-sided trade report with participant B as the declared counterparty.
2. A directed firm Order Book message (BO5) is sent to participant A. The BO5 shows all details of the unmatched trade report.
3. Participant B submits a one-sided trade report with participant A as the declared counterparty. A trade is created when the two trade reports match.
4. A BO5 message is sent to both parties to indicate that the original trade reports are matched and a trade is created.
5. Trade confirmation broadcasts are disseminated to inform about the trade.

Party

Note:
All character fields must be space padded up to the total length of the field.

3.3.16.6 Return Codes

Cstatus	Txstat	Ordidt
Successful	2 - The whole order closed.	Order number
Successful	4 - The whole order placed in the order book.	Order number
Transaction Aborted	...	-

After a successful MO75 transaction, an order number and information regarding the state of the order will be returned to the sender.

An MO75 transaction may also be aborted by the Marketplace, in which case only the reason for the transaction being aborted is returned to the sender.

Please refer to the *System Error Messages Reference* for details about why transcations are aborted.

3.3.17 MO76 [Trade Report, Two-Sided TRANSACTION]

3.3.17.1 Fingerprint

TRANSACTION properties	
transaction type	MO76
calling sequence	omniapi_tx_ex
struct name	trade_report_2_trans
facility	EP0
partitioned	true

3.3.17.2 Related Messages

MO75 is the single-sided version.

DQ45

3.3.17.3 Purpose

This transaction is used to send orders on behalf of two participants that have already closed a deal outside the Exchange.

3.3.17.4 Structure

The MO76 TRANSACTION has the following structure:

```
struct trade_report_2_trans  // Named struct no: 34022
```

3.3.17.5 Usage and conditions

The following fields are mandatory in a two-sided trade report:

- Transaction Type
- Trade Report Type
- Order Type (has to be a limit order)
- Series
- Bid or Ask (has to be either bid or ask)
- Quantity

- Premium
- Buyer, Counterparty
- Seller, Counterparty

Two-sided Trade Reporting

Two participants, A and B, have executed a trade outside the exchange. Participant A reports both sides of the trade. The following sequence of events takes place:

1. Participant A submits a two-sided trade report with participant B as the declared counterparty. A trade is created.
2. A directed firm Order Book message (BO5) is sent to both parties showing relevant information of the matched trade report. If the trade report is a crossing, same participant on buy and sell side, one BO5 will be disseminated to the reporting participant. All fields of the submitted trade report are available in this BO5.
3. Trade confirmation broadcasts are disseminated to inform about the trade.

3.3.18 MO77 [Combination Trade Report TRANSACTION]

3.3.18.1 Fingerprint

TRANSACTION properties	
transaction type	MO77
calling sequence	omniapi_tx_ex
struct name	combo_trade_report_trans
facility	EP0
partitioned	true

3.3.18.2 Related Messages

DQ45

3.3.18.3 Purpose

This transaction is used by clients to enter combination trade reports containing up to 6 legs.

3.3.18.4 Structure

The MO77 TRANSACTION has the following structure:

```
struct combo_trade_report_trans {
   struct transaction_type
   struct series  // Named struct no: 50000
   UINT8_T ext_t_state_c  // Trade Report Type
   CHAR filler_1_s  // Filler
```

```
    UINT16_T items_n  // Items
    Array ITEM [max no: 6] {
       struct series  // Named struct no: 50000
       INT64_T mp_quantity_i  // Quantity
       INT32_T premium_i  // Premium
       UINT32_T block_n  // Block Size
       char[8] settlement_date_s  // Date, Settlement
       char[8] time_of_agreement_date_s  // Time of agreement, date part
       char[6] time_of_agreement_time_s  // Time of agreement, time part
       UINT8_T deferred_publication_c  // Deferred Publication
       CHAR filler_1_s  // Filler
       struct bid_side  // Of type: TRD_RPT_CUST
       struct ask_side  // Of type: TRD_RPT_CUST
    }
}
```

3.3.18.5 Usage and conditions

The following fields are mandatory in a combination trade report:

- Transaction Type
- Trade Report Type
- Order Type (has to be a limit order)
- Series
- Bid or Ask (has to be either bid or ask)
- Quantity
- Premium
- Buyer, Counterparty
- Seller, Counterparty

Combination Trade Reporting

The participant A has executed a number of trades outside the exchange with one, or several, other participants. Participant A reports both sides of the trades. The following sequence of events takes place:

1. Participant A submits a combination trade report, with the relevant participant as the declared counterparty of each leg. Trades are created.
2. A directed firm Order Book message (BO5) is sent to each relevant party showing relevant information of each side of each leg of the matched combination trade report. The MO77 transaction will be split up into one MO76 transaction for each leg in the BO5s containing the mirrored transaction. Thus an MO77 containing two legs will be sent out as one BO5 with a mirrored MO76 for the first leg, and one BO5 with a mirrored MO76 for the second leg.
3. Trade confirmation broadcasts are disseminated to inform the clearinghouse about the trade.

3.3.19 MO96 [Mass Quote Transaction TRANSACTION]

3.3.19.1 Fingerprint

TRANSACTION properties	
transaction type	MO96
calling sequence	omniapi_tx_ex
struct name	mass_quote_trans
facility	EP0
partitioned	true

3.3.19.2 Purpose

This transaction is provided to support high frequency quoting with low latency, obtained by a double sided transaction, with only basic quote information. The transaction can only be used for trading on own account.

3.3.19.3 Structure

The MO96 TRANSACTION has the following structure:

```
struct mass_quote_trans {
   struct transaction_type
   struct series  // Named struct no: 50000
   char[2] filler_2_s  // Filler
   UINT16_T items_n  // Items
   Array ITEM [max no: 37] {
      struct series  // Named struct no: 50000
      UINT32_T bid_price_i  // Bid Price
      INT64_T bid_quantity_i  // Quantity, Bid
      UINT32_T ask_price_i  // Ask Price
      INT64_T ask_quantity_i  // Quantity, Ask
   }
}
```

3.3.19.4 Usage and Conditions

A new quote always replaces a previous quote, per order book and participant. Thus, a market maker is only allowed to have one quote per order book.

Bid and ask prices in an incoming quote are not allowed to cross or lock with each other. Should they cross or lock, the quote is rejected.

An update of only one side can be made by specifying zero in the quantity of the other side. This is similar to the order update transactions in which zero in a field indicates "no change." In this case the side that is not updated will keep its priority. If an update made to one side makes the price of that side cross or lock with the side on the book, the quote on the book is removed in order to avoid a case where you would trade with your own quote. In case zero is put in the quantity field, the price field is disregarded, i.e. it is not

possible to have "no change" of the quantity and still update the price. If a new price is to be quoted, the quantity must be specified.

Quotes are deleted by specifying minus 1 (-1) in the quantity field. If both sides are to be deleted, both bid and ask quantity should be set to -1. In case -1 is set in the quantity field, the price field is disregarded.

> **Note:** The MO96 transaction does not handle combinations.

3.3.19.5 Return Codes

After a successful MO96 transaction, the number of successful quotes will be returned to the sender.

An MO96 transaction may also be aborted by the Marketplace, in which case only the reason for the transaction being aborted is returned to the sender.

Please refer to *System Error Messages Reference* for details about why transcations are aborted.

3.3.20 MO99 [Activate Central Inactive Order TRANSACTION]

3.3.20.1 Fingerprint

TRANSACTION properties	
transaction type	MO99
calling sequence	omniapi_tx_ex
struct name	hv_alter_trans
facility	EP0
partitioned	false

3.3.20.2 Related Messages

MO483 is the proxy variant.

3.3.20.3 Purpose

This transaction may be used for activating centrally inactive orders.

3.3.20.4 Structure

The MO99 TRANSACTION has the following structure:

```
struct hv_alter_trans  // Named struct no: 34010
```

3.3.20.5 Usage and conditions

An activation will be subject to the same validations as in MO31 (without exceptions).

The activation transaction shall be sent with NULL in all fields except series, order number, and side. Thus, a non-zero value implies a change, which will trigger the error message MP_MATCH_INV_ACTIVATION "Illegal central activate order transaction".

Order Number

specifies the order that is to be activated. The activated order will keep this number.

When an API client needs to activate an existing centrally inactive order it has to send a Centrally Inactive Order Activation MO99.

3.3.21 MO388 [Proxy delete order TRANSACTION]

3.3.21.1 Fingerprint

TRANSACTION properties	
transaction type	MO388
calling sequence	omniapi_tx_ex
struct name	delete_trans_p
facility	EP0
partitioned	true

3.3.21.2 Related Messages

This is a proxy transaction for MO4.

3.3.21.3 Purpose

This is a Trader ID transaction, which is used when a trader, user or application wants to send a transaction on behalf of someone else.

3.3.21.4 Structure

The MO388 TRANSACTION has the following structure:

```
struct delete_trans_p  // Named struct no: 34111
```

3.3.21.5 Usage and Conditions

The thing that differentiates MO388 from MO4 is an extra sub-struct called trading_code, which must be filled with the trading code of the participant or user the on-behalf transaction is sent for. The whose field also contains a trading_code.This field is used in the same way as the whose field for the MO4 transaction.

3.3.21.6 Return Codes

The return codes associated with the transaction are the same as for the base transaction, MO4.

Please refer to *System Error Messages Reference* for details about why transcations are aborted.

3.3.22 MO415 [MO31 With Trader ID TRANSACTION]

3.3.22.1 Fingerprint

TRANSACTION properties	
transaction type	MO415
calling sequence	omniapi_tx_ex
struct name	hv_order_trans_p
facility	EP0
partitioned	true

3.3.22.2 Related Messages

This is a proxy transaction for MO31.

3.3.22.3 Purpose

This is a Trader ID transaction, which is used when a trader, user or application wants to send a transaction on behalf of someone else.

3.3.22.4 Structure

The MO415 TRANSACTION has the following structure:

```
struct hv_order_trans_p  // Named struct no: 34105
```

3.3.22.5 Usage and Conditions

The thing that differentiates MO415 from MO31 is an extra sub-struct called trading_code, which must be filled with the trading code of the user the order originates from.

3.3.22.5.1 Open Close Request

If the highest bit in the requested position field (open_close_req_c) is set, the open_close_req_c field will be interpreted as a bit map instead of an enum.

Bit 0 represents leg 1 in the combination, bit 1 legs 2, and so on.

It will only be possible to specify “open” or “close” for each leg. “0” for “open” and “1” for “close.”

Example

Let us assume a scenario with Open_close_req_c (8 bit):

Bit 7					...	Bit 1	Bit 0
1	0	0	0	0	1	0	1

If the highest bit is set, the field will be interpreted as a bitmap.

In the example above, the 1st and 3rd leg in the combination shall be close (1) and the rest of the legs shall be open (0).

Note:

The highest bit is only allowed for orders that are entered into combo series. If an order is entered in a series that is not a combo series, the normal **open_close_req_c** values must be specified.

3.3.22.6 Return Codes

The return codes associated with the transaction are the same as for the base transaction, MO31.

Please refer to *System Error Messages Reference* for details about why transcations are aborted.

3.3.23 MO417 [MO33 With Trader ID TRANSACTION]

3.3.23.1 Fingerprint

TRANSACTION properties	
transaction type	MO417
calling sequence	omniapi_tx_ex
struct name	hv_alter_trans_p
facility	EP0
partitioned	true

3.3.23.2 Related Messages

This is a proxy transaction for MO33.

3.3.23.3 Purpose

This is a Trader ID transaction, which is used when a trader, user or application wants to send a transaction on behalf of someone else.

3.3.23.4 Structure

The MO417 TRANSACTION has the following structure:

```
struct hv_alter_trans_p  // Named struct no: 34110
```

3.3.23.5 Usage and Conditions

The thing that differentiates MO417 from MO33 is an extra sub-struct called trading_code, which must be filled with the trading code of the user the order originates from.

3.3.23.6 Return Codes

The return codes associated with the transaction are the same as for the base transaction, MO33.

Please refer to *System Error Messages Reference* for details about why transcations are aborted.

3.3.24 MO420 [MO36 With Trader ID TRANSACTION]

3.3.24.1 Fingerprint

TRANSACTION properties	
transaction type	MO420
calling sequence	omniapi_tx_ex
struct name	block_price_trans_p
facility	EP0
partitioned	true

3.3.24.2 Related Messages

This is a proxy transaction for MO36.

3.3.24.3 Purpose

This is a Trader ID transaction, which is used when a trader, user, or application wants to send a transaction on behalf of someone else.

3.3.24.4 Structure

The MO420 TRANSACTION has the following structure:

```
struct block_price_trans_p  // Named struct no: 34107
```

3.3.24.5 Usage and Conditions

The only thing that differentiates MO420 from MO36 is an extra sub-struct called trading_code, which must be filled with the trading code of the user the order originates from.

3.3.24.6 Return Codes

The return codes associated with the transaction are the same as for the base transaction, MO36.

Please refer to *System Error Messages Reference* for details about why transcations are aborted.

3.3.25 MO421 [MO37 With Trader ID TRANSACTION]

3.3.25.1 Fingerprint

TRANSACTION properties	
transaction type	MO421
calling sequence	omniapi_tx_ex
struct name	hv_price_2_trans_p
facility	EP0
partitioned	true

3.3.25.2 Related Messages

This is a proxy transaction for MO37.

3.3.25.3 Purpose

This is a Trader ID transaction, which is used when a trader, user, or application wants to send a transaction on behalf of someone else.

3.3.25.4 Structure

The MO421 TRANSACTION has the following structure:

```
struct hv_price_2_trans_p  // Named struct no: 34101
```

3.3.25.5 Usage and Conditions

The only thing that differentiates MO421 from MO37 is an extra sub-struct called trading_code, which must be filled with the trading code of the user the order originates from.

3.3.25.6 Return Codes

The return codes associated with the transaction are the same as for the base transaction, MO37.

Please refer to *System Error Messages Reference* for details about why transcations are aborted.

3.3.26 MO424 [Proxy Delete inactive order TRANSACTION]

3.3.26.1 Fingerprint

TRANSACTION properties	
transaction type	MO424
calling sequence	omniapi_tx_ex

TRANSACTION properties	
struct name	delete_trans_p
facility	EP0
partitioned	true

3.3.26.2 Related Messages

This is a proxy transaction for MO40.

3.3.26.3 Purpose

This is a proxy version of MO40. The only thing that differentiates MO424 from MO40 is an extra sub-struct called trading_code, which must be filled with the trading code of the user the order originates from.

3.3.26.4 Structure

The MO424 TRANSACTION has the following structure:

```
struct delete_trans_p  // Named struct no: 34111
```

3.3.26.5 Usage and Conditions

This is a Trader ID transaction, which is used when a trader, user, or application wants to send a transaction on behalf of someone else.

3.3.26.6 Return Codes

The return codes associated with the transaction are the same as for the base transaction, MO40.

Please refer to *System Error Messages Reference* for details about why transcations are aborted.

3.3.27 MO459 [Trade Report, Proxy TRANSACTION]

3.3.27.1 Fingerprint

TRANSACTION properties	
transaction type	MO459
calling sequence	omniapi_tx_ex
struct name	trade_report_1_trans_p
facility	EP0
partitioned	true

3.3.27.2 Related Messages

- This is a proxy transaction for MO75.

DQ45

3.3.27.3 Purpose

This transaction is used to send orders that have led to closings outside the Exchange.

This is a proxy version of MO75. The only thing that differentiates MO459 from MO75 is an extra sub-struct called trading_code, which must be filled with the trading code of the user the order originates from.

3.3.27.4 Structure

The MO459 TRANSACTION has the following structure:

```
struct trade_report_1_trans_p  // Named struct no: 34119
```

3.3.28 MO483 [Proxy Activate Central Inactive Order TRANSACTION]

3.3.28.1 Fingerprint

TRANSACTION properties	
transaction type	MO483
calling sequence	omniapi_tx_ex
struct name	hv_alter_trans_p
facility	EP0
partitioned	false

3.3.28.2 Related Messages

MO99 is the external variant.

MO355 is the internal variant.

MO482, Enter Central Inactive Order, proxy.

3.3.28.3 Purpose

This transaction may be used for activating all central inactive orders.

3.3.28.4 Structure

The MO483 TRANSACTION has the following structure:

```
struct hv_alter_trans_p  // Named struct no: 34110
```

3.3.28.5 Usage and conditions

Note that all central inactive orders will affected, that is, not only the central inactive orders entered via the API but also orders that have been inactivated (and centrally stored) by the central system.An activation will be subject to the same validations as in MO31 (without exceptions).

All fields must be identical to the fields in the order that is to be activated. The error message MP_MATCH_INV_ACTIVATION "Illegal central activate order transaction" will be displayed if change of order is attempted.

Order Number

specifies the order that is to be activated. The activated order will keep this number.

Currency Format

does not apply to all exchanges.

When an API client needs to activate an exisiting centrally inactive order it has to send a Centrally Inactive Order Activation MO483.

3.3.29 MQ5 [Proxy Order QUERY]

3.3.29.1 Fingerprint

QUERY properties	
transaction type	MQ5
calling sequence	omniapi_query_ex
struct name	query_tot_order
facility	EP0
partitioned	true
answers	MA8

ANSWER properties	
transaction type	MA8
struct name	answer_tot_order
segmented	true

3.3.29.2 Purpose

This transaction is used for querying orders entered on behalf of someone else (with MOX+384 transactions).

3.3.29.3 Structure

The MQ5 QUERY has the following structure:

```
struct query_tot_order {
   struct transaction_type
   struct series  // Named struct no: 50000
   struct whose
   UINT32_T order_index_u  // Order Index
}
```

3.3.29.4 Usage and Conditions

Whose, trading code

must contain the member code of the participant, to which the querying user belongs. May also be specified further.

> **Note:**
> All character fields must be space padded up to the total length of the field.

3.3.29.5 Return Codes

An MQ5 transaction may also be aborted by the Marketplace, in which case only the reason for the transaction being aborted is returned to the sender.

cstatus	txstat	ordidt	rvcbuf
Successful	Normal	transaction identification	list of proxy orders – see Answer, structure
Transaction aborted	GEN_CDC_INT_CLOSED Instrument type is not open for this transaction type.	-	-
Transaction aborted	MP_QUERY_CUST_UND Underlying or Customer is not fully defined in query	-	-
Transaction aborted	…	-	-

Please refer to *System Error Messages Reference* for details about why transcations are aborted.

3.3.29.6 Answer Structure

The MA8 ANSWER has the following structure:

```
struct answer_tot_order {
   struct transaction_type
   struct series  // Named struct no: 50000
   UINT32_T order_index_u  // Order Index
   UINT16_T items_n  // Items
   char[2] filler_2_s  // Filler
   Array ITEM [max no: 300] {
      QUAD_WORD order_number_u  // Order Number
      UINT32_T sequence_number_u  // Sequence Number
```

```
        UINT32_T ob_position_u  // Order Book Position
        UINT8_T combo_mark_c  // Combination Order Mark
        UINT8_T order_category_c  // Order Category
        char[2] filler_2_s  // Filler
        struct party
        struct order
        INT64_T total_volume_i  // Total Volume
        INT64_T display_quantity_i  // Quantity, Display
        INT64_T orig_shown_quantity_i  // Shown Quantity, Original
        INT64_T orig_total_volume_i  // Total Volume, Original
        struct timestamp_in  // Of type: TIME_SPEC
        struct timestamp_created  // Of type: TIME_SPEC
    }
}
```

3.3.29.7 Answer, comments

After a successful MQ5 transaction, a list of own proxy orders in the order book is returned to the sender. The Series and Order Index must be zero-filled to get the start segment of the partition. To get the next segments and partition, the Series and Order Index in the previous answer should be used. If the Series in the answer is zero-filled the end of the last partition is reached.

3.3.30 MQ7 [Total Order Book QUERY]

3.3.30.1 Fingerprint

QUERY properties	
transaction type	MQ7
calling sequence	omniapi_query_ex
struct name	query_tot_ob
facility	EP0
partitioned	true
answers	MA42

ANSWER properties	
transaction type	MA42
struct name	answer_tot_ob
segmented	true

3.3.30.2 Purpose

This transaction is used for querying all orders in the Order Book.

3.3.30.3 Structure

The MQ7 QUERY has the following structure:

```
struct query_tot_ob {
   struct transaction type
   struct series  // Named struct no: 50000
   QUAD WORD order number u  // Order Number
   UINT8_T bid_or_ask_c  // Bid or Ask
   UINT8_T only_this_series_c  // Series, Only this
   char[2] filler_2_s  // Filler
}
```

3.3.30.4 Usage and Conditions

After a successful MQ7 transaction, a list of orders in the Order Book is returned to the sender. The Series, Order number and Bid or Ask must be zero-filled to get the start segment of the partition. To get the next segments and partition, the series, order number and bid or ask in the previous answer should be used.

If the search is made on all series, that is, if the Only this series field is zero, the last order in the last partition has been received when the series is zero-filled in an answer. If the search is made on a single series, that is, if the Only this series has a non-zero value, the last order has been received when the series is zero-filled in an answer. The Order number and Bid or Ask must be zero-filled to get the start segment.

3.3.30.5 Return Codes

An MQ7 transaction may also be aborted by the Marketplace, in which case only the reason for the transaction being aborted is returned to the sender.

cstatus	txstat	ordidt	rcvbuf
Successful	Normal	transaction identification	list of orders - see Answer, structure (Answer with Identity)
Transaction aborted	**GEN_CDC_INT_CLOSED** Instrument Type is not open for this Transaction Type.	-	-
Transaction aborted	**MP_QUERY_CUST_UND** Underlying or Customer is not fully defined in query.	-	-
Transaction aborted	…	-	-

Please refer to *System Error Messages Reference* for details about why transcations are aborted.

The MA42 ANSWER has the following structure:

```
struct answer_tot_ob {
   struct transaction_type
   struct series  // Named struct no: 50000
   QUAD_WORD order_number_u  // Order Number
   UINT16_T items_n  // Items
   UINT8_T bid_or_ask_c  // Bid or Ask
   CHAR filler_1_s  // Filler
   Array ITEM [max no: 1000] {
      QUAD_WORD order_number_u  // Order Number
      UINT32_T sequence_number_u  // Sequence Number
      UINT32_T ob_position_u  // Order Book Position
      UINT8_T combo_mark_c  // Combination Order Mark
      char[3] filler_3_s  // Filler
      struct order_no_id
      struct party
   }
}
```

3.3.30.6 Answer, comments

If the trader identity is not public information, party is blanked.

3.3.31 MQ8 [Total Order QUERY]

3.3.31.1 Fingerprint

QUERY properties	
transaction type	MQ8
calling sequence	omniapi_query_ex
struct name	query_tot_order
facility	EP0
partitioned	true
answers	MA43

ANSWER properties	
transaction type	MA43
struct name	answer_tot_order
segmented	true

3.3.31.2 Purpose

This transaction is used for querying own orders in the Order Book or for another user in the same firm or for all orders for a firm.

3.3.31.3 Structure

The MQ8 QUERY has the following structure:

```
struct query_tot_order {
   struct transaction type
   struct series  // Named struct no: 50000
   struct whose
   UINT32_T order_index_u  // Order Index
}
```

3.3.31.4 Usage and Conditions

Whose, trading code

must contain the member code of the participant, to which the querying user belongs. May also be specified further.

> **Note:**
> All character fields must be space padded up to the total length of the field.

Synchronization of BO5 and MQ8

The following steps must be done to synchronize BO5 and MQ8:

- Start subscribing to BO5.
- Keep the received BO5s and do not process them until MQ8 query is done.
- Send MQ8 and insert all records to the firm order book.
- Process the queued BO5s. They must be processed in the same order as received. For each BO5, look up the order in the firm order book and use business logic to determine operation. E.g. if change_reason_c = 9 (system delete) and the order is not present in the firm order book, discard this BO5.
- Continue to process received BO5 broadcasts.

3.3.31.5 Return Codes

An MQ8 transaction may also be aborted by the Marketplace, in which case only the reason for the transaction being aborted is returned to the sender.

cstatus	txstat	ordidt	rcvbuf
Successful	Normal	transaction identification	list of orders - see Answer, structure
Transaction aborted	GEN_CDC_INT_CLOSED Instrument Type is not open for this Transaction Type	-	-
Transaction aborted	MP_QUERY_CUST_UND	-	-

cstatus	txstat	ordidt	rcvbuf
	Underlying or Customer is not fully defined in query.		
Transaction aborted	…	-	-

Please refer to *System Error Messages Reference* for details about why transcations are aborted.

3.3.31.6 Answer Structure

The MA43 ANSWER has the following structure:

```
struct answer_tot_order {
   struct transaction_type
   struct series  // Named struct no: 50000
   UINT32_T order_index_u  // Order Index
   UINT16_T items_n  // Items
   char[2] filler_2_s  // Filler
   Array ITEM [max no: 300] {
      QUAD_WORD order_number_u  // Order Number
      UINT32_T sequence_number_u  // Sequence Number
      UINT32_T ob_position_u  // Order Book Position
      UINT8_T combo_mark_c  // Combination Order Mark
      UINT8_T order_category_c  // Order Category
      char[2] filler_2_s  // Filler
      struct party
      struct order
      INT64_T total_volume_i  // Total Volume
      INT64_T display_quantity_i  // Quantity, Display
      INT64_T orig_shown_quantity_i  // Shown Quantity, Original
      INT64_T orig_total_volume_i  // Total Volume, Original
      struct timestamp_in  // Of type: TIME_SPEC
      struct timestamp_created  // Of type: TIME_SPEC
   }
}
```

3.3.31.7 Answer, comments

Sequence Number

is a non-consecutive increasing number per series. It can be used to synchronize the answer to the MQ8 query with the corresponding broadcast flow.

Quantity

indicates how many contracts are shown in the order book.

Volume

indicates the total number of remaining contracts.

If Volume is set to zero, the order is a normal order without hidden size. In that case **Display Quantity** is zero too.

Display Quantity

indicates the limit for the new contracts that will be displayed in the order book, for a hidden order, after the previous have been traded.

A Successful MQ8 Transaction

After a successful MQ8 transaction, a list of own orders in the order book is returned to the sender. The Series and Order Index must be zero-filled to get the start segment of the partition. To get the next segments and partition, the Series and Order Index in the previous answer should be used. If the Series in the answer is zero-filled the end of the last partition is reached.

3.3.32 MQ9 [Total Inactive Order QUERY]

3.3.32.1 Fingerprint

QUERY properties	
transaction type	MQ9
calling sequence	omniapi_query_ex
struct name	query_tot_order
facility	EP0
partitioned	true
answers	MA44

ANSWER properties	
transaction type	MA44
struct name	answer_tot_order
segmented	true

3.3.32.2 Purpose

This transaction is used for querying own inactive orders in the Order Book.

3.3.32.3 Structure

The MQ9 QUERY has the following structure:

```
struct query_tot_order {
   struct transaction_type
   struct series  // Named struct no: 50000
   struct whose
   UINT32_T order_index_u  // Order Index
}
```

3.3.32.4 Usage and Conditions

Whose, trading code

must contain the member code of the participant, to which the querying user belongs. May also be specified further.

> **Note:**
> All character fields must be space padded up to the total length of the field.

3.3.32.5 Return Codes

An MQ9 transaction may also be aborted by the Marketplace, in which case only the reason for the transaction being aborted is returned to the sender.

cstatus	txstat	ordidt	rcvbuf
Successful	Normal	transaction identification	list of orders - see Answer, structure
Transaction aborted	GEN_CDC_INT_CLOSED Instrument Type is not open for this Transaction Type	-	-
Transaction aborted	MP_QUERY_CUST_UND Underlying or Customer is not fully defined in query.	-	-
Transaction aborted	…	-	-

Please refer to *System Error Messages Reference* for details about why transcations are aborted.

3.3.32.6 Answer Structure

The MA44 ANSWER has the following structure:

```
struct answer_tot_order {
   struct transaction_type
   struct series  // Named struct no: 50000
   UINT32_T order_index_u  // Order Index
   UINT16_T items_n  // Items
   char[2] filler_2_s  // Filler
   Array ITEM [max no: 300] {
      QUAD_WORD order_number_u  // Order Number
      UINT32_T sequence_number_u  // Sequence Number
      UINT32_T ob_position_u  // Order Book Position
      UINT8_T combo_mark_c  // Combination Order Mark
      UINT8_T order_category_c  // Order Category
      char[2] filler_2_s  // Filler
      struct party
      struct order
      INT64_T total_volume_i  // Total Volume
```

```
        INT64_T display_quantity_i  // Quantity, Display
        INT64_T orig_shown_quantity_i  // Shown Quantity, Original
        INT64_T orig_total_volume_i  // Total Volume, Original
        struct timestamp_in  // Of type: TIME_SPEC
        struct timestamp_created  // Of type: TIME_SPEC
    }
}
```

3.3.32.7 Answer, comments

Quantity

indicates how many contracts are shown in the order book.

Volume

indicates the total number of remaining contracts.

If **Volume** is set to zero, the order is a normal order without hidden size. In that case **Display Quantity** is zero too.

Display Quantity

indicates the limit for the new contracts that will be displayed in the order book, for a hidden order, after the previous have been traded.

After a successful MQ9 transaction, a list of own inactive orders in the order book is returned to the sender. The Series and Order Index must be zero-filled to get the start segment of the partition. To get the next segments and partition, the Series and Order Index in the previous answer should be used. If the Series in the answer is zero-filled the end of the last partition is reached.

3.3.33 MQ78 [Query Trade Reports, Unmatched QUERY]

3.3.33.1 Fingerprint

QUERY properties	
transaction type	MQ78
calling sequence	omniapi_query_ex
struct name	query_tot_order
facility	EP0
partitioned	true
answers	MA78

ANSWER properties	
transaction type	MA78
struct name	answer_trd_report
segmented	true

3.3.33.2 Purpose

This query is used to query for unmatched trade reports for a specific participant or user at the specific participant. The query can be used for the own participant and also for proxy usage (i.e. Trader ID).

3.3.33.3 Structure

The MQ78 QUERY has the following structure:

```
struct query_tot_order {
   struct transaction_type
   struct series  // Named struct no: 50000
   struct whose
   UINT32_T order_index_u  // Order Index
}
```

3.3.33.4 Usage and conditions

Series

May contain wildcards.

Client

Character "*" and "%" are **not** allowed in the Client field.

Whose, trading code

Must contain the member code of the participant, to which the querying user belongs. May also be specified further.

Order Index

If non-blank it indicates the first trade report to be included in the answer, counted as offset from the first trade report in the trade report order book for the participant in question.

> *Example*
>
> Assume a user wishes to query for all trade reports submitted by a user within the participant, to which the user submitting the query belongs. To achieve this, the fields **Order Index** and **Series** are left blank in the query structure, while the field **Whose**, **Trading Code** is filled with the trading code of the user in question.

3.3.33.5 Answer Structure

The MA78 ANSWER has the following structure:

```
struct answer_trd_report {
   struct transaction_type
   struct series  // Named struct no: 50000
   UINT32_T order_index_u  // Order Index
   UINT16_T items_n  // Items
```

```
    char[2] filler_2_s  // Filler
    Array ITEM [max no: 300] {
       struct trading_code
       struct transaction_type
       QUAD_WORD order_number_u  // Order Number
       struct series  // Named struct no: 50000
       struct order_var
       struct party
       UINT32_T sequence_number_u  // Sequence Number
       CHAR[32] exchange_info_s  // Exchange, Information
       struct give_up_member  // Named struct no: 50002
       char[8] settlement_date_s  // Date, Settlement
       char[8] time_of_agreement_date_s  // Time of agreement, date part
       char[6] time_of_agreement_time_s  // Time of agreement, time part
       UINT8_T deferred_publication_c  // Deferred Publication
       CHAR filler_1_s  // Filler
    }
}
```

3.3.33.6 Answer, comments

After as successful MQ78 transaction, a number of answer items are returned to the sender. If the number of answer items to be returned to the sender exceeds the number that can be contained in a single answer structure, the field **Order Index**will indicate the trade report, for which the query for the second segment should be submitted.

3.3.34 MQ80 [Query Trade Reports Counterpart, Unmatched QUERY]

3.3.34.1 Fingerprint

QUERY properties	
transaction type	MQ80
calling sequence	omniapi_query_ex
struct name	query_tot_party
facility	EP0
partitioned	true
answers	MA80

ANSWER properties	
transaction type	MA80
struct name	answer_trd_report_party
segmented	true

3.3.34.2 Purpose

This query is used to retrieve all unmatched trade reports where the participant has been appointed as a counterparty.

3.3.34.3 Structure

The MQ80 QUERY has the following structure:

```
struct query_tot_party {
   struct transaction type
   struct series  // Named struct no: 50000
   QUAD_WORD order_number_u  // Order Number
   UINT8_T bid_or_ask_c  // Bid or Ask
   char[3] filler_3_s  // Filler
}
```

3.3.34.4 Usage and conditions

Order Number

May be blank to indicate wildcard.

Series

May contain wildcards if order number is blank

Bid or Ask

May be blank to indicate wildcard if order number is blank

> *Example*
>
> Assume a user wishes to query for all unmatched trade reports for which the participant of the user submitting the query has been specified as the counterpart. To achieve this, the fields **Order Number** , **Series** and **Bid** or **Ask** are all left blank in the query structure.

3.3.34.5 Answer Structure

The MA80 ANSWER has the following structure:

```
struct answer_trd_report_party {
   struct transaction type
   struct series  // Named struct no: 50000
   QUAD_WORD order_number_u  // Order Number
   UINT16_T items_n  // Items
   UINT8_T bid_or_ask_c  // Bid or Ask
   CHAR filler_1_s  // Filler
   Array ITEM [max no: 300] {
      struct trading code
      struct transaction type
      QUAD_WORD order_number_u  // Order Number
```

```
        struct series  // Named struct no: 50000
        struct order_var
        struct party
        CHAR[32] exchange_info_s  // Exchange, Information
        struct give_up_member  // Named struct no: 50002
        char[8] settlement_date_s  // Date, Settlement
        char[8] time_of_agreement_date_s  // Time of agreement, date part
        char[6] time_of_agreement_time_s  // Time of agreement, time part
        UINT8_T deferred_publication_c  // Deferred Publication
        CHAR filler_1_s  // Filler
    }
}
```

3.3.34.6 Answer, comments

After as successful MQ80 transaction, a number of answer items are returned to the sender. If the number of answer items to be returned to the sender exceeds the number that can be contained in a single answer structure, the fields **Order Number**, **Series** and **Bid** or **Ask** will indicate the trade report, for which the query for the second segment should be submitted.

3.3.35 MQ99 [Maximum Block Order Sizes QUERY]

3.3.35.1 Fingerprint

QUERY properties	
transaction type	MQ99
calling sequence	omniapi_query_ex
struct name	query_block_size
facility	EP0
partitioned	true
answers	MA99

ANSWER properties	
transaction type	MA99
struct name	answer_block_size
segmented	false

3.3.35.2 Purpose

MQ99 provides the max exchange allowed limit for MO30/MO414 and MO36/MO420.

3.3.35.3 Structure

The MQ99 QUERY has the following structure:

```
struct query_block_size {
   struct transaction_type
   struct series  // Named struct no: 50000
}
```

3.3.35.4 Return Codes

An MQ99 transaction may also be aborted by the Marketplace, in which case only the reason for the transaction being aborted is returned to the sender.

cstatus	txstat	ordidt	rcvbuf
Successful	Normal	transaction identification	Max Block Order Size– see Answer, structure
Transaction aborted	**GEN_CDC_INT_CLOSED** Instrument Type is not open for this Transaction Type.	-	-
Transaction aborted	...	-	-

Please refer to *System Error Messages Reference* for details about why transcations are aborted.

3.3.35.5 Answer Structure

The MA99 ANSWER has the following structure:

```
struct answer_block_size {
   struct transaction_type
   INT32_T max_block_order_size_i  // Order Size, Max Block
   INT32_T max_block_price_size_i  // Order Price, Max Block
}
```

3.3.35.6 Answer, comments

Order Size, Max Block

maximum number of items in a block order transaction.

Order Price, Max Block

maximum number of items in a block quotation transaction.

3.3.36 MQ392 [MQ8 With Trader ID QUERY]

3.3.36.1 Fingerprint

QUERY properties	
transaction type	MQ392
calling sequence	omniapi_query_ex
struct name	query_tot_order_p
facility	EP0
partitioned	true
answers	MA43

ANSWER properties	
transaction type	MA43
struct name	answer_tot_order
segmented	true

3.3.36.2 Purpose

This is a Trader ID transaction, which is used when a trader, user, or application wants to send a transaction on behalf of someone else.

3.3.36.3 Structure

The MQ392 QUERY has the following structure:

```
struct query_tot_order_p {
   struct transaction_type
   struct series  // Named struct no: 50000
   struct whose
   UINT32_T order_index_u  // Order Index
}
```

3.3.36.4 Usage and Conditions

Whose, trading code

must contain the member code of the participant whose order information the querying user wants to retrieve. May also be specified further.

The way in which MQ392 differs from MQ8 is how the whose field is filled out.

> **Note:**
> All character fields must be space padded up to the total length of the field.

3.3.36.5 Answer Structure

The MA43 ANSWER has the following structure:

```
struct answer_tot_order {
   struct transaction_type
   struct series  // Named struct no: 50000
   UINT32_T order_index_u  // Order Index
   UINT16_T items_n  // Items
   char[2] filler_2_s  // Filler
   Array ITEM [max no: 300] {
      QUAD_WORD order_number_u  // Order Number
      UINT32_T sequence_number_u  // Sequence Number
      UINT32_T ob_position_u  // Order Book Position
      UINT8_T combo_mark_c  // Combination Order Mark
      UINT8_T order_category_c  // Order Category
      char[2] filler_2_s  // Filler
      struct party
      struct order
      INT64_T total_volume_i  // Total Volume
      INT64_T display_quantity_i  // Quantity, Display
      INT64_T orig_shown_quantity_i  // Shown Quantity, Original
      INT64_T orig_total_volume_i  // Total Volume, Original
      struct timestamp_in  // Of type: TIME_SPEC
      struct timestamp_created  // Of type: TIME_SPEC
   }
}
```

3.3.36.6 Answer, comments

The answer from the query is the same as for the base transaction, MQ8.

3.3.37 MQ393 [MQ9 With Trader ID QUERY]

3.3.37.1 Fingerprint

QUERY properties	
transaction type	MQ393
calling sequence	omniapi_query_ex
struct name	query_tot_order_p
facility	EP0
partitioned	true
answers	MA44

ANSWER properties	
transaction type	MA44
struct name	answer_tot_order
segmented	true

3.3.37.2 Purpose

This is a Trader ID transaction, which is used when a trader, user, or application wants to send a transaction on behalf of someone else.

3.3.37.3 Structure

The MQ393 QUERY has the following structure:

```
struct query_tot_order_p {
   struct transaction_type
   struct series  // Named struct no: 50000
   struct whose
   UINT32_T order_index_u  // Order Index
}
```

3.3.37.4 Usage and conditions

Whose, trading code

must contain the member code of the participant whose order information the querying user wants to retrieve. May also be specified further.

The way in which MQ393 differs from MQ9 is how the whose field is filled out.

> **Note:**
> All character fields must be space padded up to the total length of the field.

3.3.37.5 Answer Structure

The MA44 ANSWER has the following structure:

```
struct answer_tot_order {
   struct transaction_type
   struct series  // Named struct no: 50000
   UINT32_T order_index_u  // Order Index
   UINT16_T items_n  // Items
   char[2] filler_2_s  // Filler
   Array ITEM [max no: 300] {
      QUAD_WORD order_number_u  // Order Number
      UINT32_T sequence_number_u  // Sequence Number
      UINT32_T ob_position_u  // Order Book Position
      UINT8_T combo_mark_c  // Combination Order Mark
```

```
        UINT8_T order_category_c  // Order Category
        char[2] filler_2_s  // Filler
        struct party
        struct order
        INT64_T total_volume_i  // Total Volume
        INT64_T display_quantity_i  // Quantity, Display
        INT64_T orig_shown_quantity_i  // Shown Quantity, Original
        INT64_T orig_total_volume_i  // Total Volume, Original
        struct timestamp_in  // Of type: TIME_SPEC
        struct timestamp_created  // Of type: TIME_SPEC
    }
}
```

3.3.37.6 Answer, comments

The answer from the query is the same as for the base transaction, MQ9.

3.4 Trade and Position Management

3.4.1 BD6 [Dedicated Trade Information VIB]

3.4.1.1 Fingerprint

VIB properties	
transaction type	BD6
calling sequence	omniapi_read_event_ext_ex or omniapi_read_event_block
struct name	The message complies with the VIM concept and has no top struct. The sequence of possible structs is described in the Structure section.
info type	dedicated

3.4.1.2 Related Messages

CQ10

CQ11

3.4.1.3 Purpose

This is a dedicated trade broadcast distributed to the participants in real-time. The contents of the broadcast is exchange specific.

3.4.1.4 Structure

The BD6 VIB has the following structure:

```
    struct broadcast_hdr
```

```
Sequence {
   struct sub_item_hdr
   Choice {
      struct cl_trade_base_api  // Named struct no: 3
      struct cl_trade_secur_part  // Named struct no: 20
   }
}
```

3.4.1.5 Usage and Conditions

This is a variable broadcast.

The first structure after the header part is always cl_trade_base_api. In addition to that, none or several structures can follow; each preceded by a header.

On systems using BD6 the queries CQ10 and CQ11 are used in conjunction to recover trades.

When retrieving trades disseminated with BD6, the actual data structure is a sequence starting with:

- cl_trade_base_api (named struct no = 3)

3.4.1.6 Structure Contents

Exchange Info

is equivalent to the Passthrough Information field in cl_trade_api.

Date, As of and Time, As of

fields contain information about when the deal was closed or the original trade was registered (in case of rectify or overtaking trade). It is the same data as Time Stamp, last change, but in “business time” format.

Time Stamp, last change

contains date and time the deal was closed, propagated from the MP subsystem (VMS format).

Sequence Number

is assigned each broadcast to allow for a recipient to verify that no trade broadcasts are lost and to indicate the order in which they were sent. The sequence number is unique per participant and instrument type, meaning that the same trade has different sequence numbers for different recipients.

3.4.2 BD18 [Dedicated Delivery BROADCAST]

3.4.2.1 Fingerprint

BROADCAST properties	
transaction type	BD18
calling sequence	omniapi_read_event_ext_ex or omniapi_read_event_block
struct name	directed_delivery
info type	dedicated

3.4.2.2 Related Messages

CQ52, CQ53

3.4.2.3 Purpose

This broadcast distributes deliveries and is dedicated to those parties that are referenced in the delivery as either owner of the delivery, receiver of the delivery due to delivery propagation on account, or if the either parties above has a delivery obligation to another party.

3.4.2.4 Structure

The BD18 BROADCAST has the following structure:

```
struct directed_delivery {
   struct broadcast_type
   struct cl_delivery_api
}
```

3.4.2.5 Usage and Conditions

All recipients are handled within their organisation, which means that all deliveries to a customer that belongs to an organisation is sent to the customer that is defined centrally to be the organisation owner.

To interpret the information correctly it is important to remember some clearing system fundamentals:

- Every entity that in some respect can change ownership involves a series, be it money or an ordinary financial product.
- The change of ownership itself is called a delivery.
- Everything that happens to a series during its lifetime is defined through product events.
- Product events are always released through a stimulus (often regarded as being the same thing as the event itself).

Sequence Number

The Sequence Number is sequential for each customer, instrument type and clearing date. This number can be used by the customer to discover missed dedicated delivery information. To recover a missed dedicated delivery broadcast, use the Delivery query.

Date

contains the date on which this delivery is created, that is the current business date.

Series

contains the binary series from which this delivery emanates. If, for example, this delivery is due to an exercise of a stock option. The series field contains the stock option series.

Original Delivery Number
Original Key Number

are only defined when Delivery type is either rollback or overtaking. In these cases these fields together with series, points out the delivery that this delivery either rolls back or overtakes. These fields are zero when Delivery Type is Normal.

Delivery Type

defines the types Normal, Rollback and Overtaking.

Originator Type

is set to Reversing if this delivery is created from a trade and the trade type on this trade is reversing. Otherwise this field is Normal.

Delivery State

defines if this delivery is active or rectified. When the delivery is sent as a broadcast it is always Normal.

Customer Account

is the Customer and Account for the Clearing Entity, Trade or Position, that this delivery is created from.

Delivery Account

is the account that handles the delivery for the Customer. This information is defined on Account level in the central system and is either Settlement Propagation or Delivery Propagation. If no propagation is set for the account, this field has the same value as **Customer Account**.

Delivery Account will for a DVP hold the account configured to handle deliveries for the clearing account. For other items, it will hold the configured settlement account.

Clearing Account

is the account that holds the position account. For a BD18 originating from a trade, **Clearing Account** will have the account set from Position Propagation on the trading account. If no propagation is set for the account, this field has the same value as **Customer Account**.

For a BD18 originating from a Position, **Clearing Account** has the same value as **Customer Account**.

Quantity, Delivery Base

defines the calculated quantity for the delivery. The sign is set from the clearing house's point of view (i.e. is delivered from the clearing house). The number of decimals used is specified by the decimals in premium in the DQ4/DQ123 query, for the class of the series defined in the Delivery Base.

Delivery Number, Key Number

gives together with country, market and instrument group in the Series field a unique combination for this delivery.

Origin, Delivery Number

defines the origin for this delivery. When the field value is different from Delivery Number it defines a trade number from which this delivery is calculated. The trade is then identified with this field and country, market, and instrument group from the Series field.

Settlement Date

defines the date when this delivery is to be settled.

Quantity, Delivery

defines the quantity for which this delivery is calculated from. It can be a trade quantity or a position amount.

Delivery, Base

is a series that defines what is delivered. The quantity for this is defined in the **Quantity, Delivery Base**.

Class Number

is a number indicating type of settlement for a delivery item.

3.4.3 BD24 [Cover Request Information BROADCAST]

3.4.3.1 Fingerprint

BROADCAST properties	
transaction type	BD24
calling sequence	omniapi_read_event_ext_ex or omniapi_read_event_block
struct name	cover_rec_bc
info type	dedicated

3.4.3.2 Related Messages

CQ71

3.4.3.3 Purpose

This broadcast is sent when a covered call request is made. Clearing members will see all covered call requests for members that they are responsible for.

3.4.3.4 Structure

The BD24 BROADCAST has the following structure:

```
struct cover_rec_bc {
   struct broadcast_type
   struct cover_data_item {
      struct series  // Named struct no: 50000
      struct account
      UINT8_T intraday_ind_c  // Intra-day Indicator
      char[3] filler_3_s  // Filler
```

```
        UINT32_T quantity_cover_u  // Quantity Cover
        UINT32_T quantity_tot_cover_u  // Quantity, total cover
        INT32_T quantity_request_i  // Quantity, request
        UINT8_T state_c  // State
        char[2] filler_2_s  // Filler
        CHAR filler_1_s  // Filler
        UINT32_T cover_request_nbr_u  // Cover Request Number
        UINT32_T sequence_nbr_u  // Sequence Number
        char[8] created_date_s  // Date, Created
        char[6] created_time_s  // Time, Created
        char[8] modified_date_s  // Date, Modified
        char[6] modified_time_s  // Time, Modified
        struct user_code
    }
}
```

3.4.4 BD26 [Cover Request Update BROADCAST]

3.4.4.1 Fingerprint

BROADCAST properties	
transaction type	BD26
calling sequence	omniapi_read_event_ext_ex or omniapi_read_event_block
struct name	cover_rec_upd_bc
info type	dedicated

3.4.4.2 Related Messages

CQ73

3.4.4.3 Purpose

The broadcast is sent when the state of a covered call request is updated. Clearing members will see all covered call request update broadcasts for members that they are responsible for.

3.4.4.4 Structure

The BD26 BROADCAST has the following structure:

```
struct cover_rec_upd_bc {
   struct broadcast_type
   struct cover_state_item {
      struct series  // Named struct no: 50000
      UINT32_T cover_request_nbr_u  // Cover Request Number
      UINT32_T sequence_nbr_u  // Sequence Number
      UINT8_T state_c  // State
      char[3] filler_3_s  // Filler
      char[8] modified_date_s  // Date, Modified
      char[6] modified_time_s  // Time, Modified
```

```
        char[2] filler_2_s  // Filler
    }
}
```

3.4.5 BD29 [Directed Give Up BROADCAST]

3.4.5.1 Fingerprint

BROADCAST properties	
transaction type	BD29
calling sequence	omniapi_read_event_ext_ex or omniapi_read_event_block
struct name	directed_give_up
info type	dedicated

3.4.5.2 Related Messages

CQ61, CQ76

3.4.5.3 Purpose

This broadcast is directed to those parties that are referenced in the giveup as either owner of the giveup or as receiver of the giveup. It is sent every time the giveup changes state. The field Give-Up Broadcast Reason simply explains why the broadcast was sent. The information about the giveup is exactly the same as in CA61.

3.4.5.4 Structure

The BD29 BROADCAST has the following structure:

```
struct directed_give_up {
   struct broadcast_type
   struct cl_give_up_api
}
```

3.4.5.5 Usage and conditions

Account

describes the destination member in the giveup. The 10 last characters may be left blank, thus only defining the member, or set to point out a specific account.

Party

identifies the customer that gives up the trade.

Sequence Number

is sequential for each **Customer**,**Instrument Type** and **Clearing Date** and starts from one each clearing date. The Sequence Number field can be used by the customer to keep track of potentially missed broadcasts. To recover a missed dedicated broadcast, CQ76 must be used.

Give-Up Broadcast Reason

contains a slogan denoting the reason for sending the broadcast. It mirrors the change of **State** of the giveup itself.

In order to differentiate between a reject by the take-up party and a delete/withdrawal by the give-up party, the new status value “Deleted” has been added as a possible state on a give-up request:

- The system detects whether the take-up party is rejecting the give-up, in which case the give-up request will be put in state Rejected.
- If another member have been granted the right to act on behalf of the take-up party, then the give-up request will also be put in state Rejected.
- Otherwise, if the delete/withdrawal is done by the give-up party, the give-up request will be put in state “Deleted.”
- If a Clearing Office user does reject/delete a give-up request, the action will put the give up reason in state “Deleted.”

Deal Source

data refer to the original trade's deal source.

The following fields describe the trade that is subject to the giveup:

- Series
- Party
- Bought or Sold
- Quantity, Trade
- Price, Deal
- Trade Number
- Date, Created
- Time, Created
- Date, As Of
- Time, As Of
- Original Clearing Date
- Old Trade Indicator
- Deal Source
- External Trade Fee Type
- Trade Number, External
- Original Trade Number, External

The Quantity, Trade field specifies the give-up portion of the trade.

Of these, Date, As Of; Time, As Of; Original Clearing Date; Old Trade Indicator; Deal Source; and External Trade Fee Type only contain significant data for give-up requests made the current business day and whose states are either holding or completed.

Give-Up Number; State; Account; Give-Up Free Text; and Clearing Date are fields that describe the giveup. Clearing Date is the clearing date of the giveup itself.

3.4.6 BD39 [Dedicated Trade Change Information BROADCAST]

3.4.6.1 Fingerprint

BROADCAST properties	
transaction type	BD39
calling sequence	omniapi_read_event_ext_ex or omniapi_read_event_block
struct name	directed_trade_change
info type	dedicated

3.4.6.2 Related Messages

Dedicated Trade Information Broadcast and CQ39

3.4.6.3 Purpose

The purpose of BD39 is to inform API clients about changes in trades that have been previously sent out with Dedicated Trade Information Broadcasts.

3.4.6.4 Structure

The BD39 BROADCAST has the following structure:

```
struct directed_trade_change {
   struct broadcast_type
   struct cl_trade_change_api
}
```

3.4.6.5 Usage and conditions

The broadcast data is a limited number of fields in the trade that can be changed after trade creation.

The broadcast shows a snapshot of the fields at the moment the broadcast is sent.

It has a sequence number per instrument type. The receiver is guaranteed to receive an unbroken sequence of numbers. The receiver is also guaranteed that BD39 are only sent for previously received trades.

3.4.7 BD40 [Dedicated auxiliary position info update information BROADCAST]

3.4.7.1 Fingerprint

BROADCAST properties	
transaction type	BD40
calling sequence	omniapi_read_event_ext_ex or omniapi_read_event_block
struct name	directed_pos_info_update
info type	dedicated

3.4.7.2 Related Messages

CQ40

3.4.7.3 Purpose

This broadcast is disseminated when any of the auxiliary information associated with a position is updated.

3.4.7.4 Structure

The BD40 BROADCAST has the following structure:

```
struct directed_pos_info_update {
   struct broadcast_type
   struct pos_info_update_api
}
```

3.4.7.5 Usage and conditions

The auxiliary information consists of :

- quantity to be exempted from automatic/general exercise (deny exercise)
- quantity closed-out current clearing date
- quantity of underlying used as cover for short position (exchange specific)

The time for the most recent update of auxiliary information on the position is provided as modified time and date.

The time and date from the most recently received BD40 is intended to be used as input to a CQ40 query transaction in order to retrieve the information distributed in BD40 broadcasts while the API-client is disconnected.

3.4.8 BI27 [Clearing message BROADCAST]

3.4.8.1 Fingerprint

BROADCAST properties	
transaction type	BI27
calling sequence	omniapi_read_event_ext_ex or omniapi_read_event_block
struct name	clearing_message
info type	general

3.4.8.2 Purpose

This is a Clearing Message broadcast. The text is sent from the Clearinghouse and all connected Back Office applications have the possibility to display the message.

3.4.8.3 Structure

The BI27 BROADCAST has the following structure:

```
struct clearing_message {
   struct broadcast_type
   UINT16_T broadcast_number_n  // Broadcast Number
   UINT8_T country_c  // Country Number
   UINT8_T market_c  // Market Code
   UINT16_T items_n  // Items
   Array ITEM [max no: 10] {
      char[80] text_line_s  // Text, Line
   }
}
```

3.4.8.4 Usage and conditions

Market

If the **Country Number** field in Market is = 0, the message concerns all Exchanges, otherwise a specific Country Cumber is specified.

If the **Market Code** field in Market is = 0 the message concerns all markets, otherwise a specific Market Code is specified.

Text Buffer

contains 80 characters lines, completed with trailing spaces, but no carriage return or other control characters.

3.4.9 CC11 [Cancel Holding Rectify Trade TRANSACTION]

3.4.9.1 Fingerprint

TRANSACTION properties	
transaction type	CC11
calling sequence	omniapi_tx_ex
struct name	confirm_rectify_t
facility	EP3
partitioned	false

3.4.9.2 Related Messages

CQ14, CQ15

3.4.9.3 Purpose

This transaction is used to cancel a previously sent rectify trade request.

3.4.9.4 Structure

The CC11 TRANSACTION has the following structure:

```
struct confirm_rectify_t {
   struct transaction_type
   struct series  // Named struct no: 50000
   INT32_T rectify_trade_number_i  // Rectify Trade Number
   UINT8_T confirm_reject_c  // Confirm or Reject
   char[3] filler_3_s  // Filler
}
```

3.4.9.5 Usage and conditions

Series

must be set to Series from original trade.

Rectify Trade Number

must be set to the rectify trade number identifying the trade rectification in question.

Confirm or Reject

must be set to Delete.

3.4.10 CC13 [Exercise Request TRANSACTION]

3.4.10.1 Fingerprint

TRANSACTION properties	
transaction type	CC13
calling sequence	omniapi_tx_ex
struct name	exercise_req
facility	EP3
partitioned	false

3.4.10.2 Purpose

The purpose of this transaction is to request an exercise.

3.4.10.3 Structure

The CC13 TRANSACTION has the following structure:

```
struct exercise_req {
   struct transaction_type
   struct series  // Named struct no: 50000
   struct account
   INT64_T quantity_i  // Quantity
   INT32_T trade_number_i  // Trade Number
}
```

3.4.10.4 Usage and conditions

Trade Number

An exercise is done on either a position or on a trade, depending on the product (security lending is an example of a product which is exercised on trades). The Trade Number is only filled in on exercise on trades, otherwise it is zero.

3.4.11 CC14 [Deny Exercise Request TRANSACTION]

3.4.11.1 Fingerprint

TRANSACTION properties	
transaction type	CC14

TRANSACTION properties	
calling sequence	omniapi_tx_ex
struct name	set_deny_exercise
facility	EP3
partitioned	false

3.4.11.2 Purpose

The purpose of this transaction is to inform the Central System that a certain quantity for an account should not participate in an automatic exercise. If this quantity exceeds the held position, the whole position is excluded from automatic exercise.

3.4.11.3 Structure

The CC14 TRANSACTION has the following structure:

```
struct set_deny_exercise {
   struct transaction_type
   struct series  // Named struct no: 50000
   struct account
   INT64_T deny_exercise_q  // Deny Exercise
}
```

3.4.12 CC15 [Cancel Exercise Request TRANSACTION]

3.4.12.1 Fingerprint

TRANSACTION properties	
transaction type	CC15
calling sequence	omniapi_tx_ex
struct name	annul_exercise_req
facility	EP3
partitioned	false

3.4.12.2 Related Messages

CQ21

3.4.12.3 Purpose

The purpose of this transaction is to cancel an earlier entered exercise request. The exercise request must be pending, to allow cancel request. The exercise request number can be retrieved by using the Query Pending Exercise Request Transaction, see **CQ21**.

3.4.12.4 Structure

The CC15 TRANSACTION has the following structure:

```
struct annul_exercise_req {
   struct transaction_type
   struct series  // Named struct no: 50000
   INT32_T exercise_number_i  // Exercise, Request Number
}
```

3.4.12.5 Usage and conditions

Series

must be set to the Series of the exercise request to be cancelled.

Exercise Request Number

must be set to the exercise request number identifying the exercise request to be cancelled.

3.4.13 CC38 [Confirm Give up Request TRANSACTION]

3.4.13.1 Fingerprint

TRANSACTION properties	
transaction type	CC38
calling sequence	omniapi_tx_ex
struct name	confirm_give_up_request
facility	EP3
partitioned	false

3.4.13.2 Related Messages

CQ61

3.4.13.3 Purpose

This transaction is used to confirm a give-up trade to the member. Use CQ61 to retrieve information on give-up trades in holding state.

3.4.13.4 Structure

The CC38 TRANSACTION has the following structure:

```
struct confirm_give_up_request {
```

```
    struct transaction_type
    struct series  // Named struct no: 50000
    INT32_T give_up_number_i  // Give Up, Number
    UINT16_T items_n  // Items
    char[2] filler_2_s  // Filler
    Array ITEM [max no: 50] {
       struct account
       INT64_T trade_quantity_i  // Quantity, Trade
       UINT8_T open_close_req_c  // Open  Close Request
       char[15] customer_info_s  // Customer, Information
    }
}
```

3.4.13.5 Usage and conditions

Series
Give-Up Number

identifies the giveup.

Quantity, Trade

is the quantity to place on the specified account. The sum of all quantities in the destination trade must be equal to the quantity in the giveup.

Account

contains identity of the account receiving the trade.

The **Customer Information** and **Open Close Request** are optional.

3.4.14 CC40 [Reject Give up Request TRANSACTION]

3.4.14.1 Fingerprint

TRANSACTION properties	
transaction type	CC40
calling sequence	omniapi_tx_ex
struct name	reject_give_up_request
facility	EP3
partitioned	false

3.4.14.2 Related Messages

CQ61

3.4.14.3 Purpose

This transaction is used to reject a give-up request. Use CQ61 to retrieve information on give-up trades in holding state.

3.4.14.4 Structure

The CC40 TRANSACTION has the following structure:

```
struct reject_give_up_request {
   struct transaction type
   struct series  // Named struct no: 50000
   INT32_T give_up_number_i  // Give Up, Number
   char[30] give_up_text_s  // Give Up, Free Text
   char[2] filler_2_s  // Filler
}
```

3.4.14.5 Usage and conditions

Series
Give-Up Number

identifies the giveup.

Give-up Free Text

is filled with the text set by the sending user. The text can be modified to hold a reject reason for the sender.

3.4.15 CC47 [Cover Request TRANSACTION]

3.4.15.1 Fingerprint

TRANSACTION properties	
transaction type	CC47
calling sequence	omniapi_tx_ex
struct name	cover_req
facility	EP3
partitioned	false

3.4.15.2 Purpose

This transaction is used to submit a request to have a position (partially) covered by the underlying equity.

3.4.15.3 Structure

The CC47 TRANSACTION has the following structure:

```
struct cover_req {
   struct transaction_type
   struct series  // Named struct no: 50000
   struct account
   INT32_T cover_quantity_i  // Cover Quantity
}
```

3.4.15.4 Usage and conditions

Series
Account

specify the position to cover.

Quantity, Request

is the number of underlyings to add or remove as cover for the position.

Genium INET Trading will validate this request and accept it if it is all true that:

1. The position exists.
2. The series is a call option.
3. No other pending covered call request exists on the same account and underlying to use as cover.
4. The sum of the number to add and the quantity of underlying already used as cover for the position is less than (contract size) x (short position).
5. The quantity of underlying shares is a multiple of the contract size of the series.

If contract size is 100, the allowed quantities of shares to use as cover is 100, 200, 300 and so on. Note that in case of capital adjustment there can be a fractional part in the contract size. If the contract size is 104,35 only the integer part should be taken into consideration when a covered call request is made. In this case, using 104 shares as cover will cover 1 contract, 208 shares wilcall request is made. In this case, using 104 shares as cover will cover 1 contract, 208 shares cover 2 contracts and so on. In case of no coverage only the integer part of contract size is to be margined.

Releasing underlying equity is done with the same transaction by specifying a negative quantity. The transaction undergoes the same validation, except that the number to remove should be less or equal to the quantity already used as cover for the position.

> **Note:** All requests still pending at ‘After Business’ will be rejected.

3.4.16 CC48 [Cancel Cover Request TRANSACTION]

3.4.16.1 Fingerprint

TRANSACTION properties	
transaction type	CC48
calling sequence	omniapi_tx_ex
struct name	cancel_cover_req

TRANSACTION properties	
facility	EP3
partitioned	false

3.4.16.2 Purpose

This transaction is used to cancel a Covered Call Request that is in a pending state.

3.4.16.3 Structure

The CC48 TRANSACTION has the following structure:

```
struct cancel_cover_req {
   struct transaction_type
   struct series  // Named struct no: 50000
   struct account
   UINT32_T cover_request_nbr_u  // Cover Request Number
}
```

3.4.16.4 Usage and conditions

Series

must be complete up to Country Number, Market Code and Instrument Group.

Cover Request Number
Series

identify the request to be cancelled.

3.4.17 CC94 [Cross Product Netting TRANSACTION]

3.4.17.1 Fingerprint

TRANSACTION properties	
transaction type	CC94
calling sequence	omniapi_tx_ex
struct name	cl_cross_product_netting
facility	EP3
partitioned	false

3.4.17.2 Related Messages

3.4.17.3 Purpose

Command from the Back Office (application) to perform cross product netting.

3.4.17.4 Structure

The CC94 TRANSACTION has the following structure:

```
struct cl_cross_product_netting {
   struct transaction_type
   struct series  // Named struct no: 50000
   struct account
   struct netting_series  // Of type: SERIES ; Named struct no: 50000
   INT64_T td_long_q  // Today long position
   INT64_T mini_td_long_q  // Today long position ; Of type: TD_LONG_Q
   INT64_T td_short_q  // Today short position
   INT64_T mini_td_short_q  // Today short position ; Of type: TD_SHORT_Q
   INT32_T netting_ratio_i  // Netting Ratio
   char[8] clearing_date_s  // Clearing Date
}
```

3.4.17.5 Usage and conditions

Execution of cross product netting. The transaction could fail for a number of reasons. For example if there have been an update of the position between the search and the execution. Execution will be in "all or nothing" manner, i.e the action will fail if the request needs 10 lots while only 9 lots are available.

Clearing Date

specifies the clearing date for the cross product netting operation. Must be the current clearing date or, in case clearing for T + 1 is enabled for the instrument, the next clearing date. A blank field is interpreted as the current clearing date.

3.4.18 CD5 [Transitory Account Trades TRANSACTION]

3.4.18.1 Fingerprint

TRANSACTION properties	
transaction type	CD5
calling sequence	omniapi_tx_ex
struct name	cl_reregistration_bo
facility	EP3
partitioned	false

3.4.18.2 Purpose

This transaction is used to transfer trades from the daily account to the client account. It is used by the Back Office (application) and identifies a trade by using the unique Trade Number.

3.4.18.3 Structure

The CD5 TRANSACTION has the following structure:

```
struct cl_reregistration_bo {
   struct transaction type
   struct series  // Named struct no: 50000
   UINT8_T items_c  // Item
   char[3] filler_3_s  // Filler
   Array ITEM [max no: 100] {
      struct account
      INT32_T trade_number_i  // Trade Number
      INT64_T deal_quantity_i  // Quantity, Deal
      char[15] customer_info_s  // Customer, Information
      char[2] reserved_2_s  // Reserved
      CHAR reserved_1_c  // Reserved
      UINT8_T open_close_req_c  // Open  Close Request
      CHAR filler_1_s  // Filler
   }
}
```

3.4.18.4 Usage and conditions

Series

must be completely specified.

This function is related only to Client Clearing and thus not valid for Member Clearing. In a client clearing model, the Exchange provides the clearing service on anonymous client identities for the customers.

A certain trade can be transferred to one or several client accounts. It is possible to request how the positions should be updated. This transaction, a synchronous transaction, will allow the choices open, close, and normal.

If client information is omitted, the client identity in the original trade will be used.

The transaction can fail for a number of reasons. The CD5 transaction is synchronous and will not work unless the transfer actually is performed.

A Daily Account Trades transaction may be canceled. This is achieved by canceling the deal, created by the Daily Account Trades transaction that transfers the trade to the client account. The deal is canceled by use of the Rectify Deal transaction.

A Daily Account Trades transaction can only be canceled on the same business day as it is created.

3.4.19 CD27 [Rectify Trade (Open/Close) TRANSACTION]

3.4.19.1 Fingerprint

TRANSACTION properties	
transaction type	CD27
calling sequence	omniapi_tx_ex
struct name	rectify_trade
facility	EP3
partitioned	false

3.4.19.2 Related Messages

CD28

3.4.19.3 Purpose

This rectify transaction is used for changing insensitive parts of a trade. For the moment it is only possible to change Open Close Request from Open to Close. This rectify is executed immediately. For all other types of rectifications, CD28 must be used.

3.4.19.4 Structure

The CD27 TRANSACTION has the following structure:

```
struct rectify_trade {
   struct transaction_type
   struct series  // Named struct no: 50000
   INT32_T trade_number_i  // Trade Number
   UINT8_T items_c  // Item
   char[3] filler_3_s  // Filler
   Array ITEM [max no: 100] {
      struct account
      INT64_T trade_quantity_i  // Quantity, Trade
      UINT8_T open_close_req_c  // Open  Close Request
      char[15] customer_info_s  // Customer, Information
   }
}
```

3.4.19.5 Usage and conditions

Series
Trade number

identify the trade to be rectified.

Item

must be set to 1, since the trade to be rectified can not be split into several overtaking trades.

Open Close Request

must be set to Mandatory Close.

Account
Quantity, Trade
Customer Info

these fields must be identical to that of the trade to be rectified.

3.4.20 CD28 [Rectify Trade TRANSACTION]

3.4.20.1 Fingerprint

TRANSACTION properties	
transaction type	CD28
calling sequence	omniapi_tx_ex
struct name	rectify_trade
facility	EP3
partitioned	false

3.4.20.2 Related Messages

CD27

3.4.20.3 Purpose

This transaction is used for changes of trades. The changes may have to be confirmed by the clearinghouse. The externally allowed number of days for rectification for the instrument type is checked before the operation is carried through.

If Open Close request are to be changed from Open to Close, CD27 must be used.

3.4.20.4 Structure

The CD28 TRANSACTION has the following structure:

```
struct rectify_trade {
   struct transaction_type
   struct series  // Named struct no: 50000
   INT32_T trade_number_i  // Trade Number
   UINT8_T items_c  // Item
   char[3] filler_3_s  // Filler
```

```
    Array ITEM [max no: 100] {
       struct account
       INT64_T trade_quantity_i  // Quantity, Trade
       UINT8_T open_close_req_c  // Open  Close Request
       char[15] customer_info_s  // Customer, Information
    }
 }
```

3.4.20.5 Usage and conditions

Series
Trade number

identify the trade to be rectified.

Item

the number of overtaking trades to be created by the rectification.

Account

the desired destination account of an overtaking trade.

Open Close Request

the desired Open Close Request of the overtaking trade.

Customer Information

the desired Customer Information of the overtaking trade.

Quantity, Trade

the desired quantity of a overtaking trade. The sum of the quantities of the overtaking trades must equal the quantity of the trade to be rectified.

3.4.21 CD32 [Average Price Trade TRANSACTION]

3.4.21.1 Fingerprint

TRANSACTION properties	
transaction type	CD32
calling sequence	omniapi_tx_ex
struct name	average_price_trade
facility	EP3
partitioned	false

3.4.21.2 Purpose

This transaction groups a number of trades into an average price trade. All trades must be of the same type, in the same series, and on the same account.

3.4.21.3 Structure

The CD32 TRANSACTION has the following structure:

```
struct average_price_trade {
   struct transaction_type
   struct series  // Named struct no: 50000
   UINT16_T items_n  // Items
   char[2] filler_2_s  // Filler
   Array ITEM [max no: 1000] {
      INT32_T trade_number_i  // Trade Number
   }
}
```

3.4.21.4 Usage and conditions

The specified trades are transferred to a member-specific account dedicated for this transaction. A new deal with the average price for the trades is then created. It nets out the position on the account and returns the position to the original account.

> **Note:** This transaction may in the future rectify the trades to the member specific account dedicated for this transaction.

In case clearing for T + 1 is enabled for the instrument, all trades included in the average price trade must have the same clearing date, that is, T and T + 1 may not be mixed in an APT transaction. The resulting APT trade inherits the clearing date from the trades included in the operation.

The resulting trade with average price will have Deal Source set to Average Price Trade (128). Intermediate trades created during the Average Price Trade transaction will have Deal Source set to Intermediate APT (129).

An Average Price Trade transaction can only be canceled in the case the clearing date of the transaction is still open for post-trade transactions, that is, before day-end processing for the clearing date.

> **Note:** In case the resulting average price trade has been subject to Daily Account Trades transaction(s), these must first be canceled before the Average Price Trade transaction can be canceled.

Series

must be completed with **Country Number**, **Market Code** and **Instrument Group**.

3.4.21.5 Return Codes

After a successful Average Price Trade transaction, the trade number for the average price trade will be returned to the sender.

cstatus	txstat
successfull	trade number for newly created average price trade
Transaction aborted	...

Please refer to the **Error Messages Reference Manual** for details about why transactions are aborted.

3.4.22 CD34 [Transfer Position TRANSACTION]

3.4.22.1 Fingerprint

TRANSACTION properties	
transaction type	CD34
calling sequence	omniapi_tx_ex
struct name	cl_transfer_position
facility	EP3
partitioned	false

3.4.22.2 Purpose

The purpose of this transaction is to let a participant transfer positions from one account to another account.

3.4.22.3 Structure

The CD34 TRANSACTION has the following structure:

```
struct cl_transfer_position {
   struct transaction_type
   struct series  // Named struct no: 50000
   struct account
   struct new_account
   INT64_T nbr_held_q  // Held
   INT64_T nbr_written_q  // Written
   UINT8_T open_close_req_c  // Open  Close Request
   char[3] filler_3_s  // Filler
   char[8] clearing_date_s  // Clearing Date
}
```

3.4.22.4 Usage and conditions

Series

must be a complete series.

Account

is where the position exists.

New Account

is where the position is transferred. It must be an account within the same member.

Open Close Request

the desired Open Close effect of the transferred position on the destination account.

Held
Written

are the quantities that are transferred. One of the fields must have a positive value.

Clearing Date

specifies the clearing date for the transfer operation. Must be the current clearing date or, in case clearing for T + 1 is enabled for the instrument, the next clearing date. A blank field is interpreted as the current clearing date.

3.4.23 CD35 [Give up Request TRANSACTION]

3.4.23.1 Fingerprint

TRANSACTION properties	
transaction type	CD35
calling sequence	omniapi_tx_ex
struct name	give_up_request
facility	EP3
partitioned	false

3.4.23.2 Purpose

This transaction is used to give up a trade to another member.

3.4.23.3 Structure

The CD35 TRANSACTION has the following structure:

```
struct give_up_request {
   struct transaction_type
   struct series  // Named struct no: 50000
   struct account
   INT32_T trade_number_i  // Trade Number
   INT64_T trade_quantity_i  // Quantity, Trade
   INT32_T commission_i  // Commission
   char[30] give_up_text_s  // Give Up, Free Text
   char[2] filler_2_s  // Filler
}
```

3.4.23.4 Usage and conditions

Series
Trade Number

identifies the trade that is given up.

Account

must contain the country and customer identities of the member receiving the trade. It is optional to set the account id in Account. If not set, it must be left blank.

Quantity, Trade

is the given up quantity of the trade. This value does not have to be the whole trade quantity.

Give-up Free Text

contains a user supplied text as information to the receiving member.

3.4.24 CD38 [Long Position Adjustment TRANSACTION]

3.4.24.1 Fingerprint

TRANSACTION properties	
transaction type	CD38
calling sequence	omniapi_tx_ex
struct name	long_position_adj
facility	EP3
partitioned	false

3.4.24.2 Purpose

The purpose of this transaction is to net a position by closing an equal amount of long and short contracts respectively.

3.4.24.3 Structure

The CD38 TRANSACTION has the following structure:

```
struct long_position_adj {
   struct transaction_type
   struct series  // Named struct no: 50000
   char[2] filler_2_s  // Filler
   UINT16_T items_n  // Items
   char[8] clearing_date_s  // Clearing Date
   Array ITEM [max no: 1500] {
      struct account
```

```
        struct series  // Named struct no: 50000
        INT32_T long_adjustment_i  // Long Adjustment
    }
}
```

3.4.24.4 Usage and conditions

Positions is only retrieved for instruments having the Maintain Positions parameter set to Yes.

Series

must belong to the same instrument type both in the transaction header and for all items sent.

Account, Series

together identify the position to be adjusted.

Long adjustment

the number of contracts to be closed.

Clearing Date

specifies the clearing date for the position adjustment operation. Must be the current clearing date or, in case clearing for T + 1 is enabled for the instrument, the next clearing date. A blank field is interpreted as the current clearing date.

3.4.25 CQ3 [Position QUERY]

3.4.25.1 Fingerprint

QUERY properties	
transaction type	CQ3
calling sequence	omniapi_query_ex
struct name	query_position
facility	EP3
partitioned	true
answers	CA3

ANSWER properties	
transaction type	CA3
struct name	answer_position
segmented	true

3.4.25.2 Purpose

This transaction will retrieve the current positions for each deposit and series belonging to the customer, alternatively the final position for the previous date.

> **Note:** Positions will only be retrieved for instruments having the Maintain Positions property set to Yes.

3.4.25.3 Structure

The CQ3 QUERY has the following structure:

```
struct query_position {
   struct transaction_type
   struct series  // Named struct no: 50000
   struct search_series
   struct account
   UINT16_T segment_number_n  // Segment Number
   char[8] date_s  // Date
   char[2] filler_2_s  // Filler
}
```

3.4.25.4 Usage and conditions

Series

must be complete up to **Country number**, **Market code** and **Instrument group**.

Segment Number

is one for the first query and then incremented.

Search Series
Account

identifies the positions to be returned in the answer.

Date

must be valid and have one of the following values:

- Previous calendar date: The overnight (O/N) position is returned. These positions are static during the day.
- Today's business date. The current position for the current clearing date (provided it exists for the instrument) is returned.
- Next calendar date. The current position for the next clearing date is returned; trades as of next clearing date are added to the current clearing date position.

Note that the previous and next calendar date is in relation to current business date in the system. For example, the previous calendar date will refer to a Sunday when current business date is a Monday.

3.4.25.5 Answer Structure

The CA3 ANSWER has the following structure:

```
struct answer_position {
   struct transaction_type
   struct partition_low
   struct partition_high
   UINT16_T segment_number_n  // Segment Number
   UINT16_T items_n  // Items
   Array ITEM [max no: 500] {
      struct series  // Named struct no: 50000
      char[8] modified_date_s  // Date, Modified
      char[6] modified_time_s  // Time, Modified
      UINT8_T reserved_prop_c  // Reserved Properties
      CHAR filler_1_s  // Filler
      INT64_T nbr_held_q  // Held
      INT64_T nbr_written_q  // Written
      INT64_T deny_exercise_q  // Deny Exercise
      struct account
      UINT32_T quantity_cover_u  // Quantity Cover
      INT64_T qty_closed_out_q  // Quantity, Closed out
   }
}
```

3.4.25.6 Answer, comments

If the maximum number of items for one transaction is returned, the query should be repeated with incremented segment number.

Quantity, Cover

states the quantity of underlying equity that is used as cover for this position. This field is normally set to zero. Only if the query's**Date** was set to **Today's calendar date** can this field have a non-zero value.

When used to retrieve information about the position for the previous calendar day:

- If the position has not changed during the current day, the modification date and modification time have the last modification noted for that position (i.e. may be earlier than yesterday).
- If the position has changed during the current day, the modification fields are not valid (the time is set to 00:00:00 and the date to current date).

3.4.26 CQ8 [Fixing Values QUERY]

3.4.26.1 Fingerprint

QUERY properties	
transaction type	CQ8
calling sequence	omniapi_query_ex
struct name	query_fixing_val
facility	EP5
partitioned	false
answers	CA8

ANSWER properties	
transaction type	CA8
struct name	answer_fixing_val
segmented	true

3.4.26.2 Purpose

This transaction retrieves fixing value for cash settled products (on a daily basis, when they are exercised or when they are closed).

3.4.26.3 Structure

The CQ8 QUERY has the following structure:

```
struct query_fixing_val {
   struct transaction_type
   struct series  // Named struct no: 50000
   struct search_series
   UINT16_T segment_number_n  // Segment Number
   char[8] date_s  // Date
   char[2] filler_2_s  // Filler
}
```

3.4.26.4 Usage and conditions

Search Series

Country Number, **Market Code** and **Instrument Group** can be filled in to filter the response.

If zero is filled in for the fields, the avista value for all Series is returned.

Date

is Clearing date for which fixing values that are to be returned in the answer.

Segment Number

is one for the first query and then incremented.

3.4.26.5 Answer Structure

The CA8 ANSWER has the following structure:

```
struct answer_fixing_val {
   struct transaction type
   struct partition low
   struct partition high
   UINT16_T segment_number_n  // Segment Number
   UINT16_T items_n  // Items
   Array ITEM [max no: 500] {
      struct series  // Named struct no: 50000
      INT32_T fixing_value_i  // Fixing Value
      UINT16_T dec_in_fixing_n  // Decimals, Fixing
      char[2] filler_2_s  // Filler
   }
}
```

3.4.26.6 Answer, comments

If the maximum number of items for one transaction is returned, the query should be repeated with incremented segment number.

3.4.27 CQ10 [Query missing trade QUERY]

3.4.27.1 Fingerprint

QUERY properties	
transaction type	CQ10
calling sequence	omniapi_query_ex
struct name	query_missing_trade
facility	EP3
partitioned	false
answers	CA10

VIA properties	
transaction type	CA10
struct name	The message complies with the VIM concept and has no top struct. The sequence of possible structs is described in the Structure section.
segmented	false

3.4.27.2 Related Messages

BD6 (Dedicated Trade Information VIB)

CQ11 (Query Missing Trade, Historical Query).

3.4.27.3 Purpose

This query is used to retrieve trades for the trading day (T) = current business day; and the next trading day (T+1) when the next trading day commence on the same business day. For example, if a missing sequence number is detected for the trade broadcast, this query is used to get in synch with the broadcast flow again.

To retrieve trades for previous trading days, use CQ11.

3.4.27.4 Structure

The CQ10 QUERY has the following structure:

```
struct query_missing_trade {
   struct transaction_type
   struct series  // Named struct no: 50000
   INT32_T sequence_first_i  // Number, First Sequential
   INT32_T sequence_last_i  // Number, Last Sequential
   char[8] date_s  // Date
}
```

3.4.27.5 Usage and Conditions

CQ10, CQ11 and the Dedicated Trade Information Broadcast form a package. CQ10 returns data as in the format of a Dedicated Trade Information Broadcast.

Series

must be completed with **Country Number**, **Market Code** and **Instrument Group**.

Sequence Number

The first Sequence Number is the first missing one, the second is the last missing one. If the second Sequence Number is equal to zero, all available trades are sent in sequence.

If the maximum number of items for one transaction is returned, the query should be repeated with the next missing sequence number as first argument.

The maximum number of items is reached when the items_n field contains a value greater than 0.

Date

must be current or next clearing date.

Next clearing date is only allowed at installations where trading for the next day commences in the afternoon or evening on the day before. An additional requirement is that the clearing system is configured for accepting trades for the following day.

3.4.27.6 Answer Structure

The CA10 VIA has the following structure:

```
struct answer_missing_trade_hdr {
   struct transaction type
   char[2] filler 2 s  // Filler
   UINT16 T items n  // Items
}
Sequence {
   struct item hdr
   Sequence {
      struct sub item hdr
      Choice {
         struct cl trade base api  // Named struct no: 3
         struct cl trade secur part  // Named struct no: 20
      }
   }
}
```

3.4.27.7 Answer, comments

The answer is built up with variable trade structures. Each trade is built with several sub-structures to form a flow of data in which each trade can consist of one or several structures. A trade consists at least of the structure cl_trade_base_api. Each sub-structure is prefaced with a header.

3.4.28 CQ11 [Query missing trade, historical QUERY]

3.4.28.1 Fingerprint

QUERY properties	
transaction type	CQ11
calling sequence	omniapi_query_ex
struct name	query_api_trade
facility	EP5
partitioned	false
answers	CA11

VIA properties	
transaction type	CA11
struct name	The message complies with the VIM concept and has no top struct. The sequence of possible structs is described in the Structure section.
segmented	false

3.4.28.2 Related Messages

BD6 (Dedicated Trade Information VIB) and CQ10 (Query Missing Trade Query).

3.4.28.3 Purpose

This query is used to retrieve historical trades, i.e for trading days before the current business day. The information is available to the participant the next business day. Historical trades are queried per instrument type. To retrieve trades for the current trading day and next trading day, use CQ10.

3.4.28.4 Structure

The CQ11 QUERY has the following structure:

```
struct query_api_trade {
   struct transaction_type
   struct series  // Named struct no: 50000
   char[8] from_date_s  // Date, From
   INT32_T sequence_first_i  // Number, First Sequential
   char[8] to_date_s  // Date, To
   INT32_T sequence_last_i  // Number, Last Sequential
}
```

3.4.28.5 Usage and Conditions

CQ10, CQ11 and BD6 form a package. CQ11 returns data as in the format of a Dedicated Trade Information Broadcast.

Series

must be completed with **Country Number**, **Market Code** and **Instrument Group**. **Commodity** can be given to retrieve all trades for a specific instrument class. Otherwise Commodity is left to zero.

Date, From and Date, To

must be historical dates compared to current business date. **Date, From** must be less or equal to **Date, To**.

Sequence Number 1

is the first item to get for **Date, From**. Zero or one means the first item for that date.

Sequence Number 2

is the last item to get for **Date, To**. Zero means the last item for that date.

3.4.28.6 Answer Structure

The CA11 VIA has the following structure:

```
struct answer_api_trade_hdr {
   struct transaction_type
   struct series  // Named struct no: 50000
```

```
      char[8] from_date_s  // Date, From
      INT32_T sequence_first_i  // Number, First Sequential
      UINT16_T items_n  // Items
      char[2] filler_2_s  // Filler
   }
   Sequence {
      struct item_hdr
      Sequence {
         struct sub_item_hdr
         Choice {
            struct cl_trade_base_api  // Named struct no: 3
            struct cl_trade_secur_part  // Named struct no: 20
         }
      }
   }
```

3.4.28.7 Answer, comments

See CQ10.

3.4.29 CQ14 [Holding Rectify Trade QUERY]

3.4.29.1 Fingerprint

QUERY properties	
transaction type	CQ14
calling sequence	omniapi_query_ex
struct name	query_rectify_t
facility	EP3
partitioned	true
answers	CA14

ANSWER properties	
transaction type	CA14
struct name	answer_rectify_t
segmented	false

3.4.29.2 Related Messages

CQ15, CC11

3.4.29.3 Purpose

This query is used for retrieving information on requests to rectify trades. The query will only return information on requests that initially were placed in a holding state awaiting confirmation by the exchange

or clearinghouse. Whether a request to rectify a trade requires confirmation or not depends on the exchange/clearinghouse policy.

Use CQ15 to get detailed information regarding a holding rectify trade.

Use CC11 to withdraw ("reject") a request to rectify a trade.

3.4.29.4 Structure

The CQ14 QUERY has the following structure:

```
struct query_rectify_t {
   struct transaction type
   struct series  // Named struct no: 50000
   UINT8_T instance_c  // Instance, Number
   CHAR filler_1_s  // Filler
   UINT16_T segment_number_n  // Segment Number
   struct search series
}
```

3.4.29.5 Usage and conditions

Series

must be completed with **Country Number**, **Market Code** and **Instrument Group**.

Search Series

filters on instruments in trades subject to rectify trade requests that are to be returned in the answer.

Segment Number

is one for the first query and then incremented.

Instance, Number

is ignored.

3.4.29.6 Answer Structure

The CA14 ANSWER has the following structure:

```
struct answer_rectify_t {
   struct transaction type
   UINT16_T segment_number_n  // Segment Number
   char[2] reserved_2_s  // Reserved
   struct partition_low
   struct partition_high
   UINT16_T items_n  // Items
   UINT8_T instance_next_c  // Next Instance Number
   CHAR filler_1_s  // Filler
   Array ITEM [max no: 400] {
      struct ans_rect_t_item {
         char[8] created_date_s  // Date, Created
         char[6] created_time_s  // Time, Created
```

```
            char[8] asof_date_s  // Date, As Of
            char[6] asof_time_s  // Time, As Of
            char[8] clearing_date_s  // Clearing Date
            char[8] orig_clearing_date_s  // Clearing Date, Original
            struct trading_code
            struct user_code
            struct series  // Named struct no: 50000
            INT32_T trade_number_i  // Trade Number
            INT32_T rectify_trade_number_i  // Rectify Trade Number
            INT32_T ext_seq_nbr_i  // External Clearinghouse, Sequence Number
            UINT8_T state_c  // State
            UINT8_T bought_or_sold_c  // Bought or Sold
            UINT8_T reserved_prop_c  // Reserved Properties
            CHAR filler_1_s  // Filler
            struct new_account
            struct account
            INT64_T trade_quantity_i  // Quantity, Trade
            INT32_T deal_price_i  // Price, Deal
        }
    }
}
```

3.4.29.7 Answer, comments

If the maximum number of items for one transaction is returned, the query should be repeated with incremented segment number.

Date, Created
Time, Created

Creation date and time for rectify trade request.

Date, As Of
Time, As Of

Match date and time for trade subject to rectify.

Clearing Date

Clearing date for processing of rectify transaction.

Clearing Date, Original

Original Clearing date for processing of trade subject to rectify.

TRADING_CODE

Identifies user submitting the rectify trade request.

USER

Identifies user confirming or rejecting the rectify trade request.

Series

Instrument in trade subject to rectify trade request.

Trade Number

Together with instrument type of traded seres, Trade Number identifies the trade subject to rectify trade request.

Rectify Trade Number

Together with instrument type of traded seres, Rectify Trade Number identifies the rectify trade request.

External Clearing House, Sequence Number

sequence number provided by external exchange system, optional.

State

returns current state of reqeust: Holding, Active or Rejected.

Bought or Sold

indicates whether trade corresponds to bought or sold contracts.

Reserved Properties

Not applicable.

NEW_ACCOUNT

New account for trade - set to "*" if trade is moved to several accounts.

ACCOUNT

account into which trade is allocated prior to rectify operation.

Quantity, Trade

quantity in trade subject to rectify.

Price, Deal

price in trade subject to rectify.

3.4.30 CQ15 [Detailed Holding Rectify Trade QUERY]

3.4.30.1 Fingerprint

QUERY properties	
transaction type	CQ15
calling sequence	omniapi_query_ex
struct name	query_rectify_t_cont
facility	EP3
partitioned	false
answers	CA15

ANSWER properties	
transaction type	CA15
struct name	answer_rectify_ext_cont
segmented	false

3.4.30.2 Related Messages

CQ14, CC11

3.4.30.3 Purpose

This query is used for receiving detailed information about a holding rectify trade.

3.4.30.4 Structure

The CQ15 QUERY has the following structure:

```
struct query_rectify_t_cont {
   struct transaction_type
   struct series  // Named struct no: 50000
   INT32_T rectify_trade_number_i  // Rectify Trade Number
}
```

3.4.30.5 Usage and conditions

To use this query the rectify trade number must be used. It can be listed in Query to get rectified trades that are in holding state.

Use CQ14 to obtain rectify trade number to be supplied as query parameter when using CQ15. Use CC11 to confirm or rejcet the request to rectify the trade.

Series

must be completed with **Country Number**, **Market Code** and **Instrument Group**.

3.4.30.6 Answer Structure

The CA15 ANSWER has the following structure:

```
struct answer_rectify_ext_cont {
   struct transaction_type
   UINT16_T items_n  // Items
   char[2] filler_2_s  // Filler
   Array ITEM [max no: 100] {
      struct account
      INT64_T trade_quantity_i  // Quantity, Trade
      UINT8_T open_close_req_c  // Open  Close Request
      char[15] customer_info_s  // Customer, Information
   }
}
```

3.4.31 CQ19 [Account Propagation QUERY]

3.4.31.1 Fingerprint

QUERY properties	
transaction type	CQ19
calling sequence	omniapi_query_ex
struct name	query_account_prop
facility	EP5
partitioned	false
answers	CA19

ANSWER properties	
transaction type	CA19
struct name	answer_propagate
segmented	false

3.4.31.2 Purpose

This transaction retrieves information regarding all account propagations connected to a specified account. Note that the specified account must be owned by the querying customer and that this account must be fully specified.

3.4.31.3 Structure

The CQ19 QUERY has the following structure:

```
struct query_account_prop {
   struct transaction_type
   struct series  // Named struct no: 50000
   struct account
   UINT16_T segment_number_n  // Segment Number
   char[2] filler_2_s  // Filler
}
```

3.4.31.4 Usage and conditions

Series

is not relevant in this query. It has, however, to be set to zero.

Segment Number

is one for the first query and then incremented.

Account

identifies the account for which propagations are to be returned in the answer

3.4.31.5 Answer Structure

The CA19 ANSWER has the following structure:

```
struct answer_propagate {
   struct transaction_type
   UINT16_T segment_number_n  // Segment Number
   UINT16_T items_n  // Items
   Array ITEM [max no: 100] {
      struct account
      UINT8_T prop_type_c  // Type of Propagation
      char[3] filler_3_s  // Filler
   }
}
```

3.4.31.6 Answer, comments

If the maximum number of items for one transaction is returned, the query should be repeated with incremented segment number.

3.4.32 CQ21 [Pending Exercise Request QUERY]

3.4.32.1 Fingerprint

QUERY properties	
transaction type	CQ21
calling sequence	omniapi_query_ex
struct name	query_exercise_req
facility	EP3
partitioned	true
answers	CA21

ANSWER properties	
transaction type	CA21
struct name	answer_exercise_req
segmented	false

3.4.32.2 Related Messages

CC15

3.4.32.3 Purpose

The purpose of this query is to retrieve all pending exercise requests. Use CC15 to either confirm or reject the pending exercise request.

3.4.32.4 Structure

The CQ21 QUERY has the following structure:

```
struct query_exercise_req {
   struct transaction_type
   struct series  // Named struct no: 50000
   struct search_series
   struct account
   UINT16_T segment_number_n  // Segment Number
   char[2] filler_2_s  // Filler
}
```

3.4.32.5 Usage and conditions

Series

must be completed with **Country Number**, **Market Code** and **Instrument Group**.

Segment Number

is one for the first query and then incremented.

Search Series
Account

identify the pending exercise requests for which data is to be returned in the answer.

3.4.32.6 Answer Structure

The CA21 ANSWER has the following structure:

```
struct answer_exercise_req {
   struct transaction_type
   struct partition_low
   struct partition_high
   UINT16_T segment_number_n  // Segment Number
   UINT16_T items_n  // Items
   Array ITEM [max no: 250] {
      struct series  // Named struct no: 50000
      struct account
      CHAR reserved_1_c  // Reserved
      char[2] reserved_2_s  // Reserved
```

```
        CHAR filler_1_s  // Filler
        struct trading_code
        struct ex_user_code
        char[8] modified_date_s  // Date, Modified
        char[6] modified_time_s  // Time, Modified
        char[8] asof_date_s  // Date, As Of
        char[6] asof_time_s  // Time, As Of
        INT64_T quantity_i  // Quantity
        INT32_T trade_number_i  // Trade Number
        INT32_T exercise_number_i  // Exercise, Request Number
        UINT8_T state_c  // State
        char[3] filler_3_s  // Filler
    }
}
```

3.4.32.7 Answer, comments

If the maximum number of items for one transaction is returned, the query should be repeated with incremented segment number.

3.4.33 CQ22 [Error Message QUERY]

3.4.33.1 Fingerprint

QUERY properties	
transaction type	CQ22
calling sequence	omniapi_query_ex
struct name	query_error_msg
facility	EP5
partitioned	false
answers	CA22

ANSWER properties	
transaction type	CA22
struct name	answer_error_msg
segmented	true

3.4.33.2 Related Messages

BD6

3.4.33.3 Purpose

The purpose of this transaction is to retrieve possible error information. Typical information could be regarding trades or exercise requests that are invalid due to having been put on non-existing accounts.

3.4.33.4 Structure

The CQ22 QUERY has the following structure:

```
struct query_error_msg {
   struct transaction_type
   struct series  // Named struct no: 50000
   struct search_series
   struct account
   UINT32_T error_id_u  // Error Identity
   UINT16_T segment_number_n  // Segment Number
   char[8] from_date_s  // Date, From
   char[8] to_date_s  // Date, To
   char[6] from_time_s  // Time, From
   char[6] to_time_s  // Time, To
   char[2] filler_2_s  // Filler
}
```

3.4.33.5 Usage and conditions

This query is used when the Attention field, in any trade-related information received, contains a non-zero value. Detailed information is avalible in the Dedicated Trade Information Transaction.

This query should contain either an Error identity or a range in time including date. The time range

is expressed in the system time, which normally is identical to the local time at the exchange.

Series

must be completed with Country Number, Market Code and Instrument Group.

Segment Number

is one for the first query and then incremented.

3.4.33.6 Answer Structure

The CA22 ANSWER has the following structure:

```
struct answer_error_msg {
   struct transaction_type
   struct partition_low
   struct partition_high
   UINT16_T segment_number_n  // Segment Number
   UINT16_T items_n  // Items
   Array ITEM [max no: 100] {
      struct trading_code
      struct series  // Named struct no: 50000
      struct account
      char[8] created_date_s  // Date, Created
      char[6] created_time_s  // Time, Created
      char[10] error_operation_s  // Error, Operation
      UINT32_T error_id_u  // Error Identity
      char[40] error_problem_s  // Error, Problem
```

```
    }
}
```

3.4.33.7 Answer, comments

If the maximum number of items for one transaction is returned, the query should be repeated with incremented segment number.

3.4.34 CQ31 [Simulate Fee QUERY]

3.4.34.1 Fingerprint

QUERY properties	
transaction type	CQ31
calling sequence	omniapi_query_ex
struct name	query_simulate_fee
facility	EP3
partitioned	false
answers	CA31

ANSWER properties	
transaction type	CA31
struct name	answer_delivery
segmented	false

3.4.34.2 Purpose

This query calculates the fees for a particular trade. The fees are returned as delivery information (see Answer below).

3.4.34.3 Structure

The CQ31 QUERY has the following structure:

```
struct query_simulate_fee {
   struct transaction type
   struct series  // Named struct no: 50000
   INT32_T deal_price_i  // Price, Deal
   INT64_T deal_quantity_i  // Quantity, Deal
   struct account
   UINT8_T bid_or_ask_c  // Bid or Ask
   UINT8_T open_close_req_c  // Open  Close Request
   char[2] filler_2_s  // Filler
}
```

3.4.34.4 Usage and conditions

Series
Price, Deal
Quantity, Deal
Account
Bid or Ask
Open Close Request

define the characteristics of the trade and must be specified in order for the central system to be able to calculate the fee data

3.4.34.5 Answer Structure

The CA31 ANSWER has the following structure:

```
struct answer_delivery {
   struct transaction_type
   struct partition_low
   struct partition_high
   UINT16_T segment_number_n  // Segment Number
   UINT16_T items_n  // Items
   Array ITEM [max no: 100] {
      char[8] date_s  // Date
      INT32_T event_type_i  // Stimuli Event
      struct series  // Named struct no: 50000
      struct account
      INT32_T class_no_i  // Class Number
      INT64_T deliv_base_quantity_q  // Quantity, Delivery Base
      char[8] settlement_date_s  // Date, Settlement
      INT64_T delivery_quantity_q  // Quantity, Delivery
      struct deliv_base
   }
}
```

3.4.34.6 Answer, comments

Quantity, Delivery Base

identifies the number of **Delivery Base** to deliver/receive. The sign is set from the clearinghouse's point of view (i.e. is delivered from the clearinghouse). The number of decimals used in the Quantity, Delivery Base is specified by the decimals in price in the Query Underlying Transaction, see**DQ4** (referring to the **Delivery Base**).

Delivery Base

identifies what to deliver.

In the answer Quantity, Delivery Base and Quantity, Delivery is summarized per Date; Event Type; Series; Customer; Account; Class Number; Date, Settlement; and Delivery Base.

3.4.35 CQ36 [Average Price Trade QUERY]

3.4.35.1 Fingerprint

QUERY properties	
transaction type	CQ36
calling sequence	omniapi_query_ex
struct name	query_average_price_trade
facility	EP5
partitioned	false
answers	CA36

ANSWER properties	
transaction type	CA36
struct name	answer_average_price_trade
segmented	false

3.4.35.2 Purpose

This query returns the trade number of the trades that are part of an average price trade.

3.4.35.3 Structure

The CQ36 QUERY has the following structure:

```
struct query_average_price_trade {
   struct transaction_type
   struct series  // Named struct no: 50000
   UINT16_T segment_number_n  // Segment Number
   char[2] filler_2_s  // Filler
   INT32_T trade_number_i  // Trade Number
}
```

3.4.35.4 Usage and conditions

Series

must be completed with **Country Number**, **Market Code** and **Instrument Group**.

Segment Number

is one for the first query and then incremented.

Trade Number

identifies the trade, for which data is to be retrieved.

3.4.35.5 Answer Structure

The CA36 ANSWER has the following structure:

```
struct answer_average_price_trade {
   struct transaction_type
   struct series  // Named struct no: 50000
   struct partition_low
   struct partition_high
   UINT16_T segment_number_n  // Segment Number
   UINT16_T items_n  // Items
   Array ITEM [max no: 1000] {
      INT32_T trade_number_i  // Trade Number
      INT64_T quantity_i  // Quantity
   }
}
```

3.4.35.6 Answer, comments

If the maximum number of items for one transaction is returned, the query should be repeated with incremented segment number.

3.4.36 CQ38 [Account QUERY]

3.4.36.1 Fingerprint

QUERY properties	
transaction type	CQ38
calling sequence	omniapi_query_ex
struct name	query_account
facility	EP5
partitioned	false
answers	CA38

ANSWER properties	
transaction type	CA38
struct name	answer_account_ext
segmented	true

3.4.36.2 Purpose

The purpose of this query is to retrieve account information for own accounts.

3.4.36.3 Structure

The CQ38 QUERY has the following structure:

```
struct query_account {
   struct transaction_type
   struct series  // Named struct no: 50000
   struct account
   UINT16_T segment_number_n  // Segment Number
   UINT8_T query_on_date_c  // Query on Date
   char[8] date_s  // Date
   CHAR filler_1_s  // Filler
}
```

3.4.36.4 Usage and conditions

Series

is not relevant in this query. However, it has to be set to zero.

Segment Number

is one for the first query and then incremented.

A query can be done using three methods:

1. Using Account string as search string. This can be achieved by filling in Country, Customer and Account id with explicit values. The answer is one account.
2. Using Account string as wildcard search string. This can be achieved by filling in Country and Customer with explicit values, or wildcards, and Account id with account id = "*". The answer contains all accounts.
3. Using Date as search criteria. The answer contains all accounts modified since the Business Date given. The field Query on Date must be set to true.

3.4.36.5 Answer Structure

The CA38 ANSWER has the following structure:

```
struct answer_account_ext {
   struct transaction_type
   UINT16_T segment_number_n  // Segment Number
   UINT16_T items_n  // Items
   Array ITEM [max no: 160] {
      struct account_data
   }
}
```

3.4.36.6 Answer, comments

If the maximum number of items for one transaction is returned, the query should be repeated with incremented segment number.

3.4.37 CQ39 [Trade Change QUERY QUERY]

3.4.37.1 Fingerprint

QUERY properties	
transaction type	CQ39
calling sequence	omniapi_query_ex
struct name	query_missing_trade_change
facility	EP3
partitioned	false
answers	CA39

ANSWER properties	
transaction type	CA39
struct name	answer_missing_trade_change
segmented	false

3.4.37.2 Related Messages

CQ10, BD39

3.4.37.3 Purpose

The purpose of this query is to retrieve missing trade change broadcasts.

3.4.37.4 Structure

The CQ39 QUERY has the following structure:

```
struct query_missing_trade_change {
   struct transaction_type
   struct series  // Named struct no: 50000
   UINT8_T instance_c  // Instance, Number
   char[3] filler_3_s  // Filler
   INT32_T sequence_first_i  // Number, First Sequential
   INT32_T sequence_last_i  // Number, Last Sequential
   char[8] date_s  // Date
}
```

3.4.37.5 Usage and conditions

The query is intended to be used when a sequence number gap is detected or after login to read trade changes already done.

The sequence of events at startup is to first query for trades (CQ10) and then query for trade changes (CQ39).

3.4.37.6 Answer Structure

The CA39 ANSWER has the following structure:

```
struct answer_missing_trade_change {
   struct transaction_type
   char[2] filler_2_s  // Filler
   UINT16_T items_n  // Items
   Array ITEM [max no: 1000] {
      struct cl_trade_change_api
   }
}
```

3.4.38 CQ40 [Auxiliary position info updated QUERY]

3.4.38.1 Fingerprint

QUERY properties	
transaction type	CQ40
calling sequence	omniapi_query_ex
struct name	query_updated_pos_info
facility	EP3
partitioned	true
answers	CA40

ANSWER properties	
transaction type	CA40
struct name	answer_updated_pos_info
segmented	true

3.4.38.2 Related Messages

BD40, CQ3

3.4.38.3 Purpose

This query is used for retrieving auxiliary information associated with positions that have been updated since a specified date and time.

3.4.38.4 Structure

The CQ40 QUERY has the following structure:

```
struct query_updated_pos_info {
```

```
   struct transaction_type
   struct series  // Named struct no: 50000
   struct search_series
   struct account
   UINT16_T segment_number_n  // Segment Number
   char[8] modified_date_s  // Date, Modified
   char[6] modified_time_s  // Time, Modified
}
```

3.4.38.5 Usage and conditions

The query is intended to be used after login for recovering from any missed BD40 broadcasts while the API-client was disconnected.

The auxiliary information consists of :

- quantity to be exempted from automatic/general exercise (deny exercise)
- quantity closed-out current clearing date
- quantity of underlying used as cover for short position (exchange specific)

Series

must be complete up to Country Number, Market Code and Instrument Group.

Segment Number

is one for the first query and then incremented.

Search Series Account

identifies the positions for which auxiliary information is to be returned in the answer.

3.4.38.6 Answer Structure

The CA40 ANSWER has the following structure:

```
struct answer_updated_pos_info {
   struct transaction_type
   struct partition_low
   struct partition_high
   UINT16_T segment_number_n  // Segment Number
   UINT16_T items_n  // Items
   Array ITEM [max no: 900] {
      struct pos_info_update_api
   }
}
```

3.4.38.7 Answer, comments

If the maximum number of items for one transaction is returned, the query should be repeated with incremented segment number.

3.4.39 CQ51 [DC Holding Trade QUERY]

3.4.39.1 Fingerprint

QUERY properties	
transaction type	CQ51
calling sequence	omniapi_query_ex
struct name	query_trade_dc
facility	EP8
partitioned	false
answers	CA51

ANSWER properties	
transaction type	CA51
struct name	answer_trade_dc
segmented	true

3.4.39.2 Related Messages

BD41, MO75, MO76, MO459

3.4.39.3 Purpose

This query retrieves information about trade reports in holding state awaiting confirmation by the clearinghouse for subsequent entry of the trade into the clearing system. The query can also be used to retrieve information about already confirmed trade reports or rejected trade reports.

3.4.39.4 Structure

The CQ51 QUERY has the following structure:

```
struct query_trade_dc {
   struct transaction_type
   struct series  // Named struct no: 50000
   struct account
   char[8] from_date_s  // Date, From
   char[8] to_date_s  // Date, To
   char[6] from_time_s  // Time, From
   char[6] to_time_s  // Time, To
   UINT16_T segment_number_n  // Segment Number
   UINT8_T dc_deal_state_c  // State, Deal
   CHAR filler_1_s  // Filler
}
```

3.4.39.5 Usage and conditions

The clearinghouse may configure that some Trade Report Types used in MO75, MO76, and/or MO459 shall put the resulting trades in Holding State awaiting confirmation by the clearinghouse. This query is used for retrieving informtion about such trades

.

Series

Currently not used—should be zeroed.

Account

Can contain explicit value or wildcard.

Date, From, Time, From, Date, To, Time, To

Specify a time interval when the retrieved trade was created. Only trade reports created the current business day may be retrieved.

Segment Number

Set to 1 for first query and then incremented, if necessary, for retrieval of subsequent segments of the total response. Segment number returned in final response segment is set to 0.

DC Deal State

Must be filled with either Normal, Rejected, or Holding Matched.

3.4.39.6 Return Codes

After a successful CQ51 query, a list of holding trade reports awaiting clearinghouse confirmation is returned to the sender.

A CQ51 transaction may also be aborted. In that case, only the reason for the transaction being aborted is returned to the sender.

3.4.39.7 Answer Structure

The CA51 ANSWER has the following structure:

```
struct answer_trade_dc {
   struct transaction type
   struct series  // Named struct no: 50000
   struct partition low
   struct partition high
   UINT16_T segment number n  // Segment Number
   UINT16_T items n  // Items
   Array ITEM [max no: 180] {
      INT32_T deal number i  // Deal Number
      struct series  // Named struct no: 50000
      INT32_T deal price i  // Price, Deal
      UINT8_T dc deal state c  // State, Deal
      UINT8_T account validation c  // Account Validation
```

```
char[2] filler_2_s  // Filler
struct deal_part {
   INT64_T timestamp_log_q  // Timestamp, Last Change
   INT64_T settlement_date_q  // Date, Settlement
   INT64_T time_of_agreement_q  // Time Of Agreement
   INT32_T deal_price_i  // Price, Deal
   INT64_T deal_quantity_i  // Quantity, Deal
   UINT8_T deal_source_c  // Deal Source
   UINT8_T state_c  // State
   char[5] broker_id_s  // Broker, Identity
   UINT8_T client_category_c  // Client Category
   UINT8_T aggressive_c  // Aggressive
   UINT8_T external_fee_type_c  // External Fee Type
   UINT16_T state_number_n  // Trading State Number
   UINT16_T trade_condition_n  // Trade Condition
   UINT8_T combo_source_c  // Combination matching source
   UINT8_T combo_trade_seq_c  // Combo Trades Sequence Number
   UINT8_T trade_venue_c  // Trade venue
   CHAR filler_1_s  // Filler
   UINT16_T eqy_combo_trade_seq_n  // Equity Combo Trade, Counter
   UINT16_T eqy_combo_trade_tot_n  // Equity Combo Trade, Total Value
  UINT16_T eqy_combo_trade_pos_n  // Equity Combo Trade, Trade Position
   struct cl_order_record {
      INT64_T timestamp_in_q  // Timestamp In
      QUAD_WORD order_number_u  // Order Number
      struct party
      struct cl_order {
         struct series  // Named struct no: 50000
         struct trading_code
         struct cl_order_var {
            INT64_T cl_quantity_i  // CL Quantity
            INT32_T premium_i  // Premium
            UINT32_T block_n  // Block Size
            UINT16_T time_validity_n  // Validity Time
            UINT16_T exch_order_type_n  // Order Type, Exchange
            char[10] ex_client_s  // Client
            char[15] customer_info_s  // Customer, Information
            UINT8_T open_close_req_c  // Open  Close Request
            UINT8_T bid_or_ask_c  // Bid or Ask
            UINT8_T ext_t_state_c  // Trade Report Type
            UINT8_T order_type_c  // Order Type
           UINT8_T outside_info_spread_c  // Outside Information Spread
            char[2] filler_2_s  // Filler
         }
         struct ex_user_code
         struct give_up_member  // Named struct no: 50002
         char[32] exchange_info_cl_s  // Exchange Information
         UINT16_T transaction_number_n  // Transaction Type Number
         char[2] filler_2_s  // Filler
      }
      INT64_T total_volume_i  // Total Volume
      INT64_T display_quantity_i  // Quantity, Display
      INT64_T orig_total_volume_i  // Total Volume, Original
      INT64_T orig_shown_quantity_i  // Shown Quantity, Original
   }
   struct match_id
```

```
            struct combo_series
            INT32_T combo_deal_price_i  // Combo deal price
        }
    }
}
```

3.4.39.8 Answer, comments

If the maximum number of items for one transaction is returned, the query should be repeated with incremented segment number.

3.4.40 CQ52 [Delivery QUERY]

3.4.40.1 Fingerprint

QUERY properties	
transaction type	CQ52
calling sequence	omniapi_query_ex
struct name	query_missing_delivery
facility	EP3
partitioned	false
answers	CA52

ANSWER properties	
transaction type	CA52
struct name	answer_missing_delivery
segmented	false

3.4.40.2 Related Messages

BD18, CQ53

3.4.40.3 Purpose

This query retrieves deliveries. For example, if a missing sequence number is detected for the Delivery Dedicated broadcast (BD18), this query is used to get synchronized with the broadcast flow again.

3.4.40.4 Structure

The CQ52 QUERY has the following structure:

```
struct query_missing_delivery {
   struct transaction_type
   struct series  // Named struct no: 50000
```

```
   INT32_T sequence_first_i  // Number, First Sequential
   INT32_T sequence_last_i  // Number, Last Sequential
   char[8] date_s  // Date
}
```

3.4.40.5 Usage and conditions

This transaction retrieves deliveries for the current business day, to query for historical deliveries, use CQ53.

Series

must be completed with **Country Number**, **Market Code** and **Instrument Group**.

Number, first sequential

is the first missing one.

Number, last sequential

is the last missing one. If the Number, last sequential is equal to zero, all available deliveries are sent in sequence.

Date

must hold the current Clearing date for the Instrument Type in question.

Class Number

is a number indicating type of settlement for a delivery item.

3.4.40.6 Answer Structure

The CA52 ANSWER has the following structure:

```
struct answer_missing_delivery {
   struct transaction_type
   char[2] filler_2_s  // Filler
   UINT16_T items_n  // Items
   Array ITEM [max no: 280] {
      struct cl_delivery_api
   }
}
```

3.4.40.7 Answer, comments

If the maximum number of items for one transaction is returned, the query should be repeated with Number, First sequential set to the next missing sequence number after the Sequence Number of the last received item.

Apart from the header each record in the response contains the same information as the **directed_delivery** struct in the Delivery Dedicated broadcast (BD18).

Date

must hold the current business date.

3.4.41 CQ53 [Delivery History QUERY]

3.4.41.1 Fingerprint

QUERY properties	
transaction type	CQ53
calling sequence	omniapi_query_ex
struct name	query_api_delivery
facility	EP5
partitioned	true
answers	CA53

ANSWER properties	
transaction type	CA53
struct name	answer_api_delivery
segmented	false

3.4.41.2 Related Messages

BD18, CQ52

3.4.41.3 Purpose

This query retrieves historical deliveries. The information is available to the trading member and the clearing member the next trading day. To retrieve deliveries for the current trading day, use CQ52.

3.4.41.4 Structure

The CQ53 QUERY has the following structure:

```
struct query_api_delivery {
   struct transaction type
   struct series  // Named struct no: 50000
   char[8] from_date_s  // Date, From
   INT32_T sequence_first_i  // Number, First Sequential
   char[8] to_date_s  // Date, To
   INT32_T sequence_last_i  // Number, Last Sequential
}
```

3.4.41.5 Usage and conditions

The historical delivery information is available to the members the next business day and is queried per instrument type.

Series

must be completed with **Country Number**, **Market Code** and **Instrument Group**.

Date, From
Date, To

must be Clearing Dates that are historical dates compared to current Clearing date. **Date, From** must be less or equal to **Date, To**.

Number, first sequential

is the first item to get for **Date, From**. Zero or one means the first item for that date.

Number, last sequential

is the last item to get for **Date, To**. Zero means the last item for that date.

Class Number

is a number indicating type of settlement for a delivery item.

3.4.41.6 Answer Structure

The CA53 ANSWER has the following structure:

```
struct answer_api_delivery {
   struct transaction_type
   struct series  // Named struct no: 50000
   char[8] from_date_s  // Date, From
   INT32_T sequence_first_i  // Number, First Sequential
   UINT16_T items_n  // Items
   char[2] filler_2_s  // Filler
   Array ITEM [max no: 280] {
      struct cl_delivery_api
   }
}
```

3.4.41.7 Answer, comments

Date

contains the date on which this delivery was created.

Apart from the header each record in the response contains the same information as the **directed_delivery** struct in the Delivery Dedicated broadcast (BD18).

If all deliveries that reside centrally are to be fetched, the following sequence must be performed:

Loop for all instrument types defined, except for country = 255, market = 255 and instrument group = 255.

For each instrument type, do a CQ53 query until CA53 signals that no more deliveries exist.

The first CQ53 is filled with the following parameters:

- Series, filled with current instrument type.
- Date, From. Set to ‘19000101’.
- Sequence Number 1. Set to 1.
- Date, To. Set to yesterday’s date.
- Sequence Number 2. Set to 0.

If Number, first sequential in CA53 is greater than zero, more CQ53 queries must be done to retrieve data. CQ53 must be filled with the following parameters:

- Series, filled with series in CA53.
- Date, From. Filled with Date, From in CA53.
- Sequence Number 1. Filled with Sequence Number 1 in CA53.
- Date, To. Set to yesterday’s date.
- Sequence Number 2. Set to 0.

3.4.42 CQ61 [Holding Give Up Request QUERY]

3.4.42.1 Fingerprint

QUERY properties	
transaction type	CQ61
calling sequence	omniapi_query_ex
struct name	query_give_up_request
facility	EP3
partitioned	false
answers	CA61

ANSWER properties	
transaction type	CA61
struct name	answer_give_up_request
segmented	true

3.4.42.2 Related Messages

CC38

CC40

BD29

CQ76

CQ77

3.4.42.3 Purpose

The query returns Give-up requests in a holding state, but may also return Give-up requests in other states depending on the query criteria (see below). The answer contains information to facilitate the tracking of give-ups and their origins.

3.4.42.4 Structure

The CQ61 QUERY has the following structure:

```
struct query_give_up_request {
   struct transaction_type
   struct series  // Named struct no: 50000
   struct party
   UINT32_T ext_trade_number_u  // Trade Number, External
   UINT16_T segment_number_n  // Segment Number
   UINT8_T state_c  // State
   CHAR buy_or_sell_c  // Buy or Sell
   UINT8_T send_or_receive_c  // Send or Receive
   char[8] created_date_s  // Date, Created
   char[32] series_id_s  // Series, Identity
   char[2] country_id_s  // Name, Country
   char[5] ex_customer_s  // Customer, Identity
   char[30] give_up_text_s  // Give Up, Free Text
   char[2] filler_2_s  // Filler
}
```

3.4.42.5 Usage and conditions

> **Note:** It is recommended to use BD29/CQ76 instead of CQ61.

Facility EP3 should be used for current date and facility EP5 for historic dates.

The query is only partitioned when used on facility EP3.

Use CC38 to confirm or reject a Give-up request.

Series

must be complete up to **Country Number**, **Market Code** and **Instrument Group**. Determines clearing partition when querying for current business date on facility EP3.

Date, Created

must be filled with the business date when the Give-up request was created.

Segment Number

should be set to 1 for retrieving the first answer segment from a partition and then incremented for retrieval of subsequent answer segments.

State

has the following impact on the returned give-up requests in the answer:

0	all give-ups are returned regardless of state
1	Holding
5	Completed
6	Rejected

Series Id

should contain an explicit series name or a series wildcard string.

Send or Receive

defines the interpretation of the member (**Name, Country** and **Customer, Identity**) and **Party** field.

When set to '1' (send), the member field is used for filtering of the participant initiating the **Give-Up** and the **Party** fields are used for filtering the receiving/destination member for the give-up.

If set to '2' (receive), the member field is used for filtering of the participant receiving **Give-Up** and the **Party** fields are used used for filtering the member initiating the give-up.

Country, Name and Customer Identity

specifies give-up/take-up member (participant id) for filtering give-up.

Wildcard search/filtering can be used. Must be filled with "*", "*" when doing a wildcard search

Party

specifies take-up/give-up member (participant id) for filtering give-up.

Wildcard search/filtering can be used. Must be filled with "*", "*" when doing a wildcard search.

Buy or Sell

allows for filtering on give-ups on buy (1) or sell (2) trades.

Filtering will not be applied if set to 0.

Give Up, Free Text

allows searching for give-up(s) with specified "Free text".

Wildcard search/filtering can be used. Must be set to "*" when doing a wildcard search.

Trade Number, External

allows searching for give-up(s) on trade(s) with specified external trade number.

External trade number on trades is not used by all exchanges.

Must be set to 0 when doing a wildcard search.

3.4.42.6 Answer Structure

The CA61 ANSWER has the following structure:

```
struct answer_give_up_request {
```

```
    struct transaction_type
    struct partition_low
    struct partition_high
    UINT16_T segment_number_n  // Segment Number
    UINT16_T items_n  // Items
    Array ITEM [max no: 420] {
       struct series  // Named struct no: 50000
       struct account
       struct party
       INT32_T give_up_number_i  // Give Up, Number
       INT64_T trade_quantity_i  // Quantity, Trade
       INT32_T deal_price_i  // Price, Deal
       INT32_T trade_number_i  // Trade Number
       INT32_T commission_i  // Commission
       UINT8_T bought_or_sold_c  // Bought or Sold
       UINT8_T state_c  // State
       char[8] created_date_s  // Date, Created
       char[6] created_time_s  // Time, Created
       char[30] give_up_text_s  // Give Up, Free Text
       char[8] asof_date_s  // Date, As Of
       char[6] asof_time_s  // Time, As Of
       char[8] orig_clearing_date_s  // Clearing Date, Original
       UINT8_T old_trade_c  // Old Trade Indicator
       CHAR ext_trade_fee_type_c  // External Trade, Fee Type
       UINT8_T deal_source_c  // Deal Source
       UINT8_T reserved_prop_c  // Reserved Properties
       char[8] clearing_date_s  // Clearing Date
       UINT32_T ext_trade_number_u  // Trade Number, External
       UINT32_T orig_ext_trade_number_u  // Trade Number, Original External
    }
 }
```

3.4.42.7 Answer, comments

Account

describes the destination member in the giveup. The 10 last characters may be left blank, thus only defining the member, or set to point out a specific account.

Party

identifies the customer that gives up the trade.

Deal source

data refer to the original trade’s deal source. Please refer to the detailed field descriptions for further information.

The following fields describe the trade that is subject to the giveup:

- Series
- Party
- Bought or Sold
- Quantity, Trade

- Price, Deal
- Trade Number
- Date, Created
- Time, Created
- Date, As Of
- Time, As Of
- Original Clearing Date
- Old Trade Indicator
- Deal Source
- External Trade Fee Type
- Trade Number, External
- Original Trade Number, External

The Quantity, Trade field specifies the give-up portion of the trade.

Of these, Date, As Of; Time, As Of; Original Clearing Date; Old Trade Indicator; Deal Source; External Trade Fee Type only contain significant data for give-up requests made the current business day and whose states are either holding or completed.

Give-Up Number; State; Account; Give-Up Free Text, Clearing Date are fields that describe the giveup. Clearing Date is the clearing date of the giveup itself.

If the maximum number of items for one transaction is returned, the query should be repeated with incremented segment number.

3.4.43 CQ62 [Confirm Give Up Request QUERY]

3.4.43.1 Fingerprint

QUERY properties	
transaction type	CQ62
calling sequence	omniapi_query_ex
struct name	query_conf_give_up_req_items
facility	EP5
partitioned	false
answers	CA62

ANSWER properties	
transaction type	CA62
struct name	answer_conf_give_up_req_items
segmented	false

3.4.43.2 Related Messages

CC38, CQ61

3.4.43.3 Purpose

This query returns the give-up items sent when a giveup was confirmed. This query can only be sent for a confirmed giveup.

3.4.43.4 Structure

The CQ62 QUERY has the following structure:

```
struct query_conf_give_up_req_items {
   struct transaction_type
   struct series  // Named struct no: 50000
   INT32_T give_up_number_i  // Give Up, Number
}
```

3.4.43.5 Usage and conditions

Use CQ61 to query for Give-up requests in holding state.

Use CC38 to reject or confirm holding Give-up requests.

Series

must contain the whole series for the giveup.

Give up number

identifies the give-up.

3.4.43.6 Answer Structure

The CA62 ANSWER has the following structure:

```
struct answer_conf_give_up_req_items {
   struct transaction_type
   UINT16_T items_n  // Items
   char[2] filler_2_s  // Filler
   Array ITEM [max no: 50] {
      struct account
      INT64_T trade_quantity_i  // Quantity, Trade
      UINT8_T open_close_req_c  // Open  Close Request
      char[15] customer_info_s  // Customer, Information
   }
}
```

3.4.43.7 Answer, comments

This information is the same information as sent in the Confirm Give-Up Trade Transaction, see **CC38**.

3.4.44 CQ65 [Level Position QUERY]

3.4.44.1 Fingerprint

QUERY properties	
transaction type	CQ65
calling sequence	omniapi_query_ex
struct name	query_pos_level
facility	EP3
partitioned	true
answers	CA65

ANSWER properties	
transaction type	CA65
struct name	answer_position
segmented	true

3.4.44.2 Related Messages

CQ3

3.4.44.3 Purpose

The purpose of this transaction is to allow for members and clearinghouse personell to query for positions on different account levels. The positions are grouped according to their origin (e.g. Client or House) or their margin account. This allows to query for a firm's total exposure to a series.

Note: Positions will only be retrieved for instruments having the Maintain Positions parameter set to Yes.

3.4.44.4 Structure

The CQ65 QUERY has the following structure:

```
struct query_pos_level {
   struct transaction_type
   struct series  // Named struct no: 50000
   struct account
   char[32] series_id_s  // Series, Identity
   INT32_T summary_i  // Summary
   UINT16_T segment_number_n  // Segment Number
```

```
    char[2] filler_2_s  // Filler
    char[8] date_s  // Date
    char[12] account_type_s  // Account Type
    INT32_T level_type_i  // Level Type
}
```

3.4.44.5 Usage and conditions

Account

If the field Account contains any wildcards, the **Summary** field must be set to1 (yes); the query transaction will otherwise be aborted with an error-status.

Account Type

When filled must either be a valid account type name or a valid wildcard representation of an Account Type name. If Account Type is not blank, only positions on accounts with an Account Type matching the argument is returned in the answer.

Level Type

specifies the account level of interest; origin or margin.

Segment Number

is one for the first query and then incremented.

Series Id

should contain an explicit series name or a series wildcard string.

Summary

specifies whether to return the aggregated positions on the specified account level or if the individual position items are to be returned.

Summary =2 (no) is only applicable if the field **Customer Account**does not contain any wildcards, i.e. it identifies a single account. In that case, one may retrieve all the individual ‘position items’ making up the aggregated (and “propagated”) position on a margin or origin account.

Date

must be valid and have one of the following values:

- Previous calendar date: The overnight (O/N) position is returned. These positions are static during the day.
- Today’s business date. The current position for the current clearing date (provided it exists for the instrument) is returned.
- Next calendar date. The current position for the next clearing date is returned; trades as of next clearing date are added to the current clearing date position.

Note that the previous and next calendar date is in relation to current business date in the system. For example, the previous calendar date will refer to a Sunday when current business date is a Monday.

This query is used when the account structure makes it relevant to ask for Origin Level and Margin Level accounts. Use Position Information Transaction, see **CQ3**, for an ordinary account level query.

3.4.44.6 Answer Structure

The CA65 ANSWER has the following structure:

```
struct answer_position {
   struct transaction_type
   struct partition_low
   struct partition_high
   UINT16_T segment_number_n  // Segment Number
   UINT16_T items_n  // Items
   Array ITEM [max no: 500] {
      struct series  // Named struct no: 50000
      char[8] modified_date_s  // Date, Modified
      char[6] modified_time_s  // Time, Modified
      UINT8_T reserved_prop_c  // Reserved Properties
      CHAR filler_1_s  // Filler
      INT64_T nbr_held_q  // Held
      INT64_T nbr_written_q  // Written
      INT64_T deny_exercise_q  // Deny Exercise
      struct account
      UINT32_T quantity_cover_u  // Quantity Cover
      INT64_T qty_closed_out_q  // Quantity, Closed out
   }
}
```

3.4.44.7 Answer, comments

Quantity, Cover

states the quantity of underlying equity that is used as cover for this position. This field is normally set to zero. Only if the query's **Date** was set to Today's calendar date can this field have a non-zero value.

The response is structured the same way as is CA3.

If the maximum number of items for one transaction is returned, the query should be repeated with incremented segment number.

3.4.45 CQ68 [Clearing Date QUERY]

3.4.45.1 Fingerprint

QUERY properties	
transaction type	CQ68
calling sequence	omniapi_query_ex

QUERY properties	
struct name	query_clearing_date
facility	EP5
partitioned	false
answers	CA68

ANSWER properties	
transaction type	CA68
struct name	answer_clearing_date
segmented	false

3.4.45.2 Purpose

The purpose of this query is to retrieve information on the current and the next clearing date for instrument types.

3.4.45.3 Structure

The CQ68 QUERY has the following structure:

```
struct query_clearing_date {
   struct transaction_type
   struct series  // Named struct no: 50000
   struct search_series
}
```

3.4.45.4 Usage and conditions

Series, Search

may be zeroed to retrieve clearing date information on all instrument types handled by a particular clearing server.

3.4.45.5 Answer Structure

The CA68 ANSWER has the following structure:

```
struct answer_clearing_date {
   struct transaction_type
   struct partition_low
   struct partition_high
   char[16] omex_version_s  // OMEX Version
   char[8] business_date_s  // Date, Business
   UINT16_T items_n  // Items
   char[2] filler_2_s  // Filler
   Array ITEM [max no: 1000] {
      struct series  // Named struct no: 50000
```

```
        char[8] clearing_date_s  // Clearing Date
        char[8] next_clearing_date_s  // Clearing Date, Next
        char[8] prev_clearing_date_s  // Clearing Date, Previous
        CHAR tra_cl_next_day_c  // Cleared Next Day
        char[3] filler_3_s  // Filler
    }
}
```

3.4.45.6 Answer, comments

Series

is specified to Instrument Type level, i.e. **Country Number**, **Market Code** and **Instrument Group**.

Clearing Date

Please note that the Clearing Date field might be blank in case there is no current clearing date, i.e. the instrument type isn't cleared the current business date. This would typically be the case if some products are not traded or cleared due to a country specific holiday.

The answer received contains information on the preceding, current and following clearing date for a number of instrument types. Each response is prefaced with the transaction type (CA68), the current system version, the current business date in the system and an item field specifying the number of records contained in the response.

3.4.46 CQ71 [Cover Request QUERY]

3.4.46.1 Fingerprint

QUERY properties	
transaction type	CQ71
calling sequence	omniapi_query_ex
struct name	query_cover_req
facility	EP3
partitioned	false
answers	CA71

ANSWER properties	
transaction type	CA71
struct name	answer_cover_req
segmented	false

3.4.46.2 Related Messages

BD24

3.4.46.3 Purpose

This transaction will retrieve all covered call requests made the current business day.

3.4.46.4 Structure

The CQ71 QUERY has the following structure:

```
struct query_cover_req {
   struct transaction_type
   struct series  // Named struct no: 50000
   UINT32_T seq_nbr_1_u  // Sequence Number
   UINT32_T seq_nbr_2_u  // Sequence Number
   char[8] business_date_s  // Date, Business
}
```

3.4.46.5 Usage and conditions

In the unlikely event that a Cover Request Information broadcast is missed this query can be used to synchronize the broadcast flow again.

This query can also be used to get all information at application start-up.

Series

must be complete up to Country Number, Market Code and Instrument Group.

Sequence Number

Sequence number 1 is the first request missing.

Sequence number 2 is the last request missing.

If Sequence number 2 is set to 0, all available requests are returned sequentially.

Date

must contain current business date.

3.4.46.6 Answer Structure

The CA71 ANSWER has the following structure:

```
struct answer_cover_req {
   struct transaction_type
   UINT16_T items_n  // Items
   char[2] filler_2_s  // Filler
   Array ITEM [max no: 500] {
      struct cover_data_item {
         struct series  // Named struct no: 50000
         struct account
         UINT8_T intraday_ind_c  // Intra-day Indicator
         char[3] filler_3_s  // Filler
         UINT32_T quantity_cover_u  // Quantity Cover
```

```
            UINT32_T quantity_tot_cover_u  // Quantity, total cover
            INT32_T quantity_request_i  // Quantity, request
            UINT8_T state_c  // State
            char[2] filler_2_s  // Filler
            CHAR filler_1_s  // Filler
            UINT32_T cover_request_nbr_u  // Cover Request Number
            UINT32_T sequence_nbr_u  // Sequence Number
            char[8] created_date_s  // Date, Created
            char[6] created_time_s  // Time, Created
            char[8] modified_date_s  // Date, Modified
            char[6] modified_time_s  // Time, Modified
            struct user_code
        }
    }
}
```

3.4.46.7 Answer, comments

If the maximum number of items for one transaction is returned, the query should be repeated with Sequence number 1 = set to the next missing sequence number after the Sequence Number of the last received item, and Sequence number 2 = 0.

3.4.47 CQ72 [Open Interest, extended QUERY]

3.4.47.1 Fingerprint

QUERY properties	
transaction type	CQ72
calling sequence	omniapi_query_ex
struct name	query_open_interest_ext
facility	EP5
partitioned	true
answers	CA72

ANSWER properties	
transaction type	CA72
struct name	answer_open_interest_ext
segmented	true

3.4.47.2 Purpose

The purpose of this query is to retrieve the net and gross market open interest per series. The Open Interest information for a series is calculated once a day.

This query is only available for current business date when the signal BI7, Information Type 1 has been sent.

The query can also be used for retrieval of Open Interest for historical dates and can be retrieved anytime. Clearinghouse policy determines for how long Open Interest information is available.

3.4.47.3 Structure

The CQ72 QUERY has the following structure:

```
struct query_open_interest_ext {
   struct transaction_type
   struct series  // Named struct no: 50000
   struct search_series
   UINT16_T segment_number_n  // Segment Number
   char[2] filler_2_s  // Filler
   char[8] date_s  // Date
}
```

3.4.47.4 Usage and conditions

Series

must be completed with **Country Number**, **Market Code**, and **Instrument Group**.

Segment Number

is one for the first query and then incremented.

Search Series

identifies the series for which data is to be returned in the answer.

Date

date for which Open Interest information is to be retrieved

3.4.47.5 Answer Structure

The CA72 ANSWER has the following structure:

```
struct answer_open_interest_ext {
   struct transaction_type
   struct partition_low
   struct partition_high
   UINT16_T segment_number_n  // Segment Number
   UINT16_T items_n  // Items
   Array ITEM [max no: 1000] {
      struct series  // Named struct no: 50000
      UINT64_T gross_open_interest_q  // Gross Open Interest
      UINT64_T net_open_interest_q  // Net Open Interest
      UINT64_T member_net_open_interest_q  // Net Open interest, Member
   }
}
```

3.4.47.6 Answer, comments

If the maximum number of items for one transaction is returned, the query should be repeated with incremented segment number.

3.4.48 CQ73 [Cover Request Update QUERY]

3.4.48.1 Fingerprint

QUERY properties	
transaction type	CQ73
calling sequence	omniapi_query_ex
struct name	query_cover_req_upd
facility	EP3
partitioned	false
answers	CA73

ANSWER properties	
transaction type	CA73
struct name	answer_cover_req_upd
segmented	false

3.4.48.2 Related Messages

BD26

3.4.48.3 Purpose

This transaction will retrieve all covered call requests state updates made the current business day.

3.4.48.4 Structure

The CQ73 QUERY has the following structure:

```
struct query_cover_req_upd {
   struct transaction type
   struct series  // Named struct no: 50000
   UINT32_T seq_nbr_1_u  // Sequence Number
   UINT32_T seq_nbr_2_u  // Sequence Number
   char[8] business_date_s  // Date, Business
}
```

3.4.48.5 Usage and conditions

In the unlikely event that a Cover Request Update broadcast is missed this query can be used to synchronize the broadcast flow again.

This query can also be used to get all information at application start-up.

Series

must be complete up to Country Number, Market Code and Instrument Group.

Sequence Number

Sequence number 1 is the first request missing.

Sequence number 2 is the last request missing.

If Sequence number 2 is set to 0, all available requests are returned sequentially.

Date

must contain current business date.

3.4.48.6 Answer Structure

The CA73 ANSWER has the following structure:

```
struct answer_cover_req_upd {
   struct transaction_type
   UINT16_T items_n  // Items
   char[2] filler_2_s  // Filler
   Array ITEM [max no: 1000] {
      struct cover_state_item {
         struct series  // Named struct no: 50000
         UINT32_T cover_request_nbr_u  // Cover Request Number
         UINT32_T sequence_nbr_u  // Sequence Number
         UINT8_T state_c  // State
         char[3] filler_3_s  // Filler
         char[8] modified_date_s  // Date, Modified
         char[6] modified_time_s  // Time, Modified
         char[2] filler_2_s  // Filler
      }
   }
}
```

3.4.48.7 Answer, comments

If the maximum number of items for one transaction is returned, the query should be repeated with Sequence number 1 = set to the next missing sequence number after the Sequence Number of the last received item, and Sequence number 2 = 0.

3.4.49 CQ76 [Give Up QUERY]

3.4.49.1 Fingerprint

QUERY properties	
transaction type	CQ76
calling sequence	omniapi_query_ex
struct name	query_missing_give_up
facility	EP3
partitioned	true
answers	CA76

ANSWER properties	
transaction type	CA76
struct name	answer_missing_give_up
segmented	true

3.4.49.2 Related Messages

BD29

3.4.49.3 Purpose

The purpose of this transaction is to retrieve Give-up information. The information retrieved with this query is the same as is delivered in the Holding Give-up broadcast (BD29) broadcast. Thus, if a missing sequence number is detected for BD29, this query is used to get in synch with the broadcast flow again.

3.4.49.4 Structure

The CQ76 QUERY has the following structure:

```
struct query_missing_give_up {
   struct transaction_type
   struct series  // Named struct no: 50000
   INT32_T sequence_first_i  // Number, First Sequential
   INT32_T sequence_last_i  // Number, Last Sequential
   char[8] date_s  // Date
}
```

3.4.49.5 Usage and conditions

Series

must be completed with **Country Number**, **Market Code** and **Instrument Group**.

Number, first sequential

is the first missing one.

Number, last sequential

is the last missing one. If the Number, last sequential is equal to zero, all available deliveries are sent in sequence.

Date

must be current or next clearing date.

3.4.49.6 Answer Structure

The CA76 ANSWER has the following structure:

```
struct answer_missing_give_up {
   struct transaction_type
   UINT16_T items_n  // Items
   char[2] filler_2_s  // Filler
   Array ITEM [max no: 300] {
      struct cl_give_up_api
   }
}
```

3.4.49.7 Answer, comments

Apart from the header each record in response contains the same information as directed_give_up_t.

If the maximum number of items for one transaction is returned, the query should be repeated with Number, First sequential set to the next missing sequence number after the Sequence Number of the last received item.

3.4.50 CQ77 [Give Up History QUERY]

3.4.50.1 Fingerprint

QUERY properties	
transaction type	CQ77
calling sequence	omniapi_query_ex
struct name	query_api_give_up
facility	EP5
partitioned	true
answers	CA77

ANSWER properties	
transaction type	CA77
struct name	answer_api_give_up
segmented	false

3.4.50.2 Related Messages

CQ76

3.4.50.3 Purpose

This query is used to retrieve historical Give-ups. The information is available to the member the next business day. Historical Give-ups are queried per instrument type. To retrieve Give-ups for the current trading day, use CQ76.

3.4.50.4 Structure

The CQ77 QUERY has the following structure:

```
struct query_api_give_up {
   struct transaction_type
   struct series  // Named struct no: 50000
   char[8] from_date_s  // Date, From
   INT32_T sequence_first_i  // Number, First Sequential
   char[8] to_date_s  // Date, To
   INT32_T sequence_last_i  // Number, Last Sequential
}
```

3.4.50.5 Usage and conditions

Series

must be completed with **Country Number**, **Market Code** and **Instrument Group**.

Date, From
Date, To

must be Clearing Dates that are historical dates compared to current Clearing date. Clearing Date, From must be less or equal to Clearing Date, To.

Sequence Number 1

is the first item to get for **Clearing Date, From**. Zero or one means the first item for that date.

Sequence Number 2

is the last item to get for **Clearing Date, To**. Zero means the last item for that date.

3.4.50.6 Answer Structure

The CA77 ANSWER has the following structure:

```
struct answer_api_give_up {
   struct transaction_type
   struct series  // Named struct no: 50000
   char[8] from_date_s  // Date, From
   INT32_T sequence_first_i  // Number, First Sequential
   UINT16_T items_n  // Items
   char[2] filler_2_s  // Filler
   Array ITEM [max no: 300] {
      struct cl_give_up_api
   }
}
```

3.4.50.7 Answer, comments

Apart from the header each record in response contains the same information as directed_give_up_t.

If all giveups that reside centrally are to be fetched, the following sequence must be performed:

Loop for all instrument types defined, except for country = 255, market = 255 and instrument group = 255.

For each instrument type, do a CQ77 query until CA77 signals that no more give ups exist.

The first CQ77 is filled with the following parameters:

- Series, filled with current instrument type.
- Clearing Date, From. Set to ‘19000101’.
- Sequence Number 1. Set to 1.
- Clearing Date, To. Set to yesterday’s date.
- Sequence Number 2. Set to 0.

If Sequence Number 1 in CA77 is greater than zero, more CQ77 queries must be done to retrieve data. CQ77 must be filled with the following parameters:

- Series, filled with series in CA77.
- Clearing Date, From. Filled with Clearing Date, From in CA77.
- Sequence Number 1. Filled with Sequence Number 1 in CA77.
- Clearing Date, To. Set to yesterday’s date.
- Sequence Number 2. Set to 0.

3.4.51 CQ121 [Eligible for Cross Product Netting QUERY]

3.4.51.1 Fingerprint

QUERY properties	
transaction type	CQ121
calling sequence	omniapi_query_ex
struct name	cl_query_cross_product_netting
facility	EP3
partitioned	true

QUERY properties	
segmented	true
answers	CA121

ANSWER properties	
transaction type	CA121
struct name	cl_answer_cross_product_netting
segmented	false

3.4.51.2 Purpose

Request from the Back Office (application) for candidates eligible for cross product netting.

3.4.51.3 Structure

The CQ121 QUERY has the following structure:

```
struct cl_query_cross_product_netting {
   struct transaction_type
   struct series  // Named struct no: 50000
   struct account
   char[32] series_id_s  // Series, Identity
   UINT8_T auto_net_c  // Auto Netting
   UINT8_T non_auto_net_c  // Auto Netting ; Of type: AUTO_NET_C
   UINT16_T segment_number_n  // Segment Number
   char[8] clearing_date_s  // Clearing Date
}
```

3.4.51.4 Usage and conditions

Command from Back Office (application) requesting candidates eligible for cross product netting when a search is made in the Cross Product Netting window. Series and account, or part of them, could be sent to narrow the search as well as the flags auto_net_c and non_auto_net_c. If the flag auto_net_c is set (to 1) then the search will include auto netted accounts. If the flag non_auto_net_c is set (to 1) then the search will include accounts that are not auto netted. Note that the flags are not connected and it is possible to send both unset (0) but then the search will not return any matches.

Clearing Date

Clearing date is used for the retrieval of a position eligible for cross product netting, and it must have one of the following values:

- Today’s business date. Positions for current clearing date that are eligible for cross product netting are returned. A blank field is interpreted as today’s business date.
- Next calendar date. Positions for next clearing date that are eligible for cross product netting are returned.

Note:

The next calendar date is in relation to the current business date in the system. For example, the next calendar date will refer to a Saturday when the current business date is a Friday.

3.4.51.5 Answer Structure

The CA121 ANSWER has the following structure:

```
struct cl_answer_cross_product_netting {
   struct transaction type
   struct partition_low
   struct partition_high
   UINT16_T segment_number_n  // Segment Number
   UINT16_T items_n  // Items
   Array ITEM [max no: 500] {
      struct account
      struct series  // Named struct no: 50000
      struct mini_series  // Of type: SERIES ; Named struct no: 50000
      INT64_T nbr_held_q  // Held
      INT64_T mini_nbr_held_q  // Held ; Of type: NBR_HELD_Q
      INT64_T nbr_written_q  // Written
      INT64_T mini_nbr_written_q  // Written ; Of type: NBR_WRITTEN_Q
      INT64_T td_long_q  // Today long position
      INT64_T mini_td_long_q  // Today long position ; Of type: TD_LONG_Q
      INT64_T td_short_q  // Today short position
      INT64_T mini_td_short_q  // Today short position ; Of type: TD_SHORT_Q
      INT32_T netting_ratio_i  // Netting Ratio
   }
}
```

3.5 Risk Management

3.5.1 RQ1 [Margin Parameters for Series QUERY]

3.5.1.1 Fingerprint

QUERY properties	
transaction type	RQ1
calling sequence	omniapi_query_ex
struct name	query_margin_series_param
facility	EP4
partitioned	false
answers	RA1

ANSWER properties	
transaction type	RA1
struct name	answer_margin_series_param
segmented	true

3.5.1.2 Purpose

This query contains calculated margin parameter values for series. The values obtained are those used at the most recent official day-end margin calculation.

3.5.1.3 Structure

The RQ1 QUERY has the following structure:

```
struct query_margin_series_param {
   struct transaction_type
   struct series  // Named struct no: 50000
   UINT16_T segment_number_n  // Segment Number
   char[2] filler_2_s  // Filler
}
```

3.5.1.4 Usage and conditions

Series

must be completed by **Country Number** and **Market Code** or a complete Series.

3.5.1.5 Answer Structure

The RA1 ANSWER has the following structure:

```
struct answer_margin_series_param {
   struct transaction_type
   UINT16_T segment_number_n  // Segment Number
   UINT16_T items_n  // Items
   Array ITEM [max no: 500] {
      struct series  // Named struct no: 50000
      INT32_T down_int_i  // Valuation Interval, Down
      INT32_T up_int_i  // Valuation Interval, Up
      INT32_T risk_free_rate_i  // Interest, Risk Free
      INT32_T held_vol_down_i  // Volatility Held Down
      INT32_T held_vol_up_i  // Volatility Held Up
      INT32_T writ_vol_down_i  // Volatility Written, Down
      INT32_T writ_vol_up_i  // Volatility Written, Up
      INT32_T fixed_vol_i  // Volatility, Fixed
      INT32_T held_for_adj_i  // Future Adjustment Held
      INT32_T writ_for_adj_i  // Future Adjustment Written
      char[15] par_id_s  // Parameter Block
      char[15] win_id_s  // Window Class
      char[2] filler_2_s  // Filler
```

```
        INT32_T bid_marg_vol_i  // Margin, Volatility Bid
        INT32_T ask_marg_vol_i  // Margin, Volatility Ask
        INT32_T dividend_yield_i  // Dividend, Yield
        INT32_T ind_bid_marg_vol_i  // Margin, Individual Volatility Bid
        INT32_T ind_ask_marg_vol_i  // Margin, Individual Volatility Ask
    }
}
```

3.5.1.6 Answer, comments

Data will not be returned for new TM series that have been added during the day.

3.5.2 RQ2 [Margin Parameter Block QUERY]

3.5.2.1 Fingerprint

QUERY properties	
transaction type	RQ2
calling sequence	omniapi_query_ex
struct name	query_margin_param_block
facility	EP4
partitioned	false
answers	RA2

ANSWER properties	
transaction type	RA2
struct name	answer_margin_param_block
segmented	true

3.5.2.2 Purpose

This query contains margin parameter blocks (user + system risk parameters). This may be queried either from evening calculations or from intra day calculations.

3.5.2.3 Structure

The RQ2 QUERY has the following structure:

```
struct query_margin_param_block {
   struct transaction_type
   struct series  // Named struct no: 50000
   UINT16_T segment_number_n  // Segment Number
   char[8] date_s  // Date
   UINT8_T intra_day2_c  // Intra Day2
   CHAR filler_1_s  // Filler
```

```
    UINT16_T marg_run_nbr_n  // Margin run number
    UINT16_T marg_call_nbr_n  // Margin call number
}
```

3.5.2.4 Usage and conditions

Series

must be completed by **Country Number** and **Market Code**.

Date

This could either be set to a business date or to blank. If set to blank, data for most recent day end calculation will be returned..

Results from evening calculations are only available when the signal BI7, Information type 8 has been sent.

New intra day calculations are available when the signal BI7, information type 42 has been sent.

New margin call results are available when the signal BI7, information type 10 has been sent.

3.5.2.5 Answer Structure

The RA2 ANSWER has the following structure:

```
struct answer_margin_param_block {
   struct transaction_type
   UINT16_T segment_number_n  // Segment Number
   UINT16_T items_n  // Items
   UINT16_T marg_run_nbr_n  // Margin run number
   UINT16_T marg_call_nbr_n  // Margin call number
   char[8] created_date_s  // Date, Created
   char[6] created_time_s  // Time, Created
   char[6] filler_6_s  // Filler
   Array ITEM [max no: 100] {
      char[15] pur_id_s  // Parameter Block
      CHAR neg_time_adj_c  // Negative Time Value Adjustment
      INT64_T iter_accuracy_q  // Iteration, Accuracy
      INT32_T erosion_i  // Erosion Adjustment
      INT32_T held_vol_max_i  // Volatility Held, Max
      INT32_T writ_vol_min_i  // Volatility Written, Min
      INT32_T held_val_max_i  // Spread Minimum
      INT32_T writ_val_min_i  // Value Written, Min
      INT32_T held_zero_limit_i  // Value Held Zero Limit
      INT32_T swap_lead_time_i  // Swaps, Lead Time
      INT32_T fix_rate_down_i  // Fix Rate Step Down
      INT32_T fix_rate_up_i  // Fix Rate Step Up
      INT32_T val_closed_risk_down_i  // Closed Risk Down
      INT32_T val_closed_risk_up_i  // Closed Risk Up
      INT32_T hrm_corr_i  // Reduction Correlation, Hourly
      INT32_T pow_offset_days_i  // Value Offset, Known
      INT32_T points_reg_i  // Points, Number of
      INT32_T vol_steps_held_i  // Volatility Steps, Held
      INT32_T vol_steps_writ_i  // Volatility Steps, Written
      INT32_T vol_spread_held_i  // Volatility Spread, Held
```

```
        INT32_T vol_spread_writ_i  // Volatility Spread, Written
        INT32_T float_swap_steps_i  // Float Rate Steps
        INT32_T er_trd_days_in_year_i  // Trading Days Erosion
        INT32_T sw_trd_days_in_year_i  // Swaps, Trading Days
        INT32_T sw_settl_days_i  // Settlement Days, Swap
        INT32_T marg_settl_days_i  // Settlement Days
        INT32_T iter_low_bound_i  // Iteration, Low Bound
        INT32_T iter_high_bound_i  // Iteration, High Bound
        INT32_T iter_max_no_i  // Iteration, Max
        INT32_T ulg_price_spread_i  // Underlying Price Spread
        INT32_T bin_val_time_step_i  // Time Steps
        INT32_T fut_spread_rate_i  // Future Spread Rate
        INT32_T prod_grp_offset_i  // Product Group Offset
        CHAR vol_used_c  // Volatility Type
        CHAR opt_price_base_1_c  // Option Price Base 1
        CHAR opt_price_base_2_c  // Option Price Base 2
        CHAR corr_method_c  // Margining Method
        UINT8_T margin_dividend_c  // Margin, Dividend
        UINT8_T margin_deliv_c  // Margin, Delivery
        UINT8_T margin_payment_c  // Payment Margin
        UINT8_T val_ivl_type_c  // Valuation Interval, Type
        INT32_T allwd_price_move_i  // Price Movement Max
        UINT8_T val_ivl_base_c  // Valuation Interval, Base
        UINT8_T price_move_guard_c  // Price Movement Guard
        UINT8_T vol_interval_type_c  // Volatility Interval, Type
        UINT8_T base_offset_days_c  // Offset Days For Settlement Margin Base
        UINT16_T offset_days_n  // Offset Days for Settlement Margin
        UINT8_T real_time_price_use_c  // Real Time Price Usage
        UINT8_T interest_rate_type_c  // Interest Rate Type
        UINT16_T day_count_n  // Day Count
        UINT8_T forw_margin_c  // Forward Margining
        UINT8_T real_time_price_opt_c  // Real Time Price, Options
        UINT8_T real_time_price_fut_c  // Real Time Price, Futures/Forwards
        char[3] filler_3_s  // Filler
        INT32_T cover_margin_i  // Cover Margin
        INT32_T base_srs_cutoff_time_i  // Cutoff time base series
        UINT16_T price_carrier_code_n  // Price carrier
        char[11] reserved_11_s  // Reserved
        char[7] filler_7_s  // Filler
    }
}
```

3.5.2.6 Answer, comments

Time created

equals calculation time in the intra day case. The field is blank in the evening case.

3.5.3 RQ3 [Extended Margin Parameters for series QUERY]

3.5.3.1 Fingerprint

QUERY properties	
transaction type	RQ3
calling sequence	omniapi_query_ex
struct name	query_margin_series_param_ext
facility	EP4
partitioned	false
answers	RA3

ANSWER properties	
transaction type	RA3
struct name	answer_margin_series_param_ext
segmented	true

3.5.3.2 Purpose

This query contains calculated margin and price parameter values for series. This may be queried either from evening calculations or from intra day calculations.

3.5.3.3 Structure

The RQ3 QUERY has the following structure:

```
struct query_margin_series_param_ext {
   struct transaction_type
   struct series  // Named struct no: 50000
   UINT16_T segment_number_n  // Segment Number
   char[8] date_s  // Date
   UINT8_T intra_day2_c  // Intra Day2
   CHAR filler_1_s  // Filler
   UINT16_T marg_run_nbr_n  // Margin run number
   UINT16_T marg_call_nbr_n  // Margin call number
}
```

3.5.3.4 Usage and conditions

Series

must be completed with Country Number and Market Code or a complete Series.

Results from evening calculations are only available when the signal BI7, Information type 8 has been sent.

New intra day calculations are available when the signal BI7, information type 42 has been sent.

New margin call results are available when the signal BI7, information type 10 has been sent.

3.5.3.5 Answer Structure

The RA3 ANSWER has the following structure:

```
struct answer_margin_series_param_ext {
   struct transaction_type
   UINT16_T segment_number_n  // Segment Number
   UINT16_T items_n  // Items
   UINT16_T marg_run_nbr_n  // Margin run number
   UINT16_T marg_call_nbr_n  // Margin call number
   char[8] created_date_s  // Date, Created
   char[6] created_time_s  // Time, Created
   char[6] filler_6_s  // Filler
   Array ITEM [max no: 500] {
      struct series  // Named struct no: 50000
      INT32_T down_int_i  // Valuation Interval, Down
      INT32_T up_int_i  // Valuation Interval, Up
      INT32_T risk_free_rate_i  // Interest, Risk Free
      INT32_T held_vol_down_i  // Volatility Held Down
      INT32_T held_vol_up_i  // Volatility Held Up
      INT32_T writ_vol_down_i  // Volatility Written, Down
      INT32_T writ_vol_up_i  // Volatility Written, Up
      INT32_T fixed_vol_i  // Volatility, Fixed
      INT32_T held_for_adj_i  // Future Adjustment Held
      INT32_T writ_for_adj_i  // Future Adjustment Written
      INT32_T dividend_yield_i  // Dividend, Yield
      char[15] marg_param_id_s  // Margin Parameter
      char[15] price_param_id_s  // Price Parameter
      char[15] win_id_s  // Window Class
      char[16] tdp_id_s  // Parameter, Time Dependent Identity
      char[3] filler_3_s  // Filler
   }
}
```

3.5.3.6 Answer, comments

Time created

equals calculation time in the intra day case. The field is blank in the evening case.

For intra day calculations, data will not be returned for new TM series that have been added during the day.

3.5.4 RQ6 [Extended Margin Information QUERY]

3.5.4.1 Fingerprint

QUERY properties	
transaction type	RQ6
calling sequence	omniapi_query_ex
struct name	query_margin_ext
facility	EP4
partitioned	false
answers	RA6

ANSWER properties	
transaction type	RA6
struct name	answer_margin_ext
segmented	true

3.5.4.2 Purpose

This query contains margin requirements at a detailed level per account and series.

3.5.4.3 Structure

The RQ6 QUERY has the following structure:

```
struct query_margin_ext {
   struct transaction_type
   struct series  // Named struct no: 50000
   UINT16_T segment_number_n  // Segment Number
   char[8] date_s  // Date
   char[2] filler_2_s  // Filler
}
```

3.5.4.4 Usage and conditions

This query is only available when the signal BI7, Information type 8 has been sent.

Series

must be completed with **Country Number** and **Market Code**.

3.5.4.5 Answer Structure

The RA6 ANSWER has the following structure:

```
struct answer_margin_ext {
   struct transaction_type
   UINT16_T segment_number_n  // Segment Number
   UINT16_T items_n  // Items
   Array ITEM [max no: 500] {
      struct series  // Named struct no: 50000
      INT64_T margin_req_u  // Margin Requirements
      INT64_T market_value_q  // Market Value
      struct account
      char[3] currency_s  // Currency
      CHAR filler_1_s  // Filler
   }
}
```

3.5.5 RQ7 [Margin Detail QUERY]

3.5.5.1 Fingerprint

QUERY properties	
transaction type	RQ7
calling sequence	omniapi_query_ex
struct name	query_margin_detail
facility	EP4
partitioned	false
answers	RA7

ANSWER properties	
transaction type	RA7
struct name	answer_margin_detail
segmented	true

3.5.5.2 Purpose

The purpose of this transaction is to retrieve margin results on a detailed level, that is, per account and series.

3.5.5.3 Structure

The RQ7 QUERY has the following structure:

```
struct query_margin_detail {
   struct transaction_type
   struct series  // Named struct no: 50000
   UINT16_T segment_number_n  // Segment Number
   char[8] date_s  // Date
   UINT8_T intra_day2_c  // Intra Day2
   CHAR filler_1_s  // Filler
   struct account
   UINT16_T marg_run_nbr_n  // Margin run number
   UINT16_T marg_call_nbr_n  // Margin call number
}
```

3.5.5.4 Usage and conditions

Series

must be complete up to Country Number and Market Code.

Account

must be filled in one of the following ways:

- Fill in the field with explicit value. All answers must match this field

- Fill in the field with “*”. No test is made on the value for this field.

- Fill in the field with a string ended by “*”. All answers must in this field start with the string specified.

Intra Day2

Possible values:

0	Evening data, propagated
1	Intra day calculation, propagated
2	Intra day margin call, propagated
10	Evening data, non-propagated
11	Intra day calculation, non-propagated

Results from evening calculations are only available when the signal BI7, information type 8 has been sent.

New intra day calculations are available when the signal BI7, information type 42 has been sent.

New margin call results are available when the signal BI7, information type 10 has been sent.

3.5.5.5 Answer Structure

The RA7 ANSWER has the following structure:

```
struct answer_margin_detail {
   struct transaction_type
   UINT16_T segment_number_n  // Segment Number
   UINT16_T items_n  // Items
   UINT16_T marg_run_nbr_n  // Margin run number
```

```
   UINT16_T marg_call_nbr_n  // Margin call number
   char[8] created_date_s  // Date, Created
   char[6] created_time_s  // Time, Created
   char[6] filler_6_s  // Filler
   Array ITEM [max no: 450] {
      struct account
      struct series  // Named struct no: 50000
      INT64_T margin_req_u  // Margin Requirements
      INT64_T market_value_q  // Market Value
      INT64_T nbr_held_q  // Held
      INT64_T nbr_written_q  // Written
      INT64_T held_marg_q  // Marginables, Held
      INT64_T writ_marg_q  // Marginables, Written
      INT64_T cash_margin_q  // Cash Margin
      INT64_T naked_margin_q  // Margin Requirements, Naked
      INT64_T pay_margin_q  // Payment Margin
      INT64_T orig_market_value_q  // Original market value
      INT64_T unconv_market_value_q  // Unconverted market value
      UINT32_T quantity_cover_u  // Quantity Cover
      char[3] currency_s  // Currency
      UINT8_T gross_or_net_c  // Gross Or Net
      char[3] cash_currency_s  // Currency, Cash
      char[3] margin_class_s  // Margin class
      UINT8_T marg_meth_inst_c  // Margin method, for instrument class and
instrument series
      UINT8_T marg_item_type_c  // Margin item type
   }
}
```

3.5.5.6 Answer, comments

Time Created

equals calculation time in the intra day case. The field is blank in the evening case.

Marginables, Held
Marginables, Written

are derived from Held, Written and Quantity Cover in the following way:

- Held marginable = Held

- Written marginable = Written – Quantity Cover

- If net margining is applied, Held marginable and Written Marginable are netted down so that one of the sides equals zero (0).

3.5.6 RQ14 [Risk Array QUERY]

3.5.6.1 Fingerprint

QUERY properties	
transaction type	RQ14

QUERY properties	
calling sequence	omniapi_query_ex
struct name	query_risk_array
facility	EP4
partitioned	false
answers	RA14

ANSWER properties	
transaction type	RA14
struct name	answer_risk_array
segmented	true

3.5.6.2 Purpose

This query contains risk array values and composite delta for instruments using any style of Delta Hedging Method as margining method. This may be queried either from evening calculations or from intra day calculations.

3.5.6.3 Structure

The RQ14 QUERY has the following structure:

```
struct query_risk_array {
   struct transaction_type
   struct series  // Named struct no: 50000
   UINT16_T segment_number_n  // Segment Number
   char[8] date_s  // Date
   UINT8_T intra_day4_c  // Intra Day4
   UINT8_T clh_or_cst_c  // Clearing house or customer value
   UINT16_T marg_run_nbr_n  // Margin run number
   UINT16_T marg_call_nbr_n  // Margin call number
}
```

3.5.6.4 Usage and conditions

Series

must be completed by **Country Number** and **Market Code**.

Results from evening calculations are only available when the signal BI7, Information type 8 has been sent.

New intra day calculations are available when the signal BI7, information type 42 has been sent.

New margin call results are available when the signal BI7, information type 10 has been sent.

Results from preliminary evening calculations are available when the signal BI7, information type 41, has been sent.

3.5.6.5 Answer Structure

The RA14 ANSWER has the following structure:

```
struct answer_risk_array {
   struct transaction_type
   UINT16_T segment_number_n  // Segment Number
   UINT16_T items_n  // Items
   UINT16_T marg_run_nbr_n  // Margin run number
   UINT16_T marg_call_nbr_n  // Margin call number
   char[8] created_date_s  // Date, Created
   char[6] created_time_s  // Time, Created
   char[6] filler_6_s  // Filler
   Array ITEM [max no: 700] {
      struct series  // Named struct no: 50000
      INT32_T risk_array_p1_i  // Risk array point 1
      INT32_T risk_array_p2_i  // Risk array point 2
      INT32_T risk_array_p3_i  // Risk array point 3
      INT32_T risk_array_p4_i  // Risk array point 4
      INT32_T risk_array_p5_i  // Risk array point 5
      INT32_T risk_array_p6_i  // Risk array point 6
      INT32_T risk_array_p7_i  // Risk array point 7
      INT32_T risk_array_p8_i  // Risk array point 8
      INT32_T risk_array_p9_i  // Risk array point 9
      INT32_T risk_array_p10_i  // Risk array point 10
      INT32_T risk_array_p11_i  // Risk array point 11
      INT32_T risk_array_p12_i  // Risk array point 12
      INT32_T risk_array_p13_i  // Risk array point 13
      INT32_T risk_array_p14_i  // Risk array point 14
      INT32_T risk_array_p15_i  // Risk array point 15
      INT32_T risk_array_p16_i  // Risk array point 16
      INT32_T comp_delta_i  // Composite Delta
      char[8] closing_date_s  // Date, Closing
   }
}
```

3.5.6.6 Answer, comments

Time created

equals calculation time in the intra day case. The field is blank in the evening case.

3.5.7 RQ20 [Account Product Area Margin QUERY]

3.5.7.1 Fingerprint

QUERY properties	
transaction type	RQ20
calling sequence	omniapi_query_ex
struct name	query_margin_pa_acc

QUERY properties	
facility	EP4
partitioned	false
answers	RA20

ANSWER properties	
transaction type	RA20
struct name	answer_margin_pa_acc
segmented	true

3.5.7.2 Purpose

This query contains sum margin requirement per account, product area and instrument currency.

3.5.7.3 Structure

The RQ20 QUERY has the following structure:

```
struct query_margin_pa_acc {
   struct transaction_type
   struct series  // Named struct no: 50000
   UINT16_T segment_number_n  // Segment Number
   char[2] filler_2_s  // Filler
   char[8] date_s  // Date
   struct account
   char[12] cust_bank_id_s  // Custodian Bank
}
```

3.5.7.4 Usage and conditions

A product area is the entity that is margined together. It may be one market or a set of markets.

This query is only available when the signal BI7, Information type 11 has been sent.

Series

The query does not filter on series, hence the series chould be completed with any Country Number and Market Code.

Customer
Account
Custodian Bank

must all be filled in with values in one of the following ways:

1. Fill in the field with explicit value. All answers must match that field.
2. Fill in the field with “*”. No test is made on the value for that field.

3. Fill in the field with a string ended by "*". All answers must in this field start with the string specified.

3.5.7.5 Answer Structure

The RA20 ANSWER has the following structure:

```
struct answer_margin_pa_acc {
   struct transaction_type
   UINT16_T segment_number_n  // Segment Number
   UINT16_T items_n  // Items
   Array ITEM [max no: 500] {
      struct account
      char[3] market_currency_s  // Currency, Market
      CHAR filler_1_s  // Filler
      INT64_T market_margin_q  // Margin Requirements, Market
      INT64_T market_value_q  // Market Value
      INT64_T cash_margin_q  // Cash Margin
      UINT8_T prod_area_c  // Product Area, RIVA
      UINT8_T acc_risk_type_c  // Account Risk Type
      char[10] prod_area_text_s  // Product Area Text, RIVA
      char[12] cust_bank_id_s  // Custodian Bank
   }
}
```

3.5.7.6 Answer, comments

The key to the answer items consists of the following fields:

- **Customer**
- **Account**
- **Product Area**
- **Currency, Market**

3.5.8 RQ21 [Account Sum Margin QUERY]

3.5.8.1 Fingerprint

QUERY properties	
transaction type	RQ21
calling sequence	omniapi_query_ex
struct name	query_margin_acc
facility	EP4
partitioned	false
answers	RA21

ANSWER properties	
transaction type	RA21
struct name	answer_margin_acc
segmented	true

3.5.8.2 Purpose

This query contains sum margin requirement per account, currency and custodian bank together with currency conversions made.

3.5.8.3 Structure

The RQ21 QUERY has the following structure:

```
struct query_margin_acc {
   struct transaction_type
   struct series  // Named struct no: 50000
   UINT16_T segment_number_n  // Segment Number
   char[2] filler_2_s  // Filler
   char[8] date_s  // Date
   struct account
   char[12] cust_bank_id_s  // Custodian Bank
}
```

3.5.8.4 Usage and conditions

This query is only available when the signal BI7, Information type 11 has been sent.

Series

could be completed with any **Country Number** and **Market Code**.

Customer
Account
Custodian Bank

must all be filled in with values in one of the following ways:

1. Fill in the field with explicit value. All answers must match that field.
2. Fill in the field with “*”. No test is made on the value for that field.
3. Fill in the field with a string ended by “*”. All answers must in this field start with the

3.5.8.5 Answer Structure

The RA21 ANSWER has the following structure:

```
struct answer_margin_acc {
   struct transaction_type
```

```
    UINT16_T segment_number_n  // Segment Number
    UINT16_T items_n  // Items
    Array ITEM [max no: 500] {
       struct account
       char[3] market_currency_s  // Currency, Market
       CHAR filler_1_s  // Filler
       INT64_T market_margin_q  // Margin Requirements, Market
       INT64_T risk_margin_q  // Margining Requirements, Risk
       char[12] cust_bank_id_s  // Custodian Bank
       char[3] risk_currency_s  // Currency, Risk
       UINT8_T acc_risk_type_c  // Account Risk Type
    }
}
```

3.5.8.6 Answer, comments

Currency, Market
Margining Requirements, Market

apply to the native currencies of the markets.

Currency, Risk
Margining Requirements, Risk

apply to margin requirements after currency conversions have been made.

The key to the answer items consists of the following fields:

- **Customer**
- **Account**
- **Currency, Market**
- **Custodian Bank**

3.5.9 RQ23 [Member Sum Margin QUERY]

3.5.9.1 Fingerprint

QUERY properties	
transaction type	RQ23
calling sequence	omniapi_query_ex
struct name	query_margin_mem
facility	EP4
partitioned	false
answers	RA23

ANSWER properties	
transaction type	RA23

ANSWER properties	
struct name	answer_margin_mem
segmented	true

3.5.9.2 Purpose

This query contains sum margin requirement per member, currency and custodian bank. It only contains the indirect pledging accounts belonging to the member; direct pledging accounts are not included.

3.5.9.3 Structure

The RQ23 QUERY has the following structure:

```
struct query_margin_mem {
   struct transaction_type
   struct series  // Named struct no: 50000
   UINT16_T segment_number_n  // Segment Number
   char[8] date_s  // Date
   char[2] filler_2_s  // Filler
}
```

3.5.9.4 Usage and conditions

This query is only available when the signal BI7, Information type 11 has been sent.

Series

could be completed with any **Country Number** and **Market Code**.

3.5.9.5 Answer Structure

The RA23 ANSWER has the following structure:

```
struct answer_margin_mem {
   struct transaction_type
   UINT16_T segment_number_n  // Segment Number
   UINT16_T items_n  // Items
   Array ITEM [max no: 500] {
      char[2] country_id_s  // Name, Country
      char[5] ex_customer_s  // Customer, Identity
      char[3] risk_currency_s  // Currency, Risk
      char[12] cust_bank_id_s  // Custodian Bank
      char[2] filler_2_s  // Filler
      INT64_T risk_margin_q  // Margining Requirements, Risk
   }
}
```

3.5.9.6 Answer, comments

The key to the answer items consists of the following fields:

- **Customer**
- **Currency, Risk**
- **Custodian Bank**

3.5.10 RQ31 [Margin Exchange Rate QUERY]

3.5.10.1 Fingerprint

QUERY properties	
transaction type	RQ31
calling sequence	omniapi_query_ex
struct name	query_exchange_rate
facility	EP4
partitioned	false
answers	RA31

ANSWER properties	
transaction type	RA31
struct name	answer_exchange_rate
segmented	true

3.5.10.2 Purpose

This query contains exchange rates used in margin calculations.

3.5.10.3 Structure

The RQ31 QUERY has the following structure:

```
struct query_exchange_rate {
   struct transaction type
   struct series  // Named struct no: 50000
   UINT16_T segment_number_n  // Segment Number
   char[8] date_s  // Date
   char[2] filler_2_s  // Filler
}
```

3.5.10.4 Usage and conditions

This query is only available when the signal BI7, Information type 11 has been sent.

Series

could be completed with any **Country Number** and **Market Code**.

3.5.10.5 Answer Structure

The RA31 ANSWER has the following structure:

```
struct answer_exchange_rate {
   struct transaction_type
   UINT16_T segment_number_n  // Segment Number
   UINT16_T items_n  // Items
   Array ITEM [max no: 500] {
      INT32_T rate_nominal_i  // Rate, Nominal
      INT32_T price_quot_factor_i  // Price, Quotation Factor
      INT32_T rate_low_i  // Rate, Low
      INT32_T rate_high_i  // Rate, High
      UINT16_T dec_in_rate_n  // Decimals, Rate
      UINT16_T dec_in_contr_size_n  // Decimals, Contract Size
      char[3] price_currency_s  // Currency, Price
      char[3] other_currency_s  // Currency, Other
      char[2] filler_2_s  // Filler
   }
}
```

3.5.10.6 Answer, comments

Currency, Price

is the currency in which the exchange rate is defined.

Currency, Other

is the other leg of the exchange rate.

The key to the answer items consists of the fields:

- **Currency, Price**
- **Currency, Other**

> *Example*
>
> If 1 USD costs 8 SEK, Currency Price is SEK and Currency, other is USD.
>
> Price Quotation Factor applies to the rate fields, and means the amount by which the rates should be multiplied in order to get the price of 1 Currency, other expressed in Currency, Price.
>
> Decimals, Contract Size equals the number of decimals in the Price Quotation Factor field.

3.5.11 RQ35 [Data Used for Margin Calculation QUERY]

3.5.11.1 Fingerprint

QUERY properties	
transaction type	RQ35
calling sequence	omniapi_query_ex
struct name	query_margin_data_used
facility	EP4
partitioned	false
answers	RA35

ANSWER properties	
transaction type	RA35
struct name	answer_margin_data_used
segmented	true

3.5.11.2 Purpose

The purpose of this transaction is to retrieve data that was used for margin calculations This may be queried either from evening calculations or from intra day calculations.

3.5.11.3 Structure

The RQ35 QUERY has the following structure:

```
struct query_margin_data_used {
   struct transaction_type
   struct series  // Named struct no: 50000
   UINT16_T segment_number_n  // Segment Number
   char[8] date_s  // Date
   UINT8_T intra_day2_c  // Intra Day2
   CHAR filler_1_s  // Filler
   UINT16_T marg_run_nbr_n  // Margin run number
   UINT16_T marg_call_nbr_n  // Margin call number
}
```

3.5.11.4 Usage and conditions

Series

must be complete up to **Country Number** and **Market Code**.

Results from evening calculations are only available when the signal BI7, Information type 8 has been sent.

New intra day calculations are available when the signal BI7, information type 42 has been sent.

New margin call results are available when the signal BI7, information type 10 has been sent.

3.5.11.5 Answer Structure

The RA35 ANSWER has the following structure:

```
struct answer_margin_data_used {
   struct transaction_type
   UINT16_T segment_number_n  // Segment Number
   UINT16_T items_n  // Items
   UINT16_T marg_run_nbr_n  // Margin run number
   UINT16_T marg_call_nbr_n  // Margin call number
   char[8] created_date_s  // Date, Created
   char[6] created_time_s  // Time, Created
   char[6] filler_6_s  // Filler
   Array ITEM [max no: 600] {
      struct series  // Named struct no: 50000
      char[3] currency_s  // Currency
      UINT8_T vol_src_c  // Volatility Source
      INT64_T margin_one_writ_opt_q  // Margining Requirements, One Written
Option
      UINT32_T bid_price_i  // Bid Price
      UINT32_T ask_price_i  // Ask Price
      INT32_T marg_price_i  // Margin, Settlement Price
      INT32_T fixing_value_i  // Fixing Value
      INT32_T val_ivl_mid_i  // Valuation Interval, Mid
      INT32_T val_ivl_low_i  // Valuation Interval, Low
      INT32_T val_ivl_high_i  // Valuation Interval, High
      INT32_T vol_ivl_held_mid_i  // Volatility Interval Held, Mid
      INT32_T vol_ivl_writ_mid_i  // Volatility Interval Written, Mid
      INT32_T vol_ivl_held_low_i  // Volatility Interval Held, Low
      INT32_T vol_ivl_writ_low_i  // Volatility Interval Written, Low
      INT32_T vol_ivl_held_high_i  // Volatility Interval Held, High
      INT32_T vol_ivl_writ_high_i  // Volatility Interval Written, High
      INT32_T remaining_contract_size_i  // Contract Size, Remaining
      UINT16_T dec_in_price_n  // Decimals, Price
      UINT8_T opt_price_model_c  // Option Price Model
      UINT8_T opt_ulg_price_src_c  // Option Underlying Price Source
      INT32_T ulg_vola_i  // Underlying volatility value
      INT32_T flat_rate_increase_i  // Flat rate increase
      INT32_T flat_rate_decrease_i  // Flat rate decrease
      INT32_T flat_rate_gain_discount_i  // Flat rate gain discount
      char[4] filler_4_s  // Filler
   }
}
```

3.5.11.6 Answer, comments

Time created

equals calculation time in the intra day case. The field is blank in the evening case.

Decimals, price

equals number of decimals in valuation intervals mid/low/high.

Margining requirements, one written option
Volatility interval held, mid
Volatility interval written, mid
Volatility interval held, low
Volatility interval written, low
Volatility interval held, high
Volatility interval written, high
Option price model
Option underlying price source

are all zero for instruments that are not options.

Flat rate increase/decrease/gain discount

For instrument series where flat rate margin is not applied, these fields will always equal zero.

The answer received contains a list of data per series. Each response is prefaced with the transaction type and an Item field specifying the number of records contained in the response.

For intra day calculations, data will not be returned for new TM series that have been added during the day.

3.5.12 RQ36 [Greeks QUERY]

3.5.12.1 Fingerprint

QUERY properties	
transaction type	RQ36
calling sequence	omniapi_query_ex
struct name	query_greeks
facility	EP4
partitioned	false
answers	RA36

ANSWER properties	
transaction type	RA36
struct name	answer_greeks
segmented	true

3.5.12.2 Purpose

The purpose of this transaction is to retrieve Option Greeks calculated by the margin system. These may be queried either from evening calculations or from intra day calculations.

3.5.12.3 Structure

The RQ36 QUERY has the following structure:

```
struct query_greeks {
   struct transaction_type
   struct series  // Named struct no: 50000
   UINT16_T segment_number_n  // Segment Number
   char[8] date_s  // Date
   UINT8_T intra_day2_c  // Intra Day2
   CHAR filler_1_s  // Filler
   UINT16_T marg_run_nbr_n  // Margin run number
   UINT16_T marg_call_nbr_n  // Margin call number
}
```

3.5.12.4 Usage and conditions

Series

must be complete up to Country Number and Market Code.

The interpretation of BI7 signals is as following:

Information type 8 (some exchanges uses Information type 47)	Results from evening calculations are available.
Information type 10	New margin call results are available.
Information type 42 and 43	Results from latest available intra-day margin calculations (intra day2=1) are available.

3.5.12.5 Answer Structure

The RA36 ANSWER has the following structure:

```
struct answer_greeks {
   struct transaction_type
   UINT16_T segment_number_n  // Segment Number
   UINT16_T items_n  // Items
   UINT16_T marg_run_nbr_n  // Margin run number
   UINT16_T marg_call_nbr_n  // Margin call number
   char[8] created_date_s  // Date, Created
   char[6] created_time_s  // Time, Created
   char[6] filler_6_s  // Filler
   Array ITEM [max no: 1500] {
      struct series  // Named struct no: 50000
      INT32_T delta_i  // Delta
      INT32_T gamma_i  // Gamma
      INT32_T vega_i  // Vega
      INT32_T theta_i  // Theta
      INT32_T rho_i  // Rate Of Change, Option Value
   }
}
```

3.5.12.6 Answer, comments

Time Created

equals calculation time in the intra day case. The field is blank in the evening case.

For intra day calculations, data will not be returned for new TM series that have been added during the day.

3.5.13 RQ37 [Volatility Skew QUERY]

3.5.13.1 Fingerprint

QUERY properties	
transaction type	RQ37
calling sequence	omniapi_query_ex
struct name	query_volatility_skew
facility	EP4
partitioned	false
answers	RA37

ANSWER properties	
transaction type	RA37
struct name	answer_volatility_skew
segmented	true

3.5.13.2 Purpose

The purpose of this transaction is to retrieve volatility skew calculated by the margin system.

3.5.13.3 Structure

The RQ37 QUERY has the following structure:

```
struct query_volatility_skew {
   struct transaction_type
   struct series  // Named struct no: 50000
   UINT16_T segment_number_n  // Segment Number
   char[8] date_s  // Date
   char[2] filler_2_s  // Filler
}
```

3.5.13.4 Usage and conditions

Series

must be complete up to Country Number and Market Code.

3.5.13.5 Answer Structure

The RA37 ANSWER has the following structure:

```
struct answer_volatility_skew {
   struct transaction_type
   UINT16_T segment_number_n  // Segment Number
   UINT16_T items_n  // Items
   Array ITEM [max no: 200] {
      UINT16_T commodity_n  // Commodity Code
      char[8] date_s  // Date
      char[2] filler_2_s  // Filler
      INT32_T vol_base_i  // Volatility Base
      INT32_T skewness_down_i  // Skewness, Down
      INT32_T skewness_up_i  // Skewness, Up
      INT32_T strike_interval_i  // Strike Interval
   }
}
```

3.5.13.6 Answer, comments

The answer received contains a list of volatility and skewness per underlying and expiration.

Each response is prefaced with the transaction type (RA37) and an item field specifying the number of records contained in the response.

3.5.14 RQ41 [Margin Underlying Price QUERY]

3.5.14.1 Fingerprint

QUERY properties	
transaction type	RQ41
calling sequence	omniapi_query_ex
struct name	query_margin_ulg_price
facility	EP4
partitioned	false
answers	RA41

ANSWER properties	
transaction type	RA41

ANSWER properties	
struct name	answer_margin_ulg_price
segmented	true

3.5.14.2 Purpose

This query contains underlying prices used in margin calculations.

Note: RQ41 will be replaced by RQ45.

3.5.14.3 Structure

The RQ41 QUERY has the following structure:

```
struct query_margin_ulg_price {
   struct transaction_type
   struct series  // Named struct no: 50000
   UINT16_T segment_number_n  // Segment Number
   char[8] date_s  // Date
   char[2] filler_2_s  // Filler
}
```

3.5.14.4 Usage and conditions

Series

must be completed by **Country Number** and **Market Code** Data will be returned for underlyings having series in the specified market.

3.5.14.5 Answer Structure

The RA41 ANSWER has the following structure:

```
struct answer_margin_ulg_price {
   struct transaction_type
   UINT16_T segment_number_n  // Segment Number
   UINT16_T items_n  // Items
   Array ITEM [max no: 300] {
      UINT16_T commodity_n  // Commodity Code
      char[2] filler_2_s  // Filler
      UINT32_T bid_price_i  // Bid Price
      UINT32_T ask_price_i  // Ask Price
      INT32_T marg_price_i  // Margin, Settlement Price
      INT32_T last_paid_i  // Last, Paid
      UINT8_T bid_theo_c  // Bid, Theoretical Mark
      UINT8_T ask_theo_c  // Ask, Theoretical Mark
      UINT8_T last_theo_c  // Last Paid, Theoretical Mark
      UINT8_T marg_theo_c  // Margin, Settlement Price Theoretical Mark
   }
```

```
}
```

3.5.14.6 Answer, comments

The response is a list of underlyings together with prices used in margin calculations.

The underlyings received are the underlyings that have series in the market specified in the query.

The answer is available at the same time as the margin information is available, as indicated by the broadcast BI7, information type 8.

3.5.15 RQ44 [Margin Underlying Real Time Price QUERY]

3.5.15.1 Fingerprint

QUERY properties	
transaction type	RQ44
calling sequence	omniapi_query_ex
struct name	query_realtime_ulg_price
facility	EP4
partitioned	false
answers	RA44

ANSWER properties	
transaction type	RA44
struct name	answer_realtime_ulg_price
segmented	true

3.5.15.2 Purpose

This query contains real time underlying prices.

3.5.15.3 Structure

The RQ44 QUERY has the following structure:

```
struct query_realtime_ulg_price {
   struct transaction_type
   struct series  // Named struct no: 50000
   UINT16_T segment_number_n  // Segment Number
   char[8] date_s  // Date
   char[2] filler_2_s  // Filler
}
```

3.5.15.4 Usage and conditions

Series

All components in the Series field except the **Commodity Code** field should always be filled with zeros. The Commodity Code component could either be a specific commodity number, or zero. Zero means that all underlyings will be returned.

3.5.15.5 Answer Structure

The RA44 ANSWER has the following structure:

```
struct answer_realtime_ulg_price {
   struct transaction type
   UINT16_T segment_number_n  // Segment Number
   UINT16_T items_n  // Items
   Array ITEM [max no: 300] {
      UINT32_T bid_price_i  // Bid Price
      UINT32_T ask_price_i  // Ask Price
      INT32_T last_paid_i  // Last, Paid
      UINT16_T commodity_n  // Commodity Code
      UINT8_T bid_theo_c  // Bid, Theoretical Mark
      UINT8_T ask_theo_c  // Ask, Theoretical Mark
      UINT8_T last_theo_c  // Last Paid, Theoretical Mark
      char[3] filler_3_s  // Filler
   }
}
```

3.5.16 RQ71 [Margin Simulation QUERY]

3.5.16.1 Fingerprint

QUERY properties	
transaction type	RQ71
calling sequence	omniapi_query_ex
struct name	query_margin_simulation
facility	EP4
partitioned	false
answers	RA71

ANSWER properties	
transaction type	RA71
struct name	answer_margin_simulation
segmented	true

3.5.16.2 Purpose

This query is used for simulating margin requirements. It is possible to calculate indicative margin requirements for a specific account with current prices and positions plus a list of supplied trades. It is also possible not to use any existing position, but to supply all trades used in the query.

3.5.16.3 Structure

The RQ71 QUERY has the following structure:

```
struct query_margin_simulation {
   struct transaction type
   struct series  // Named struct no: 50000
   struct account
   UINT16_T segment_number_n  // Segment Number
   UINT16_T qry_segment_number_n  // Segment Number, Query
   UINT16_T items_n  // Items
   UINT8_T pos_sim_c  // Positions, Simulated
   UINT8_T price_sim_c  // Prices Simulated
   UINT8_T vol_sim_c  // Volatility Simulated
   UINT8_T output_level_c  // Output Level
   UINT8_T last_qry_segment_c  // Last, Query Segment
   UINT8_T added_trade_sim_c  // Added Trades Simulated
   char[8] date_s  // Date
   UINT8_T series_exp_today_sim_c  // Series expiring today simulated
   UINT8_T fut_pl_sim_c  // Futures profit/loss Simulated
   char[32] sub_user_s  // Sub User
   char[3] margin_class_s  // Margin class
   char[3] filler_3_s  // Filler
   Array ITEM [max no: 1000] {
      UINT8_T item_type_c  // Item Type
      char[3] filler_3_s  // Filler
      struct series  // Named struct no: 50000
      INT64_T sim_qty_q  // Quantity, Simulation
      INT32_T trade_price_sim_i  // Trade Price, Simulated
      INT32_T reserved_i  // Reserved
      char[8] closing_date_s  // Date, Closing
      char[8] date_settlement_s  // Date, Settlement
      char[8] reserved_8_s  // Reserved
   }
}
```

3.5.16.4 Usage and conditions

Series

should be filled with zeros.

Date

must be set to current business date.

Account

may be filled with a specific account or may be left blank.

Prices Simulated
Volatilities Simulated
Last Query Segment
Segment Number, Query

must all be set to 1 .

Output Level

specifies which amount of results that is desired. If Positions Simulated = 2, then Output level must be set to 1.

Item Number

record specifies how many items that are provided in the query.

The **Item type, Simulation Query** field specifies what type of input this item contains. It can take the following values:

value	type
1	Specify market to use. If no item with type 1 is provided, all markets are used. It is possible to use 2 markets, by providing two items with item type = 1
2	Bought trade
3	Sold trade
4	Payment
5	Bought Delivery
6	Sold Delivery

Items with item type 1:

- The Series field should be filled in with Country Number and Market code
- The other fields are not used

Items with item type 2 or 3:

- The Series field should contain the series used
- The Quantity, Simulation field contains the quantity desired. Negative numbers are allowed, meaning reduce existing position by the number specified.
- The Trade Price, Simulated field is used if the Series is a future, forward, FRA, or a T/N swap. In that case, the field should contain the price of the trade.
- The fields Date, Closing and Date, Settlement are not used.

Items with item type 4:

- The Series field should contain the series used
- The Quantity, Simulation field contains the payment desired
- The other fields are not used

Items with item type 5 or 6:

- The Series field should contain the series used
- The Quantity, Simulation field contains the quantity desired. Negative numbers are allowed, meaning reduce existing delivery by the number specified.
- The Trade Price, Simulated field should contain the amount in money for 1 delivered unit.
- The Date, Closing field should contain the closing date of the corresponding derivative.
- The Date, Settlement field should contain the settlement date of the delivery.

Note: Closing trades may be entered by using trades with negative quantity.

Note: If negative quantity is used for a trade or a delivery, the transaction will end with an error if there is no position/delivery present for the series used.

Note: If Positions Simulated = 2, then the only items allowed are those with Item type = 1 (that is 2-6 are not allowed).

3.5.16.5 Return Codes

The error handling in this query is as follows:

cstatus	txstat	
Successful	RI_OMN_NORMAL	Successful completion
Successful	Other value than RI_OMN_NORMAL	Calculations failed

Please refer to the **Error Messages Reference Manual** for the meaning of error codes in txstat. In case of failure, additional information is available in the Failure Reason field of the answer struct.

3.5.16.6 Answer Structure

The RA71 ANSWER has the following structure:

```
struct answer_margin_simulation {
   struct transaction_type
   UINT16_T segment_number_n  // Segment Number
   UINT16_T items_n  // Items
   char[160] failure_reason_s  // Failure Reason
   char[40] filler_40_s  // Filler
   Array ITEM [max no: 500] {
      INT64_T market_margin_q  // Margin Requirements, Market
      INT64_T risk_margin_q  // Margining Requirements, Risk
      char[3] market_currency_s  // Currency, Market
      char[3] risk_currency_s  // Currency, Risk
      UINT8_T sim_item_type_c  // Item type, Simulation Answer
      CHAR filler_1_s  // Filler
      INT64_T nbr_held_q  // Held
      INT64_T nbr_written_q  // Written
      INT64_T market_value_q  // Market Value
```

```
        INT64_T naked_margin_q  // Margin Requirements, Naked
        struct series  // Named struct no: 50000
        UINT32_T bid_price_i  // Bid Price
        UINT32_T ask_price_i  // Ask Price
        INT32_T marg_price_i  // Margin, Settlement Price
        INT32_T fixing_value_i  // Fixing Value
        INT32_T val_ivl_mid_i  // Valuation Interval, Mid
        INT32_T val_ivl_low_i  // Valuation Interval, Low
        INT32_T val_ivl_high_i  // Valuation Interval, High
        UINT16_T dec_in_price_n  // Decimals, Price
        char[2] filler_2_s  // Filler
        char[8] filler_8_s  // Filler
    }
}
```

3.5.16.7 Answer, comments

The response received is a list of indicative margin requirements per instrument currency. The results are also translated to the risk currency of the account specified in the query. If a blank account was specified, the translation will be to the risk currency of the member putting the query.

The contents of each item are dependent on the value of the field Item Type, Simulation Answer.

The items of different type come in the following order:

1	Item type 1	
2	Item type 2-6 mixed	only present if output level >= 2
3	Item type 7	only present if output level = 3
4	Item type 8	only present if output level = 3 and if options are present

Items type 1 contain sum margin requirement per currency. The following fields are used:

- Margining Requirements, Market
- Margining Requirements, Risk
- Currency, Market
- Currency, Risk

Items type 2 contain individual margin requirement for a single open position. The following fields are used:

- Series
- Held
- Written
- Market Value
- Margining Requirements, Market
- Margining Requirements, Naked
- Currency, Market

Items type 3 contain individual margin requirement for a single delivery position. The following fields are used:

- Series

- Held
- Written
- Margining Requirements, Market
- Margining requirements, naked
- Currency, Market

Items type 4 contain individual margin requirement for a single payment position. The following fields are used:

- Series
- Margining Requirements, Market
- Margining requirements, naked

  Note: Always equal to Margining Requirements, Market

- Currency, Market

Items type 5 contain sum margin requirement of open and delivery positions for an underlying. The following fields are used:

- Series

  This is really an underlying, so it is only the commodity component of the struct that not equals zero.
- Margin Settlement Price

  This equals the “Based on price”
- Margining Requirements, Market
- Margining requirements, naked
- Currency, Market
- Decimals, price

  Number of decimals used in Margin Settlement Price

Items type 6 contain sum margin requirement of payment positions for an underlying. The following fields are used:

- Series

  This is really an underlying, so it is only the commodity component of the struct that not equals zero.
- Margining Requirements, Market
- Margining requirements, naked

  Note: Always equal to Margining Requirements, Market

- Currency, Market

Items type 7 contain prices and valuation intervals used in the calculations. The following fields are used:

- Series
- Bid
- Ask
- Margin Settlement Price

- Fixing value
- Valuation interval, mid
- Valuation interval, low
- Valuation interval, high
- Currency, Market
- Decimals, price

  Contains number of decimals used for valuation interval mid/low/high.

  Note: It does NOT contain number of decimals for bid/ask/margin settlement price/fixing.

Items type 8 contain volatilities and naked margin requirements for options used in the calculations. The following fields are used:

- Series
- Margining requirements, naked

  Contains margin requirement of one single written option.
- Bid

  This contains closing volatility for held options.
- Valuation interval, mid

  This contains closing volatility for written options.
- Ask

  This contains low volatility for held options.
- Valuation interval, low

  This contains low volatility for written options.
- Margin Settlement Price

  This contains high volatility for held options.
- Valuation interval, high

  This contains high volatility for written options.

Note: All volatilities for item type 8 come as percentages with 4 decimals.

4 Common Structures

4.1 ACCOUNT

```
struct account {
   char[2] country_id_s  // Name, Country
   char[5] ex_customer_s  // Customer, Identity
   char[10] account_id_s  // Account, Identity
   char[3] filler_3_s  // Filler
}
```

4.2 ACCOUNT_DATA

```
struct account_data {
   struct account
   struct countersign
   struct prop_trade_account
   struct prop_deliv_account
   struct prop_pos_account
   struct prop_margin_account
   struct sink_account
   struct prop_origin_account
   struct prop_call_account
   char[3] risk_currency_s  // Currency, Risk
   INT32_T rank_class_i  // Risk Ranking Class
   char[8] modified_date_s  // Date, Modified
   char[6] modified_time_s  // Time, Modified
   char[8] created_date_s  // Date, Created
   char[6] created_time_s  // Time, Created
   char[4] investor_type_s  // Investor Type
   char[4] nationality_s  // Nationality
   char[20] account_text_s  // Account Text
   char[34] ext_acc_id_s  // External Account ID
   char[15] ext_acc_controller_s  // External Account Controller
   char[12] ext_acc_registrar_s  // External Account Registrar
   char[16] org_number_s  // Organization number
   char[32] account_alias_s  // Account alias
   char[15] diary_number_s  // Diary Number
   char[12] acc_type_s  // Account Type
   char[12] fee_type_s  // Account Fee Type
   char[12] cust_bank_id_s  // Custodian Bank
   UINT8_T acc_state_c  // Account State
   UINT8_T read_access_c  // Read Access
   UINT8_T auto_net_c  // Auto Netting
   UINT8_T risk_cur_conv_c  // Risk, Currency Conversion
   UINT8_T risk_margin_net_c  // Risk, Margin Net
   UINT8_T acc_allow_nov_c  // Novation Allowed
   char[2] filler_2_s  // Filler
}
```

4.3 ANSWER_HDR

```
struct answer_hdr {
   struct transaction_type
   UINT16_T items_n  // Items
   UINT16_T size_n  // Size
}
```

4.4 ANSWER_SEGMENT_HDR

```
struct answer_segment_hdr {
   struct transaction_type
   UINT16_T items_n  // Items
   UINT16_T size_n  // Size
   UINT16_T segment_number_n  // Segment Number
   char[2] filler_2_s  // Filler
}
```

4.5 BROADCAST_HDR

```
struct broadcast_hdr {
   struct broadcast_type
   UINT16_T items_n  // Items
   UINT16_T size_n  // Size
}
```

4.6 BROADCAST_SEGMENT_HDR

```
struct broadcast_segment_hdr {
   struct broadcast_type
   UINT16_T items_n  // Items
   UINT16_T size_n  // Size
   UINT16_T segment_number_n  // Segment Number
   char[2] filler_2_s  // Filler
}
```

4.7 BROADCAST_TYPE

```
struct broadcast_type {
   CHAR central_module_c  // Central Module
   CHAR server_type_c  // Server Type
   UINT16_T transaction_number_n  // Transaction Type Number
}
```

4.8 CL_DELIVERY_API

```
struct cl_delivery_api {
   struct account
   struct delivery_account {
      char[2] country_id_s  // Name, Country
      char[5] ex_customer_s  // Customer, Identity
      char[10] account_id_s  // Account, Identity
      char[3] filler_3_s  // Filler
   }
   struct series
   struct deliv_base
   INT64_T deliv_base_quantity_q  // Quantity, Delivery Base
   INT64_T delivery_quantity_q  // Quantity, Delivery
   INT32_T delivery_number_i  // Delivery, Number
   INT32_T key_number_i  // Key Number
   INT32_T delivery_origin_i  // Delivery Origin
   INT32_T class_no_i  // Class Number
   INT32_T sequence_number_i  // Sequence Number
   INT32_T event_type_i  // Stimuli Event
   INT32_T original_delivery_number_i  // Original, Delivery Number
   INT32_T original_key_number_i  // Original, Key Number
   UINT32_T delivery_unit_u  // Delivery Unit
   UINT32_T delivery_properties_u  // Delivery Properties
   UINT32_T propagation_u  // Propagation
   char[8] settlement_date_s  // Date, Settlement
   char[8] date_s  // Date
   char[24] dvp_account_s  // DVP Account
   char[8] original_date_s  // Original Date
   char[32] passthrough_s  // Passthrough Information
   UINT8_T delivery_type_c  // Delivery, Type
   UINT8_T originator_type_c  // Originator Type
   UINT8_T delivery_state_c  // Delivery, State
   UINT8_T bought_or_sold_c  // Bought or Sold
   CHAR ext_trade_fee_type_c  // External Trade, Fee Type
   CHAR filler_1_s  // Filler
   char[2] giving_up_exchange_s  // Giving Up Exchange
   char[8] settlement_instr_date_s  // Date, Settlement instruction
}
```

4.9 CL_GIVE_UP_API

```
struct cl_give_up_api {
   struct series
   struct account
   struct party
   INT32_T sequence_number_i  // Sequence Number
   INT32_T gup_reason_i  // Give Up, Broadcast Reason
   INT32_T give_up_number_i  // Give Up, Number
   INT64_T trade_quantity_i  // Quantity, Trade
   INT32_T deal_price_i  // Price, Deal
   INT32_T trade_number_i  // Trade Number
```

```
   INT32_T commission_i  // Commission
   UINT8_T bought_or_sold_c  // Bought or Sold
   UINT8_T state_c  // State
   char[8] created_date_s  // Date, Created
   char[6] created_time_s  // Time, Created
   char[30] give_up_text_s  // Give Up, Free Text
   char[8] asof_date_s  // Date, As Of
   char[6] asof_time_s  // Time, As Of
   char[8] orig_clearing_date_s  // Clearing Date, Original
   UINT8_T old_trade_c  // Old Trade Indicator
   CHAR ext_trade_fee_type_c  // External Trade, Fee Type
   UINT8_T deal_source_c  // Deal Source
   UINT8_T reserved_prop_c  // Reserved Properties
   char[8] clearing_date_s  // Clearing Date
   UINT32_T ext_trade_number_u  // Trade Number, External
   UINT32_T orig_ext_trade_number_u  // Trade Number, Original External
   UINT8_T trade_venue_c  // Trade venue
   char[3] filler_3_s  // Filler
}
```

4.10 CL_TRADE_CHANGE_API

```
struct cl_trade_change_api {
   struct series
   INT32_T trade_number_i  // Trade Number
   INT32_T sequence_number_i  // Sequence Number
   UINT8_T trade_state_c  // Trade, State
   UINT8_T le_state_c  // Type, Legal Event
   UINT8_T give_up_state_c  // Give Up, State
   UINT8_T instance_c  // Instance, Number
   INT64_T rem_quantity_i  // Quantity, Remaining
   char[8] modified_date_s  // Date, Modified
   char[6] modified_time_s  // Time, Modified
   char[2] filler_2_s  // Filler
   UINT32_T big_attention_u  // Big Attention
}
```

4.11 COMBO_SERIES

```
struct combo_series {
   UINT8_T country_c  // Country Number
   UINT8_T market_c  // Market Code
   UINT8_T instrument_group_c  // Instrument Group
   UINT8_T modifier_c  // Modifier
   UINT16_T commodity_n  // Commodity Code
   UINT16_T expiration_date_n  // Date, Expiration
   INT32_T strike_price_i  // Strike Price
}
```

4.12 COUNTERSIGN

```
struct countersign {
   char[2] country_id_s  // Name, Country
   char[5] ex_customer_s  // Customer, Identity
   CHAR filler_1_s  // Filler
}
```

4.13 COUNTERSIGN_CODE

```
struct countersign_code {
   char[2] country_id_s  // Name, Country
   char[5] ex_customer_s  // Customer, Identity
   char[5] user_id_s  // User
}
```

4.14 DELIV_BASE

```
struct deliv_base {
   UINT8_T country_c  // Country Number
   UINT8_T market_c  // Market Code
   UINT8_T instrument_group_c  // Instrument Group
   UINT8_T modifier_c  // Modifier
   UINT16_T commodity_n  // Commodity Code
   UINT16_T expiration_date_n  // Date, Expiration
   INT32_T strike_price_i  // Strike Price
}
```

4.15 EX_USER_CODE

```
struct ex_user_code {
   char[2] country_id_s  // Name, Country
   char[5] ex_customer_s  // Customer, Identity
   char[5] user_id_s  // User
}
```

4.16 GIVE_UP_MEMBER

```
struct give_up_member {
   char[2] country_id_s  // Name, Country
   char[5] ex_customer_s  // Customer, Identity
   CHAR filler_1_s  // Filler
}
```

4.17 ITEM_HDR

```
struct item_hdr {
   UINT16_T items_n  // Items
   UINT16_T size_n  // Size
}
```

4.18 MATCH_ID

```
struct match_id {
   UINT64_T execution_event_nbr_u  // Execution number
   UINT32_T match_group_nbr_u  // Match group number, group inside an execution
   UINT32_T match_item_nbr_u  // Match Item Number
}
```

4.19 NEW_ACCOUNT

```
struct new_account {
   char[2] country_id_s  // Name, Country
   char[5] ex_customer_s  // Customer, Identity
   char[10] account_id_s  // Account, Identity
   char[3] filler_3_s  // Filler
}
```

4.20 NEW_SERIES

```
struct new_series {
   UINT8_T country_c  // Country Number
   UINT8_T market_c  // Market Code
   UINT8_T instrument_group_c  // Instrument Group
   UINT8_T modifier_c  // Modifier
   UINT16_T commodity_n  // Commodity Code
   UINT16_T expiration_date_n  // Date, Expiration
   INT32_T strike_price_i  // Strike Price
}
```

4.21 OLD_SERIES

```
struct old_series {
   UINT8_T country_c  // Country Number
   UINT8_T market_c  // Market Code
   UINT8_T instrument_group_c  // Instrument Group
   UINT8_T modifier_c  // Modifier
   UINT16_T commodity_n  // Commodity Code
   UINT16_T expiration_date_n  // Date, Expiration
```

```
    INT32_T strike_price_i  // Strike Price
}
```

4.22 ORDER

```
struct order {
    struct series
    struct trading_code
    struct order_var
    struct ex_user_code
    struct give_up_member
    CHAR[32] exchange_info_s  // Exchange, Information
    UINT32_T order_index_u  // Order Index
    UINT16_T transaction_number_n  // Transaction Type Number
    UINT8_T change_reason_c  // Change Reason
    CHAR filler_1_s  // Filler
}
```

4.23 ORDER_NO_ID

```
struct order_no_id {
    struct series
    INT64_T mp_quantity_i  // Quantity
    INT32_T premium_i  // Premium
    UINT32_T block_n  // Block Size
    UINT16_T exch_order_type_n  // Order Type, Exchange
    UINT8_T bid_or_ask_c  // Bid or Ask
    CHAR filler_1_s  // Filler
}
```

4.24 ORDER_VAR

```
struct order_var {
    INT64_T mp_quantity_i  // Quantity
    INT32_T premium_i  // Premium
    UINT32_T block_n  // Block Size
    UINT16_T time_validity_n  // Validity Time
    UINT16_T exch_order_type_n  // Order Type, Exchange
    char[10] ex_client_s  // Client
    char[15] customer_info_s  // Customer, Information
    UINT8_T open_close_req_c  // Open  Close Request
    UINT8_T bid_or_ask_c  // Bid or Ask
    UINT8_T ext_t_state_c  // Trade Report Type
    UINT8_T order_type_c  // Order Type
    UINT8_T stop_condition_c  // Stop Condition
    char[2] filler_2_s  // Filler
}
```

4.25 ORIGINATOR_TRADING_CODE

```
struct originator_trading_code {
   char[2] country_id_s  // Name, Country
   char[5] ex_customer_s  // Customer, Identity
   char[5] user_id_s  // User
}
```

4.26 ORIG_SERIES

```
struct orig_series {
   UINT8_T country_c  // Country Number
   UINT8_T market_c  // Market Code
   UINT8_T instrument_group_c  // Instrument Group
   UINT8_T modifier_c  // Modifier
   UINT16_T commodity_n  // Commodity Code
   UINT16_T expiration_date_n  // Date, Expiration
   INT32_T strike_price_i  // Strike Price
}
```

4.27 PARTITION_HIGH

```
struct partition_high {
   UINT8_T country_c  // Country Number
   UINT8_T market_c  // Market Code
   UINT8_T instrument_group_c  // Instrument Group
   UINT8_T modifier_c  // Modifier
   UINT16_T commodity_n  // Commodity Code
   UINT16_T expiration_date_n  // Date, Expiration
   INT32_T strike_price_i  // Strike Price
}
```

4.28 PARTITION_LOW

```
struct partition_low {
   UINT8_T country_c  // Country Number
   UINT8_T market_c  // Market Code
   UINT8_T instrument_group_c  // Instrument Group
   UINT8_T modifier_c  // Modifier
   UINT16_T commodity_n  // Commodity Code
   UINT16_T expiration_date_n  // Date, Expiration
   INT32_T strike_price_i  // Strike Price
}
```

4.29 PARTY

```
struct party {
   char[2] country_id_s  // Name, Country
   char[5] ex_customer_s  // Customer, Identity
   CHAR filler_1_s  // Filler
}
```

4.30 POS_ACCOUNT

```
struct pos_account {
   char[2] country_id_s  // Name, Country
   char[5] ex_customer_s  // Customer, Identity
   char[10] account_id_s  // Account, Identity
   char[3] filler_3_s  // Filler
}
```

4.31 POS_INFO_UPDATE_API

```
struct pos_info_update_api {
   struct series
   struct account
   INT64_T deny_exercise_q  // Deny Exercise
   INT64_T qty_closed_out_q  // Quantity, Closed out
   UINT32_T quantity_cover_u  // Quantity Cover
   char[8] modified_date_s  // Date, Modified
   char[6] modified_time_s  // Time, Modified
   UINT8_T reserved_prop_c  // Reserved Properties
   CHAR filler_1_s  // Filler
}
```

4.32 PRIO_CROSSING

```
struct prio_crossing {
   INT64_T mp_quantity_i  // Quantity
   INT32_T premium_i  // Premium
   struct buy_give_up_member  // Of type: GIVE_UP_MEMBER
   struct sell_give_up_member  // Of type: GIVE_UP_MEMBER
   UINT32_T block_n  // Block Size
   UINT16_T exch_order_type_n  // Order Type, Exchange
   char[15] buy_customer_info_s  // Customer, Information ; Of type:
CUSTOMER_INFO_S
   char[15] sell_customer_info_s  // Customer, Information ; Of type:
CUSTOMER_INFO_S
   char[10] buy_ex_client_s  // Client ; Of type: EX_CLIENT_S
   char[10] sell_ex_client_s  // Client ; Of type: EX_CLIENT_S
   CHAR[32] exchange_info_s  // Exchange, Information
```

```
   UINT8_T order_type_c  // Order Type
   char[3] filler_3_s  // Filler
}
```

4.33 PROP_CALL_ACCOUNT

```
struct prop_call_account {
   char[2] country_id_s  // Name, Country
   char[5] ex_customer_s  // Customer, Identity
   char[10] account_id_s  // Account, Identity
}
```

4.34 PROP_DELIV_ACCOUNT

```
struct prop_deliv_account {
   char[2] country_id_s  // Name, Country
   char[5] ex_customer_s  // Customer, Identity
   char[10] account_id_s  // Account, Identity
}
```

4.35 PROP_MARGIN_ACCOUNT

```
struct prop_margin_account {
   char[2] country_id_s  // Name, Country
   char[5] ex_customer_s  // Customer, Identity
   char[10] account_id_s  // Account, Identity
}
```

4.36 PROP_ORIGIN_ACCOUNT

```
struct prop_origin_account {
   char[2] country_id_s  // Name, Country
   char[5] ex_customer_s  // Customer, Identity
   char[10] account_id_s  // Account, Identity
   char[3] filler_3_s  // Filler
}
```

4.37 PROP_POS_ACCOUNT

```
struct prop_pos_account {
   char[2] country_id_s  // Name, Country
   char[5] ex_customer_s  // Customer, Identity
   char[10] account_id_s  // Account, Identity
}
```

4.38 PROP_TRADE_ACCOUNT

```
struct prop_trade_account {
   char[2] country_id_s  // Name, Country
   char[5] ex_customer_s  // Customer, Identity
   char[10] account_id_s  // Account, Identity
}
```

4.39 QUERY_DELTA

```
struct query_delta {
   struct transaction_type
   struct series
   UINT16_T segment_number_n  // Segment Number
   char[2] filler_2_s  // Filler
   INT64_T download_ref_number_q  // Download Reference Number
   struct full_answer_timestamp  // Of type: TIME_SPEC
}
```

4.40 QUERY_HDR

```
struct query_hdr {
   struct transaction_type
   struct series
   UINT16_T items_n  // Items
   UINT16_T size_n  // Size
}
```

4.41 SEARCH_SERIES

```
struct search_series {
   UINT8_T country_c  // Country Number
   UINT8_T market_c  // Market Code
   UINT8_T instrument_group_c  // Instrument Group
   UINT8_T modifier_c  // Modifier
   UINT16_T commodity_n  // Commodity Code
   UINT16_T expiration_date_n  // Date, Expiration
   INT32_T strike_price_i  // Strike Price
}
```

4.42 SERIES

```
struct series {
   UINT8_T country_c  // Country Number
   UINT8_T market_c  // Market Code
```

```
    UINT8_T instrument_group_c  // Instrument Group
    UINT8_T modifier_c  // Modifier
    UINT16_T commodity_n  // Commodity Code
    UINT16_T expiration_date_n  // Date, Expiration
    INT32_T strike_price_i  // Strike Price
}
```

4.43 SERIES_NEXT

```
struct series_next {
    UINT8_T country_c  // Country Number
    UINT8_T market_c  // Market Code
    UINT8_T instrument_group_c  // Instrument Group
    UINT8_T modifier_c  // Modifier
    UINT16_T commodity_n  // Commodity Code
    UINT16_T expiration_date_n  // Date, Expiration
    INT32_T strike_price_i  // Strike Price
}
```

4.44 SINK_ACCOUNT

```
struct sink_account {
    char[2] country_id_s  // Name, Country
    char[5] ex_customer_s  // Customer, Identity
    char[10] account_id_s  // Account, Identity
    char[3] filler_3_s  // Filler
}
```

4.45 STOP_SERIES

```
struct stop_series {
    UINT8_T country_c  // Country Number
    UINT8_T market_c  // Market Code
    UINT8_T instrument_group_c  // Instrument Group
    UINT8_T modifier_c  // Modifier
    UINT16_T commodity_n  // Commodity Code
    UINT16_T expiration_date_n  // Date, Expiration
    INT32_T strike_price_i  // Strike Price
}
```

4.46 SUB_ITEM_HDR

```
struct sub_item_hdr {
    UINT16_T named_struct_n  // Named Struct, Number
    UINT16_T size_n  // Size
}
```

4.47 TICK_SIZE

```
struct tick_size {
   INT32_T step_size_i  // Tick Size
   INT32_T lower_limit_i  // Premium/Price, Low Limit
   INT32_T upper_limit_i  // Premium/Price, High Limit
}
```

4.48 TIME_SPEC

```
struct time_spec {
   UINT32_T tv_sec  // Time in seconds
   INT32_T tv_nsec  // Time in nanoseconds
}
```

4.49 TRADING_CODE

```
struct trading_code {
   char[2] country_id_s  // Name, Country
   char[5] ex_customer_s  // Customer, Identity
   char[5] user_id_s  // User
}
```

4.50 TRANSACTION_TYPE

```
struct transaction_type {
   CHAR central_module_c  // Central Module
   CHAR server_type_c  // Server Type
   UINT16_T transaction_number_n  // Transaction Type Number
}
```

4.51 TRD_RPT_CUST

```
struct trd_rpt_cust {
   struct party
   char[10] ex_client_s  // Client
   char[15] customer_info_s  // Customer, Information
   CHAR[32] exchange_info_s  // Exchange, Information
   UINT8_T open_close_req_c  // Open  Close Request
   UINT16_T exch_order_type_n  // Order Type, Exchange
   struct give_up_member
}
```

4.52 TRD_RPT_PART

```
struct trd_rpt_part {
   struct party
   char[10] ex_client_s  // Client
   char[15] customer_info_s  // Customer, Information
   CHAR[32] exchange_info_s  // Exchange, Information
   UINT8_T open_close_req_c  // Open  Close Request
   char[2] filler_2_s  // Filler
}
```

4.53 UPPER_LEVEL_SERIES

```
struct upper_level_series {
   UINT8_T country_c  // Country Number
   UINT8_T market_c  // Market Code
   UINT8_T instrument_group_c  // Instrument Group
   UINT8_T modifier_c  // Modifier
   UINT16_T commodity_n  // Commodity Code
   UINT16_T expiration_date_n  // Date, Expiration
   INT32_T strike_price_i  // Strike Price
}
```

4.54 USER_CODE

```
struct user_code {
   char[2] country_id_s  // Name, Country
   char[5] ex_customer_s  // Customer, Identity
   char[5] user_id_s  // User
}
```

4.55 WHOSE

```
struct whose {
   struct trading_code
   char[10] ex_client_s  // Client
   char[2] filler_2_s  // Filler
}
```

5 Named Structs Involved in VIMs

Named structs used in the variable information messages (VIM) included in this message reference are listed here in numerical order.

5.1 CL_TRADE_BASE_API (3)

```
struct cl_trade_base_api {
   struct trading_code
   struct series  // Named struct no: 50000
   struct give_up_member  // Named struct no: 50002
   QUAD_WORD order_number_u  // Order Number
   INT32_T sequence_number_i  // Sequence Number
   INT32_T trade_number_i  // Trade Number
   INT32_T deal_price_i  // Price, Deal
   INT64_T trade_quantity_i  // Quantity, Trade
   struct account
   char[15] customer_info_s  // Customer, Information
   UINT8_T bought_or_sold_c  // Bought or Sold
   UINT8_T deal_source_c  // Deal Source
   UINT8_T open_close_req_c  // Open  Close Request
   UINT8_T trade_type_c  // Type, Trade
   UINT8_T le_state_c  // Type, Legal Event
   struct user_code
   char[8] created_date_s  // Date, Created
   char[6] created_time_s  // Time, Created
   char[8] asof_date_s  // Date, As Of
   char[6] asof_time_s  // Time, As Of
   char[8] modified_date_s  // Date, Modified
   char[6] modified_time_s  // Time, Modified
   UINT8_T trade_state_c  // Trade, State
   UINT8_T attention_c  // Attention
   INT32_T deal_number_i  // Deal Number
   UINT32_T global_deal_no_u  // Global Deal Number
   INT32_T orig_trade_number_i  // Trade Number, Original
   struct orig_series
   CHAR[32] exchange_info_s  // Exchange, Information
   UINT32_T big_attention_u  // Big Attention
   char[8] clearing_date_s  // Clearing Date
   struct execution_timestamp  // Of type: TIME_SPEC
   UINT8_T trade_venue_c  // Trade venue
   UINT8_T instance_c  // Instance, Number
   UINT16_T exch_order_type_n  // Order Type, Exchange
   struct party
   UINT16_T trade_rep_code_n  // Trade Report Code
   char[2] filler_2_s  // Filler
   struct match_id
}
```

5.2 CL_TRADE_SECUR_PART (20)

```
struct cl_trade_secur_part {
   struct countersign_code
   struct new_series
   struct party
   struct pos_account
   struct combo_series
   INT64_T nbr_held_q  // Held
   INT64_T nbr_written_q  // Written
   INT64_T total_held_q  // Held, Total
   INT64_T total_written_q  // Written Total
   INT32_T ext_seq_nbr_i  // External Clearinghouse, Sequence Number
   INT32_T ext_status_i  // Return Status
   INT64_T rem_quantity_i  // Quantity, Remaining
   INT64_T quantity_i  // Quantity
   UINT32_T ext_trade_number_u  // Trade Number, External
   UINT32_T orig_ext_trade_number_u  // Trade Number, Original External
   INT32_T residual_i  // Residual
   INT32_T give_up_number_i  // Give Up, Number
   INT32_T commission_i  // Commission
   INT32_T combo_deal_price_i  // Combo deal price
   char[8] clearing_date_s  // Clearing Date
   char[32] passthrough_s  // Passthrough Information
   char[10] ex_client_s  // Client
   CHAR ext_trade_fee_type_c  // External Trade, Fee Type
   UINT8_T give_up_state_c  // Give Up, State
   char[2] reserved_2_s  // Reserved
   UINT8_T orig_trade_type_c  // Trade Type, Original
   UINT8_T open_close_c  // Open or Closed
   CHAR reserved_1_c  // Reserved
   UINT8_T account_type_c  // Account Type
   UINT8_T instigant_c  // Instigant
   UINT8_T cab_price_ind_c  // Cabinet Price Indicator
}
```

5.3 OB_LEVELS_SEQUENCE_NUMBER (33001)

```
struct ob_levels_sequence_number {
   UINT32_T sequence_number_u  // Sequence Number
}
```

5.4 OB_LEVELS_ID (33002)

```
struct ob_levels_id {
   struct series  // Named struct no: 50000
   UINT32_T block_n  // Block Size
}
```

5.5 OB_LEVELS_PRICE_VOLUMES (33003)

```
struct ob_levels_price_volumes {
   UINT16_T bid_mask_n  // Mask, Bid
   UINT16_T ask_mask_n  // Mask, Ask
   UINT8_T premium_levels_c  // Premium Levels
   UINT8_T demands_populated_c  // Demands, Populated
   UINT8_T items_c  // Item
   CHAR filler_1_s  // Filler
   Array ITEM [max no: 32] {
      INT32_T premium_i  // Premium
      INT64_T demand_u  // Demand
   }
}
```

5.6 OB_LEVELS_ORDER_NUMBER (33004)

```
struct ob_levels_order_number {
   QUAD_WORD order_number_bid_u  // Order Number, Bid
   QUAD_WORD order_number_ask_u  // Order Number, Ask
}
```

5.7 OB_LEVELS_TOTAL_QUANTITY (33005)

```
struct ob_levels_total_quantity {
   INT64_T total_quantity_bid_u  // Quantity, Total Bid
   INT64_T total_quantity_ask_u  // Quantity, Total Ask
}
```

5.8 OB_LEVELS_PRICE (33006)

```
struct ob_levels_price {
   UINT16_T bid_mask_n  // Mask, Bid
   UINT16_T ask_mask_n  // Mask, Ask
   UINT8_T premium_levels_c  // Premium Levels
   UINT8_T demands_populated_c  // Demands, Populated
   UINT8_T items_c  // Item
   CHAR filler_1_s  // Filler
   Array ITEM [max no: 32] {
      INT32_T premium_i  // Premium
   }
}
```

5.9 OB_LEVELS_HIDDEN_QUANTITY (33007)

```
struct ob_levels_hidden_quantity {
   UINT8_T undisclosed_bid_volume_c  // Undisclosed Bid Volume
   UINT8_T undisclosed_ask_volume_c  // Undisclosed Ask Volume
   char[2] filler_2_s  // Filler
}
```

5.10 OB_LEVELS_QUERY_DATA (33020)

```
struct ob_levels_query_data {
   UINT16_T segment_number_n  // Segment Number
   char[2] filler_2_s  // Filler
}
```

5.11 OB_LEVELS_CLOSING (33031)

```
struct ob_levels_closing {
   INT32_T closing_price_i  // Price, Closing
   INT64_T open_balance_u  // Open Interest
}
```

5.12 OB_LEVELS_NEXT_QUERY (33032)

```
struct ob_levels_next_query {
   UINT16_T segment_number_n  // Segment Number
   UINT8_T instance_c  // Instance, Number
   UINT8_T instance_next_c  // Next Instance Number
   struct series_next
}
```

5.13 OB_LEVELS_NO_OF_ORDERS (33033)

```
struct ob_levels_no_of_orders {
   UINT16_T bid_mask_n  // Mask, Bid
   UINT16_T ask_mask_n  // Mask, Ask
   UINT32_T total_no_of_bid_orders_u  // Bid Orders, Total Number
   UINT32_T total_no_of_ask_orders_u  // Ask Orders, Total Number
   UINT8_T premium_levels_c  // Premium Levels
   char[2] filler_2_s  // Filler
   UINT8_T items_c  // Item
   Array ITEM [max no: 32] {
      UINT32_T no_of_orders_u  // Orders, Number of
   }
}
```

5.14 MARKET_INFO_BASE (33034)

```
struct market_info_base {
   INT32_T opening_price_i  // Price, First
   INT32_T high_price_i  // Price, High
   INT32_T low_price_i  // Price, Low
   INT32_T last_price_i  // Price, Last
   INT64_T volume_u  // Volume
   INT64_T turnover_u  // Turnover
   UINT32_T number_of_deals_u  // Deals, Number
   char[6] hhmmss_s  // Time, External
   CHAR trend_indicator_c  // Trend Indicator
   UINT8_T deal_source_c  // Deal Source
}
```

5.15 MARKET_INFO_SERIES (33038)

```
struct market_info_series {
   struct series  // Named struct no: 50000
   INT32_T reserved_i  // Reserved
   UINT8_T all_or_none_c  // All Or None
   char[3] filler_3_s  // Filler
}
```

5.16 OB_LEVELS_UNDISCLOSED_QUANTITY (33041)

```
struct ob_levels_undisclosed_quantity {
   UINT16_T bid_mask_n  // Mask, Bid
   UINT16_T ask_mask_n  // Mask, Ask
}
```

5.17 MARKET_INFO_REASON (33043)

```
struct market_info_reason {
   UINT8_T edited_price_info_reason_c  // Reason for Edited Price Information
 update
   char[3] filler_3_s  // Filler
}
```

5.18 MARKET_INFO_HKE (33044)

```
struct market_info_hke {
   INT64_T turnover_value_q  // Turnover, Value
   INT64_T trade_reported_volume_u  // Volume, Trade Reported
}
```

5.19 HV_PRICE_2_TRANS (34001)

```
struct hv_price_2_trans {
   struct transaction_type
   struct series  // Named struct no: 50000
   struct give_up_member  // Named struct no: 50002
   QUAD_WORD order_number_bid_u  // Order Number, Bid
   QUAD_WORD order_number_ask_u  // Order Number, Ask
   INT32_T bid_premium_i  // Bid Premium
   INT32_T ask_premium_i  // Ask Premium
   INT64_T bid_quantity_i  // Quantity, Bid
   INT64_T ask_quantity_i  // Quantity, Ask
   INT64_T bid_total_volume_i  // Total Volume, Bid
   INT64_T ask_total_volume_i  // Total Volume, Ask
   UINT32_T block_n  // Block Size
   UINT16_T time_validity_n  // Validity Time
   char[10] ex_client_s  // Client
   UINT8_T order_type_c  // Order Type
   char[15] customer_info_s  // Customer, Information
   CHAR[32] exchange_info_s  // Exchange, Information
}
```

5.20 PRICE_2_TRANS (34002)

```
struct price_2_trans {
   struct transaction_type
   struct series  // Named struct no: 50000
   INT32_T bid_premium_i  // Bid Premium
   INT32_T ask_premium_i  // Ask Premium
   QUAD_WORD order_number_bid_u  // Order Number, Bid
   QUAD_WORD order_number_ask_u  // Order Number, Ask
   INT64_T bid_quantity_i  // Quantity, Bid
   INT64_T ask_quantity_i  // Quantity, Ask
   UINT32_T block_n  // Block Size
   UINT16_T time_validity_n  // Validity Time
   char[10] ex_client_s  // Client
}
```

5.21 PRICE_TRANS (34003)

```
struct price_trans {
   struct transaction_type
   struct series  // Named struct no: 50000
   struct order_var
   QUAD_WORD order_number_u  // Order Number
   struct give_up_member  // Named struct no: 50002
}
```

5.22 ORDER_TRANS (34004)

```
struct order_trans {
   struct transaction_type
   struct series  // Named struct no: 50000
   struct order_var
   CHAR[32] exchange_info_s  // Exchange, Information
   struct give_up_member  // Named struct no: 50002
}
```

5.23 HV_ORDER_TRANS (34005)

```
struct hv_order_trans {
   struct transaction_type
   struct series  // Named struct no: 50000
   struct order_var
   struct give_up_member  // Named struct no: 50002
   CHAR[32] exchange_info_s  // Exchange, Information
   INT64_T total_volume_i  // Total Volume
}
```

5.24 BLOCK_ORDER_TRANS (34006)

```
struct block_order_trans {
   struct transaction_type
   struct series  // Named struct no: 50000
   struct give_up_member  // Named struct no: 50002
   CHAR[32] exchange_info_s  // Exchange, Information
   UINT8_T items_c  // Item
   char[3] filler_3_s  // Filler
   Array ITEM [max no: 100] {
      struct series  // Named struct no: 50000
      struct order_var
      INT64_T total_volume_i  // Total Volume
   }
}
```

5.25 BLOCK_PRICE_TRANS (34007)

```
struct block_price_trans {
   struct transaction_type
   struct series  // Named struct no: 50000
   struct give_up_member  // Named struct no: 50002
   CHAR[32] exchange_info_s  // Exchange, Information
   char[15] customer_info_s  // Customer, Information
   UINT8_T items_c  // Item
   Array ITEM [max no: 14] {
```

```
        struct series  // Named struct no: 50000
        QUAD_WORD order_number_bid_u  // Order Number, Bid
        QUAD_WORD order_number_ask_u  // Order Number, Ask
        INT32_T bid_premium_i  // Bid Premium
        INT32_T ask_premium_i  // Ask Premium
        INT64_T bid_quantity_i  // Quantity, Bid
        INT64_T ask_quantity_i  // Quantity, Ask
        INT64_T bid_total_volume_i  // Total Volume, Bid
        INT64_T ask_total_volume_i  // Total Volume, Ask
        UINT32_T block_n  // Block Size
        UINT16_T time_validity_n  // Validity Time
        UINT8_T order_type_c  // Order Type
        char[10] ex_client_s  // Client
        UINT8_T delta_quantity_c  // Delta Quantity
        char[2] filler_2_s  // Filler
    }
}
```

5.26 ALTER_TRANS (34009)

```
struct alter_trans {
    struct transaction_type
    struct series  // Named struct no: 50000
    QUAD_WORD order_number_u  // Order Number
    struct order_var
}
```

5.27 HV_ALTER_TRANS (34010)

```
struct hv_alter_trans {
    struct transaction_type
    struct series  // Named struct no: 50000
    QUAD_WORD order_number_u  // Order Number
    struct order_var
    struct give_up_member  // Named struct no: 50002
    CHAR[32] exchange_info_s  // Exchange, Information
    INT64_T total_volume_i  // Total Volume
    UINT8_T delta_quantity_c  // Delta Quantity
    char[3] filler_3_s  // Filler
    INT64_T balance_quantity_i  // Balance Quantity
}
```

5.28 DELETE_TRANS (34011)

```
struct delete_trans {
    struct transaction_type
    struct series  // Named struct no: 50000
    QUAD_WORD order_number_u  // Order Number
    struct whose
```

```
   UINT8_T bid_or_ask_c  // Bid or Ask
   char[15] customer_info_s  // Customer, Information
   CHAR[32] exchange_info_s  // Exchange, Information
}
```

5.29 BROKER_TRANS (34013)

```
struct broker_trans {
   struct transaction_type
   struct series  // Named struct no: 50000
   struct order_var
   struct party
   CHAR[32] exchange_info_s  // Exchange, Information
}
```

5.30 TM_TRADE_RPT_TRANS (34014)

```
struct tm_trade_rpt_trans {
   struct transaction_type
   struct series  // Named struct no: 50000
   struct order_var
   struct party
   UINT16_T commodity_n  // Commodity Code
   UINT16_T expiration_date_n  // Date, Expiration
   INT32_T strike_price_i  // Strike Price
   CHAR[32] exchange_info_s  // Exchange, Information
}
```

5.31 COMBO_ACC_TRANS (34016)

```
struct combo_acc_trans {
   struct transaction_type
   struct series  // Named struct no: 50000
   struct order_var
   CHAR[32] exchange_info_s  // Exchange, Information
   QUAD_WORD order_number_u  // Order Number
   struct give_up_member  // Named struct no: 50002
}
```

5.32 STOP_ORDER_TRANS (34017)

```
struct stop_order_trans {
   struct transaction_type
   struct series  // Named struct no: 50000
   struct order_var
   struct stop_series
   INT32_T limit_premium_i  // Premium, Limit
```

```
   struct give_up_member  // Named struct no: 50002
   CHAR[32] exchange_info_s  // Exchange, Information
   INT64_T total_volume_i  // Total Volume
}
```

5.33 TRADE_REPORT_TRANS (34018)

```
struct trade_report_trans {
   struct transaction_type
   struct series  // Named struct no: 50000
   INT64_T mp_quantity_i  // Quantity
   INT32_T premium_i  // Premium
   UINT32_T block_n  // Block Size
   UINT8_T ext_t_state_c  // Trade Report Type
   char[3] filler_3_s  // Filler
   struct bid_side {
      struct trd_rpt_part
   }
   struct ask_side {
      struct trd_rpt_part
   }
}
```

5.34 PRIO_CROSSING_TRANS (34020)

```
struct prio_crossing_trans {
   struct transaction_type
   struct series  // Named struct no: 50000
   struct prio_crossing
}
```

5.35 TRADE_REPORT_1_TRANS (34021)

```
struct trade_report_1_trans {
   struct transaction_type
   struct series  // Named struct no: 50000
   struct order_var
   struct party
   CHAR[32] exchange_info_s  // Exchange, Information
   struct give_up_member  // Named struct no: 50002
   char[8] settlement_date_s  // Date, Settlement
   char[8] time_of_agreement_date_s  // Time of agreement, date part
   char[6] time_of_agreement_time_s  // Time of agreement, time part
   UINT8_T deferred_publication_c  // Deferred Publication
   CHAR filler_1_s  // Filler
}
```

5.36 TRADE_REPORT_2_TRANS (34022)

```
struct trade_report_2_trans {
   struct transaction_type
   struct series  // Named struct no: 50000
   INT64_T mp_quantity_i  // Quantity
   INT32_T premium_i  // Premium
   UINT32_T block_n  // Block Size
   char[8] settlement_date_s  // Date, Settlement
   char[8] time_of_agreement_date_s  // Time of agreement, date part
   char[6] time_of_agreement_time_s  // Time of agreement, time part
   UINT8_T ext_t_state_c  // Trade Report Type
   UINT8_T deferred_publication_c  // Deferred Publication
   struct bid_side  // Of type: TRD_RPT_CUST
   struct ask_side  // Of type: TRD_RPT_CUST
}
```

5.37 CPPX_INITIATION_TRANS (34023)

```
struct cppx_initiation_trans {
   struct transaction_type
   struct series  // Named struct no: 50000
   QUAD_WORD order_number_u  // Order Number
   struct prio_crossing
}
```

5.38 LONG_STOP_ORDER_TRANS (34024)

```
struct long_stop_order_trans {
   struct transaction_type
   struct series  // Named struct no: 50000
   struct order_var
   struct stop_series
   INT32_T limit_premium_i  // Premium, Limit
   struct give_up_member  // Named struct no: 50002
   CHAR[32] exchange_info_s  // Exchange, Information
   INT64_T total_volume_i  // Total Volume
}
```

5.39 INDICATIVE_QUOTE (34025)

```
struct indicative_quote {
   struct series  // Named struct no: 50000
   INT64_T buy_quantity_u  // Buy Quantity
   INT64_T sell_quantity_u  // Sell Quantity
   INT32_T buy_price_i  // Buy Price
   INT32_T sell_price_i  // Ask Price
```

```
    UINT8_T bid_quote_action  // Quote Action ; Of type: QUOTE_ACTION_C
    UINT8_T ask_quote_action  // Quote Action ; Of type: QUOTE_ACTION_C
    char[2] filler_2_s  // Filler
}
```

5.40 CPPX_CONFIRMATION_TRANS (34028)

```
struct cppx_confirmation_trans {
    struct transaction_type
    struct series  // Named struct no: 50000
    QUAD_WORD order_number_u  // Order Number
    struct prio_crossing
}
```

5.41 HV_PRICE_2_TRANS_P (34101)

```
struct hv_price_2_trans_p {
    struct transaction_type
    struct series  // Named struct no: 50000
    struct trading_code
    struct give_up_member  // Named struct no: 50002
    QUAD_WORD order_number_bid_u  // Order Number, Bid
    QUAD_WORD order_number_ask_u  // Order Number, Ask
    INT32_T bid_premium_i  // Bid Premium
    INT32_T ask_premium_i  // Ask Premium
    INT64_T bid_quantity_i  // Quantity, Bid
    INT64_T ask_quantity_i  // Quantity, Ask
    INT64_T bid_total_volume_i  // Total Volume, Bid
    INT64_T ask_total_volume_i  // Total Volume, Ask
    UINT32_T block_n  // Block Size
    UINT16_T time_validity_n  // Validity Time
    char[10] ex_client_s  // Client
    UINT8_T order_type_c  // Order Type
    char[15] customer_info_s  // Customer, Information
    CHAR[32] exchange_info_s  // Exchange, Information
}
```

5.42 PRICE_TRANS_P (34103)

```
struct price_trans_p {
    struct transaction_type
    struct series  // Named struct no: 50000
    struct trading_code
    struct order_var
    QUAD_WORD order_number_u  // Order Number
    struct give_up_member  // Named struct no: 50002
}
```

5.43 HV_ORDER_TRANS_P (34105)

```
struct hv_order_trans_p {
   struct transaction_type
   struct series  // Named struct no: 50000
   struct trading_code
   struct order_var
   struct give_up_member  // Named struct no: 50002
   CHAR[32] exchange_info_s  // Exchange, Information
   INT64_T total_volume_i  // Total Volume
}
```

5.44 BLOCK_ORDER_TRANS_P (34106)

```
struct block_order_trans_p {
   struct transaction_type
   struct series  // Named struct no: 50000
   struct trading_code
   struct give_up_member  // Named struct no: 50002
   CHAR[32] exchange_info_s  // Exchange, Information
   UINT8_T items_c  // Item
   char[3] filler_3_s  // Filler
   Array ITEM [max no: 100] {
      struct series  // Named struct no: 50000
      struct order_var
      INT64_T total_volume_i  // Total Volume
   }
}
```

5.45 BLOCK_PRICE_TRANS_P (34107)

```
struct block_price_trans_p {
   struct transaction_type
   struct series  // Named struct no: 50000
   struct trading_code
   struct give_up_member  // Named struct no: 50002
   CHAR[32] exchange_info_s  // Exchange, Information
   char[15] customer_info_s  // Customer, Information
   UINT8_T items_c  // Item
   Array ITEM [max no: 14] {
      struct series  // Named struct no: 50000
      QUAD_WORD order_number_bid_u  // Order Number, Bid
      QUAD_WORD order_number_ask_u  // Order Number, Ask
      INT32_T bid_premium_i  // Bid Premium
      INT32_T ask_premium_i  // Ask Premium
      INT64_T bid_quantity_i  // Quantity, Bid
      INT64_T ask_quantity_i  // Quantity, Ask
      INT64_T bid_total_volume_i  // Total Volume, Bid
      INT64_T ask_total_volume_i  // Total Volume, Ask
```

```
      UINT32_T block_n  // Block Size
      UINT16_T time_validity_n  // Validity Time
      UINT8_T order_type_c  // Order Type
      char[10] ex_client_s  // Client
      UINT8_T delta_quantity_c  // Delta Quantity
      char[2] filler_2_s  // Filler
   }
}
```

5.46 HV_ALTER_TRANS_P (34110)

```
struct hv_alter_trans_p {
   struct transaction_type
   struct series  // Named struct no: 50000
   struct trading_code
   QUAD_WORD order_number_u  // Order Number
   struct order_var
   struct give_up_member  // Named struct no: 50002
   CHAR[32] exchange_info_s  // Exchange, Information
   INT64_T total_volume_i  // Total Volume
   UINT8_T delta_quantity_c  // Delta Quantity
   char[3] filler_3_s  // Filler
   INT64_T balance_quantity_i  // Balance Quantity
}
```

5.47 DELETE_TRANS_P (34111)

```
struct delete_trans_p {
   struct transaction_type
   struct series  // Named struct no: 50000
   struct trading_code
   QUAD_WORD order_number_u  // Order Number
   struct whose
   UINT8_T bid_or_ask_c  // Bid or Ask
   char[15] customer_info_s  // Customer, Information
   CHAR[32] exchange_info_s  // Exchange, Information
}
```

5.48 BROKER_TRANS_P (34113)

```
struct broker_trans_p {
   struct transaction_type
   struct series  // Named struct no: 50000
   struct trading_code
   struct order_var
   struct party
   CHAR[32] exchange_info_s  // Exchange, Information
}
```

5.49 COMBO_ACC_TRANS_P (34116)

```
struct combo_acc_trans_p {
   struct transaction_type
   struct series  // Named struct no: 50000
   struct trading_code
   struct order_var
   CHAR[32] exchange_info_s  // Exchange, Information
   QUAD_WORD order_number_u  // Order Number
   struct give_up_member  // Named struct no: 50002
}
```

5.50 STOP_ORDER_TRANS_P (34117)

```
struct stop_order_trans_p {
   struct transaction_type
   struct series  // Named struct no: 50000
   struct trading_code
   struct order_var
   struct stop_series
   INT32_T limit_premium_i  // Premium, Limit
   struct give_up_member  // Named struct no: 50002
   CHAR[32] exchange_info_s  // Exchange, Information
   INT64_T total_volume_i  // Total Volume
}
```

5.51 PRIO_CROSSING_TRANS_P (34118)

```
struct prio_crossing_trans_p {
   struct transaction_type
   struct series  // Named struct no: 50000
   struct trading_code
   struct prio_crossing
}
```

5.52 TRADE_REPORT_1_TRANS_P (34119)

```
struct trade_report_1_trans_p {
   struct transaction_type
   struct series  // Named struct no: 50000
   struct trading_code
   struct order_var
   struct party
   CHAR[32] exchange_info_s  // Exchange, Information
   struct give_up_member  // Named struct no: 50002
   char[8] settlement_date_s  // Date, Settlement
   char[8] time_of_agreement_date_s  // Time of agreement, date part
```

```
   char[6] time_of_agreement_time_s  // Time of agreement, time part
   UINT8_T deferred_publication_c  // Deferred Publication
   CHAR filler_1_s  // Filler
}
```

5.53 CPPX_INITIATION_TRANS_P (34123)

```
struct cppx_initiation_trans_p {
   struct transaction_type
   struct series  // Named struct no: 50000
   struct trading_code
   QUAD_WORD order_number_u  // Order Number
   struct prio_crossing
}
```

5.54 LONG_STOP_ORDER_TRANS_P (34124)

```
struct long_stop_order_trans_p {
   struct transaction_type
   struct series  // Named struct no: 50000
   struct trading_code
   struct order_var
   struct stop_series
   INT32_T limit_premium_i  // Premium, Limit
   struct give_up_member  // Named struct no: 50002
   CHAR[32] exchange_info_s  // Exchange, Information
   INT64_T total_volume_i  // Total Volume
}
```

5.55 CPPX_CONFIRMATION_TRANS_P (34125)

```
struct cppx_confirmation_trans_p {
   struct transaction_type
   struct series  // Named struct no: 50000
   struct trading_code
   QUAD_WORD order_number_u  // Order Number
   struct prio_crossing
}
```

5.56 DEAL_USER (34251)

```
struct deal_user {
   struct broadcast_type
   struct series  // Named struct no: 50000
   struct timestamp_match  // Of type: TIME_SPEC
   UINT32_T sequence_number_u  // Sequence Number
   INT32_T deal_price_i  // Price, Deal
```

```
   INT64_T deal_quantity_i  // Quantity, Deal
   UINT16_T segment_number_n  // Segment Number
   UINT8_T hidden_price_c  // Hidden Price
   UINT8_T ext_t_state_c  // Trade Report Type
   UINT8_T items_c  // Item
   CHAR filler_1_s  // Filler
   UINT16_T trade_condition_n  // Trade Condition
   Array ITEM [max no: 42] {
      QUAD_WORD order_number_u  // Order Number
      INT64_T deal_quantity_i  // Quantity, Deal
      INT64_T rem_quantity_i  // Quantity, Remaining
      UINT32_T block_n  // Block Size
      UINT8_T bid_or_ask_c  // Bid or Ask
      UINT8_T deal_source_c  // Deal Source
      UINT16_T exch_order_type_n  // Order Type, Exchange
   }
}
```

5.57 BASIC_TRADE_TICKER (34401)

```
struct basic_trade_ticker {
   struct series  // Named struct no: 50000
   struct timestamp_match  // Of type: TIME_SPEC
   struct time_of_publication  // Of type: TIME_SPEC
   UINT64_T execution_event_nbr_u  // Execution number
   UINT32_T match_group_nbr_u  // Match group number, group inside an execution
   INT64_T deal_quantity_i  // Quantity, Deal
   INT32_T deal_price_i  // Price, Deal
   UINT16_T segment_number_n  // Segment Number
   UINT8_T aggressive  // Bid or Ask ; Of type: BID_OR_ASK_C
   CHAR filler_1_s  // Filler
}
```

5.58 EXTENDED_TRADE_TICKER (34402)

```
struct extended_trade_ticker {
   UINT16_T trade_condition_n  // Trade Condition
   UINT16_T deal_info_n  // Deal Information
}
```

5.59 TRADE_REPORT_TRADE_TICKER (34403)

```
struct trade_report_trade_ticker {
   UINT8_T trade_report_type  // Trade Report Type ; Of type: EXT_T_STATE_C
   char[8] settlement_date_s  // Date, Settlement
   char[8] time_of_agreement_date_s  // Time of agreement, date part
   char[6] time_of_agreement_time_s  // Time of agreement, time part
   UINT8_T outside_info_spread_c  // Outside Information Spread
}
```

5.60 HALF_TRADE_TICKER (34405)

```
struct half_trade_ticker {
   struct trading_code
   INT64_T trade_quantity_i  // Quantity, Trade
   UINT32_T block_n  // Block Size
   UINT8_T bid_or_ask_c  // Bid or Ask
   UINT8_T deal_source_c  // Deal Source
   char[2] filler_2_s  // Filler
}
```

5.61 TRADE_TICKER_AMEND (34406)

```
struct trade_ticker_amend {
   UINT64_T execution_event_nbr_u  // Execution number
   UINT32_T match_group_nbr_u  // Match group number, group inside an execution
   UINT8_T trade_state_c  // Trade, State
   CHAR include_in_statistics_c  // Include in statistics
   char[2] filler_2_s  // Filler
}
```

5.62 FREE_TEXT (34801)

```
struct free_text {
   char[15] customer_info_s  // Customer, Information
   CHAR filler_1_s  // Filler
}
```

5.63 CLEARING_INFO (34802)

```
struct clearing_info {
   struct give_up_member  // Named struct no: 50002
   char[10] ex_client_s  // Client
   UINT8_T open_close_req_c  // Open  Close Request
   CHAR filler_1_s  // Filler
}
```

5.64 LINKED_ORDER_LEG (34803)

```
struct linked_order_leg {
   struct series  // Named struct no: 50000
   INT32_T premium_i  // Premium
   INT64_T quantity_i  // Quantity
   UINT32_T block_n  // Block Size
   UINT8_T order_type_c  // Order Type
```

```
   UINT8_T bid_or_ask_c  // Bid or Ask
   char[2] filler_2_s  // Filler
}
```

5.65 ORDER_OWNER (34804)

```
struct order_owner {
   struct owner  // Of type: TRADING_CODE
}
```

5.66 TIME_IN_FORCE (34807)

```
struct time_in_force {
   UINT16_T time_validity_n  // Validity Time
   char[2] filler_2_s  // Filler
}
```

5.67 TRADE_REPORT_BASE (34808)

```
struct trade_report_base {
   struct series  // Named struct no: 50000
   struct party
   QUAD_WORD order_number_u  // Order Number
   INT32_T premium_i  // Premium
   INT64_T quantity_i  // Quantity
   UINT8_T bid_or_ask_c  // Bid or Ask
   UINT8_T trade_report_type  // Trade Report Type ; Of type: EXT_T_STATE_C
   char[8] time_of_agreement_date_s  // Time of agreement, date part
   char[6] time_of_agreement_time_s  // Time of agreement, time part
   char[8] settlement_date_s  // Date, Settlement
   UINT8_T deferred_publication_c  // Deferred Publication
   UINT8_T ob_command_c  // Order-Book Command
   char[2] filler_2_s  // Filler
}
```

5.68 LINKED_ORDER_LEG_NUMBER (34809)

```
struct linked_order_leg_number {
   UINT8_T leg_number  // Item Number ; Of type: ITEM_NUMBER_C
   char[3] filler_3_s  // Filler
}
```

5.69 MULTI_LEG_ORDER_INSERT (34817)

```
struct multi_leg_order_insert {
```

```
   struct transaction_type
   struct series  // Named struct no: 50000
   INT32_T premium_i  // Premium
   struct give_up_member  // Named struct no: 50002
   CHAR[32] exchange_info_s  // Exchange, Information
   char[15] customer_info_s  // Customer, Information
   char[10] ex_client_s  // Client
   UINT8_T open_close_req_c  // Open  Close Request
   UINT8_T multi_leg_price_type_c  // Multi Leg Price Type
   UINT8_T order_type_c  // Order Type
   UINT8_T items_c  // Item
   char[3] filler_3_s  // Filler
   Array ITEM [max no: 5] {
      struct series  // Named struct no: 50000
      INT64_T quantity_i  // Quantity
      INT32_T premium_i  // Premium
      UINT8_T bid_or_ask_c  // Bid or Ask
      UINT8_T calculate_quantity_method_c  // Calculate Quantity Method
      char[2] filler_2_s  // Filler
   }
}
```

5.70 MULTI_LEG_ORDER_LEG_NUMBER (34818)

```
struct multi_leg_order_leg_number {
   UINT8_T leg_number  // Item Number ; Of type: ITEM_NUMBER_C
   char[3] filler_3_s  // Filler
}
```

5.71 MULTI_LEG_ORDER_INSERT_P (34819)

```
struct multi_leg_order_insert_p {
   struct transaction_type
   struct series  // Named struct no: 50000
   struct trading_code
   INT32_T premium_i  // Premium
   struct give_up_member  // Named struct no: 50002
   CHAR[32] exchange_info_s  // Exchange, Information
   char[15] customer_info_s  // Customer, Information
   char[10] ex_client_s  // Client
   UINT8_T open_close_req_c  // Open  Close Request
   UINT8_T multi_leg_price_type_c  // Multi Leg Price Type
   UINT8_T order_type_c  // Order Type
   UINT8_T items_c  // Item
   char[3] filler_3_s  // Filler
   Array ITEM [max no: 5] {
      struct series  // Named struct no: 50000
      INT64_T quantity_i  // Quantity
      INT32_T premium_i  // Premium
      UINT8_T bid_or_ask_c  // Bid or Ask
      UINT8_T calculate_quantity_method_c  // Calculate Quantity Method
      char[2] filler_2_s  // Filler
```

```
    }
}
```

5.72 SEGMENT_INSTANCE_NUMBER (34901)

```
struct segment_instance_number {
    UINT16_T segment_number_n  // Segment Number
    UINT8_T instance_c  // Instance, Number
    CHAR filler_1_s  // Filler
    UINT32_T sequence_number_u  // Sequence Number
    struct trading_code
}
```

5.73 ORDER_CHANGE_COMBINED (34902)

```
struct order_change_combined {
    INT64_T mp_quantity_i  // Quantity
    INT64_T total_volume_i  // Total Volume
    UINT8_T item_number_c  // Item Number
    UINT8_T bid_or_ask_c  // Bid or Ask
    UINT8_T change_reason_c  // Change Reason
    CHAR filler_1_s  // Filler
}
```

5.74 ORDER_CHANGE_SEPARATE (34903)

```
struct order_change_separate {
    struct series  // Named struct no: 50000
    QUAD_WORD order_number_u  // Order Number
    INT64_T mp_quantity_i  // Quantity
    INT64_T total_volume_i  // Total Volume
    UINT8_T bid_or_ask_c  // Bid or Ask
    UINT8_T change_reason_c  // Change Reason
    char[10] ex_client_s  // Client
    char[15] customer_info_s  // Customer, Information
    CHAR filler_1_s  // Filler
    struct originator_trading_code
    struct execution_timestamp  // Of type: TIME_SPEC
}
```

5.75 ORDER_RETURN_INFO (34904)

```
struct order_return_info {
    INT32_T trans_ack_i  // Transaction, Acknowledgement
    QUAD_WORD order_number_u  // Order Number
    struct originator_trading_code
    struct execution_timestamp  // Of type: TIME_SPEC
```

```
}
```

5.76 ORDER_PRICE_CHANGE (34905)

```
struct order_price_change {
   struct series  // Named struct no: 50000
   QUAD_WORD order_number_u  // Order Number
   INT32_T premium_i  // Premium
   struct execution_timestamp  // Of type: TIME_SPEC
   UINT8_T bid_or_ask_c  // Bid or Ask
   UINT8_T change_reason_c  // Change Reason
   char[2] filler_2_s  // Filler
}
```

5.77 MULTI_ORDER_RESPONSE (34906)

```
struct multi_order_response {
   INT32_T transaction_status_i  // Transaction, Status
   INT32_T trans_ack_i  // Transaction, Acknowledgement
   UINT8_T item_number_c  // Item Number
   char[3] filler_3_s  // Filler
}
```

5.78 COMBO_TRANS_PART (34907)

```
struct combo_trans_part {
   struct transaction_type
   struct series  // Named struct no: 50000
   struct order_var
   CHAR[32] exchange_info_s  // Exchange, Information
   struct give_up_member  // Named struct no: 50002
}
```

5.79 COMBO_TRANS_PART_P (34908)

```
struct combo_trans_part_p {
   struct transaction_type
   struct series  // Named struct no: 50000
   struct trading_code
   struct order_var
   CHAR[32] exchange_info_s  // Exchange, Information
   struct give_up_member  // Named struct no: 50002
}
```

5.80 BB_CHANGE_SEPARATE (34909)

```
struct bb_change_separate {
   QUAD_WORD order_number_u  // Order Number
   INT64_T mp_quantity_i  // Quantity
   UINT8_T change_reason_c  // Change Reason
   char[3] filler_3_s  // Filler
}
```

5.81 ORDER_STATUS (34910)

```
struct order_status {
   CHAR[32] exchange_info_s  // Exchange, Information
   char[15] customer_info_s  // Customer, Information
   UINT8_T open_close_req_c  // Open  Close Request
   INT32_T premium_i  // Premium
   struct party
   INT64_T orig_shown_quantity_i  // Shown Quantity, Original
   INT64_T orig_total_volume_i  // Total Volume, Original
   INT64_T rem_quantity_i  // Quantity, Remaining
   UINT16_T transaction_number_n  // Transaction Type Number
   UINT16_T exch_order_type_n  // Order Type, Exchange
   char[10] ex_client_s  // Client
   UINT8_T order_type_c  // Order Type
   struct give_up_member  // Named struct no: 50002
   CHAR filler_1_s  // Filler
}
```

5.82 ORDER_STATE (34913)

```
struct order_state {
   UINT32_T order_state_u  // Order State
}
```

5.83 ORDER_INFO (34917)

```
struct order_info {
   struct timestamp_in  // Of type: TIME_SPEC
   struct timestamp_created  // Of type: TIME_SPEC
   QUAD_WORD order_number_u  // Order Number
   struct party
   struct order
   INT64_T total_volume_i  // Total Volume
   INT64_T display_quantity_i  // Quantity, Display
   INT64_T orig_total_volume_i  // Total Volume, Original
   INT64_T orig_shown_quantity_i  // Shown Quantity, Original
   UINT32_T order_state_u  // Order State
```

```
}
```

5.84 MP_TRADE_PRICE (34918)

```
struct mp_trade_price {
   struct series  // Named struct no: 50000
   QUAD_WORD order_number_u  // Order Number
   UINT8_T bid_or_ask_c  // Bid or Ask
   UINT8_T deal_source_c  // Deal Source
   UINT16_T trade_condition_n  // Trade Condition
   INT32_T deal_price_i  // Price, Deal
   INT64_T deal_quantity_i  // Quantity, Deal
   UINT8_T ext_t_state_c  // Trade Report Type
   UINT8_T opposing_deal_source_c  // Opposing Deal Source
   UINT16_T exch_order_type_n  // Order Type, Exchange
   QUAD_WORD opposing_order_number_u  // Order Number, Opposing
}
```

5.85 ORDER_CHG_SEP_TRANS_ACK (34919)

```
struct order_chg_sep_trans_ack {
   INT32_T trans_ack_i  // Transaction, Acknowledgement
   struct order_change_separate  // Named struct no: 34903
}
```

5.86 ORDER_TRADE_INFO (34920)

```
struct order_trade_info {
   struct match_id
   INT32_T trade_price_i  // Price, Trade
   INT64_T trade_quantity_i  // Quantity, Trade
   UINT8_T item_number_c  // Item Number
   UINT8_T deal_source_c  // Deal Source
   UINT8_T bid_or_ask_c  // Bid or Ask
   CHAR filler_1_s  // Filler
}
```

5.87 ORDER_LEG_TRADE_INFO (34921)

```
struct order_leg_trade_info {
   struct series  // Named struct no: 50000
   struct match_id
   QUAD_WORD order_number_u  // Order Number
   INT32_T trade_price_i  // Price, Trade
   INT64_T trade_quantity_i  // Quantity, Trade
   UINT8_T item_number_c  // Item Number
   UINT8_T deal_source_c  // Deal Source
```

```
   UINT8_T bid_or_ask_c  // Bid or Ask
   CHAR filler_1_s  // Filler
}
```

5.88 MESSAGE_CORE_INFO (35001)

```
struct message_core_info {
   UINT32_T sequence_number_u  // Sequence Number
   UINT8_T message_information_type_c  // Message Information, Type
   char[80] message_source_s  // Message, Source
   char[8] yyyymmdd_s  // Date
   char[6] hhmmss_s  // Time, External
   UINT8_T message_priority_c  // Message, Priority
   char[80] message_header_s  // Message, Header
   UINT8_T update_status_note_c  // Status Note, Update
   char[3] filler_3_s  // Filler
}
```

5.89 MESSAGE_INFORMATION (35002)

```
struct message_information {
   UINT16_T items_n  // Items
   char[2] filler_2_s  // Filler
   Array ITEM [max no: 10] {
      char[80] text_line_s  // Text, Line
   }
}
```

5.90 DESTINATION_ITEM (35003)

```
struct destination_item {
   struct series  // Named struct no: 50000
   UINT8_T destination_level_c  // Destination, Level
   char[3] filler_3_s  // Filler
}
```

5.91 DOCUMENT_URL (35004)

```
struct document_url {
   UINT8_T items_c  // Item
   CHAR[255] url_link_s  // Link, URL
}
```

5.92 NS_DELTA_HEADER (37001)

```
struct ns_delta_header {
   INT64_T download_ref_number_q  // Download Reference Number
   struct full_answer_timestamp  // Of type: TIME_SPEC
   UINT8_T full_answer_c  // Full Answer
   char[3] filler_3_s  // Filler
}
```

5.93 NS_REMOVE (37002)

```
struct ns_remove {
   struct series  // Named struct no: 50000
}
```

5.94 NS_INST_CLASS_BASIC (37101)

```
struct ns_inst_class_basic {
   struct series  // Named struct no: 50000
   struct upper_level_series
   INT32_T price_quot_factor_i  // Price, Quotation Factor
   INT32_T contract_size_i  // Contract Size
   INT32_T redemption_value_i  // Redemption Value
   INT32_T undisclosed_min_ord_val_i  // Minimum Order Value, Undisclosed
Quantity
   INT32_T opt_min_ord_val_i  // Optional minimum order value
   INT32_T opt_min_trade_val_i  // Optional minimum trade value
   UINT16_T derivate_level_n  // Derivate Level
   UINT16_T dec_in_strike_price_n  // Decimals, Strike Price
   UINT16_T dec_in_contr_size_n  // Decimals, Contract Size
   UINT16_T rnt_id_n  // Ranking Type
   UINT16_T virt_commodity_n  // Virtual Underlying
   UINT16_T settlement_days_n  // Settlement, Days or Month
   UINT8_T settl_day_unit_c  // Settlement Day Unit
   char[14] inc_id_s  // Instrument Class, Identity
   char[32] name_s  // Name
   char[10] trc_id_s  // Trade Report Class
   char[3] base_cur_s  // Currency, Trading
   UINT8_T traded_c  // Traded
   UINT8_T price_unit_premium_c  // Price Unit, Premium
   UINT8_T price_unit_strike_c  // Price Unit, Strike
   UINT8_T indicative_prices_c  // Indicative Prices
   UINT8_T trd_cur_unit_c  // Traded Currency Unit
   UINT8_T db_operation_c  // Operation
   char[12] csd_id_s  // CSD, Identity
   char[2] filler_2_s  // Filler
}
```

5.95 NS_PRICE_TICK (37102)

```
struct ns_price_tick {
   struct tick_size
   UINT16_T dec_in_premium_n  // Decimals, Premium
   CHAR is_fractions_c  // Fraction, Premium
   UINT8_T price_format_c  // Premium/Price Format
}
```

5.96 NS_BLOCK_SIZE (37103)

```
struct ns_block_size {
   INT64_T maximum_size_u  // Block Size, Maximum Volume
   UINT32_T minimum_size_n  // Block Size, Minimum Volume
   UINT32_T block_n  // Block Size
   UINT8_T lot_type_c  // Lot, Type
   char[3] filler_3_s  // Filler
}
```

5.97 NS_INST_CLASS_SECUR (37105)

```
struct ns_inst_class_secur {
   INT32_T exerc_limit_i  // Exercise, Limit
   UINT16_T dec_in_deliv_n  // Decimals, Delivery
   UINT16_T cleared_dec_in_qty_n  // Decimals, Quantity
   UINT16_T dec_in_fixing_n  // Decimals, Fixing
   UINT8_T exerc_limit_unit_c  // Exercise, Limit Unit
   char[32] settl_cur_id_s  // Currency, Settlement
   char[12] csd_id_s  // CSD, Identity
   UINT8_T fixing_req_c  // FIXING_REQ_C
}
```

5.98 NS_UNDERLYING_BASIC (37201)

```
struct ns_underlying_basic {
   UINT16_T commodity_n  // Commodity Code
   UINT16_T linked_commodity_n  // Linked Commodity Code
   UINT16_T state_number_n  // Trading State Number
   UINT16_T dec_in_price_n  // Decimals, Price
   char[6] com_id_s  // Underlying Identity
   char[12] isin_code_s  // ISIN Code
   char[32] name_s  // Name
   char[3] base_cur_s  // Currency, Trading
   UINT8_T deliverable_c  // Deliverable
   UINT8_T underlying_type_c  // Type, Underlying
   UINT8_T price_unit_c  // Price Unit, Underlying
   UINT8_T underlying_status_c  // Underlying Status
```

```
   char[6] underlying_issuer_s  // Underlying Issuer
   char[4] sector_code_s  // Sector Code
   UINT8_T virtual_c  // Virtual
   char[2] country_id_s  // Name, Country
   CHAR ext_provider_c  // External Price Feed Provider
   char[40] external_id_s  // External Price Feed Identity
   UINT8_T cur_unit_c  // Currency Unit
   UINT8_T db_operation_c  // Operation
   char[3] filler_3_s  // Filler
}
```

5.99 NS_FIXED_INCOME (37202)

```
struct ns_fixed_income {
   INT64_T nominal_value_q  // Nominal Value
   UINT32_T coupon_interest_i  // Coupon Interest
   UINT16_T dec_in_nominal_n  // Decimals, Nominal
   UINT16_T coupon_settlement_days_n  // Coupon Settlement Days
   UINT16_T coupon_frequency_n  // Coupon Frequency
   UINT16_T rate_determ_days_n  // Rate Determination Days
   char[8] date_release_s  // Date, Issue
   char[8] date_termination_s  // Date, Maturity
   char[8] date_dated_s  // Date, Dated
   char[8] date_proceed_s  // Date, Proceed
   UINT8_T fixed_income_type_c  // Fixed Income Type
   UINT8_T day_calc_rule_c  // Day Calculation Rule
   char[2] filler_2_s  // Filler
}
```

5.100 NS_COUPON_DATES (37203)

```
struct ns_coupon_dates {
   char[8] date_coupdiv_s  // Coupon/Dividend Date
   char[8] date_booksclose_s  // Booksclose Date
   UINT32_T dividend_i  // Dividend
}
```

5.101 NS_INST_SERIES_BASIC (37301)

```
struct ns_inst_series_basic {
   struct series  // Named struct no: 50000
   UINT16_T step_size_multiple_n  // Tick Size, Multiple
   char[32] ins_id_s  // Series, Identity
   char[32] long_ins_id_s  // Series Name, Long
   char[8] date_last_trading_s  // Date, Last Trading
   char[6] time_last_trading_s  // Time, Last Trading
   char[8] date_first_trading_s  // Date, First Trading
   char[6] time_first_trading_s  // Time, First Trading
   UINT8_T series_status_c  // Series, Status
```

```
   UINT8_T suspended_c  // Suspended
   UINT8_T traded_in_click_c  // Traded in GENIUM
   UINT8_T db_operation_c  // Operation
   UINT8_T trade_reporting_only_c  // Only trade reports allowed
   CHAR filler_1_s  // Filler
}
```

5.102 NS_INST_SERIES_BASIC_SINGLE (37302)

```
struct ns_inst_series_basic_single {
   struct upper_level_series
   INT32_T contract_size_i  // Contract Size
   INT32_T price_quot_factor_i  // Price, Quotation Factor
   UINT16_T state_number_n  // Trading State Number
   UINT16_T ex_coupon_n  // Period, Ex Coupon
   char[12] isin_code_s  // ISIN Code
   char[8] settlement_date_s  // Date, Settlement
   char[8] first_settlement_date_s  // Date, First Settlement
   char[8] date_notation_s  // Date, Notation
   UINT8_T deliverable_c  // Deliverable
   char[8] effective_exp_date_s  // Effective Expiration Date
   UINT8_T ext_info_source_c  // External Information Source
   char[2] filler_2_s  // Filler
}
```

5.103 NS_INST_SERIES_BO (37306)

```
struct ns_inst_series_bo {
   char[12] isin_code_old_s  // ISIN Code, Old Series
   UINT8_T tm_template_c  // Template Series
   UINT8_T tm_series_c  // Tailor Made Series
   UINT8_T accept_collateral_c  // Accepted as Collateral
}
```

5.104 NS_COMBO_SERIES_LEG (37308)

```
struct ns_combo_series_leg {
   struct series  // Named struct no: 50000
   UINT16_T ratio_n  // Ratio
   CHAR op_if_buy_c  // Operation if Buy
   CHAR op_if_sell_c  // Operation if Sell
}
```

5.105 NS_INST_SERIES_ID (37310)

```
struct ns_inst_series_id {
   INT32_T orderbook_id_i  // Order book id
```

```
}
```

5.106 SERIES (50000)

```
struct series {
   UINT8_T country_c  // Country Number
   UINT8_T market_c  // Market Code
   UINT8_T instrument_group_c  // Instrument Group
   UINT8_T modifier_c  // Modifier
   UINT16_T commodity_n  // Commodity Code
   UINT16_T expiration_date_n  // Date, Expiration
   INT32_T strike_price_i  // Strike Price
}
```

5.107 GIVE_UP_MEMBER (50002)

```
struct give_up_member {
   char[2] country_id_s  // Name, Country
   char[5] ex_customer_s  // Customer, Identity
   CHAR filler_1_s  // Filler
}
```

5.108 EXCHANGE_INFO (50004)

```
struct exchange_info {
   CHAR[32] exchange_info_s  // Exchange, Information
}
```

6 Broadcast Overview

The table below lists all broadcasts provided in this message reference. This is also where each broadcast's Information Type Value is provided.

Table 1: Broadcast properties

Transaction Type	Name	Design	Information Type	Information Type Value
BD1	Deals in the Market	Standard	instrument class	7
BD2	Edited Price Information	Variable	instrument class	7
BD3	Underlying Information	Standard	general	1
BD6	Dedicated Trade Information	Variable	dedicated	4
BD18	Dedicated Delivery	Standard	dedicated	4
BD24	Cover Request Information	Standard	dedicated	4
BD26	Cover Request Update	Standard	dedicated	4
BD29	Directed Give Up	Standard	dedicated	4
BD39	Dedicated Trade Change Information	Standard	dedicated	4
BD40	Dedicated auxiliary position info update information	Standard	dedicated	4
BD70	Trade Ticker	Variable	instrument class	7
BD71	Amended Trades	Variable	instrument class	7
BI1	Resumption and Suspension of Trading	Standard	general	1
BI5	Indices Information	Standard	general	1
BI7	Signal Information Ready	Standard	general	1
BI9	Price Information Heartbeat	Standard	general	1
BI27	Clearing message	Standard	general	1
BI41	Instrument Status Information	Standard	general	1
BI63	Preliminary Settlement Prices	Standard	general	1
BI73	Undo Signal Ready Info	Standard	general	1
BI81	Market Announcement Information	Variable	general	1

Transaction Type	Name	Design	Information Type	Information Type Value
BO5	Firm Order Book	Variable	instrument dedicated	8
BO10	Equilibrium Price Update	Standard	instrument class	7
BO14	Order Book Levels	Variable	instrument class	7
BO15	Order Book Levels	Variable	instrument class	7
BO38	Market Maker Protection Settings Information	Standard	dedicated	4
BO55	Trade Report Notification	Variable	dedicated	4
BO99	Block Transaction Response	Standard	dedicated	4
BU2	Series Update	Standard	general	1
BU4	Underlying Update	Standard	general	1
BU5	Combination Update	Standard	general	1
BU9	Series Backoffice Update	Standard	general	1
BU10	Instrument Class Update	Standard	general	1
BU12	Account Type Update	Standard	general	1
BU13	Account Fee Type Update	Standard	general	1
BU18	Non-Trading Days Update	Standard	general	1
BU19	Underlying Backoffice Update	Standard	general	1
BU20	Instrument Class Backoffice Update	Standard	general	1
BU28	Central Group Update	Standard	general	1
BU50	Non-Settlement Days Update	Standard	general	1
BU87	Market Maker Protection Update	Standard	dedicated	4
BU120	Delta Underlying Update	Variable	general	1
BU121	Delta Underlying Update for Back Office	Variable	general	1
BU122	Delta Instrument Class Update	Variable	general	1
BU123	Delta Instrument Class Update for Back Office	Variable	general	1

Transaction Type	Name	Design	Information Type	Information Type Value
BU124	Delta Instrument Series Update	Variable	general	1
BU125	Delta Instrument Series Update for Back Office	Variable	general	1
BU126	Combo Series Update	Variable	general	1
BU136	Combo Series Update for Back Office	Variable	general	1
MI4	Quote Request with Volume Information	Standard	derivative	2

7 Detailed Field Information

All fields used in the messages included in this message reference are listed in alphabetical order here.

The field descriptions provided here cover the general standard usage and interpretation. Message specific behaviour of a field is provided in each respective message chapter.

abbr_name_s (Abbreviated Name)	
Datatype	char[8]
Description	Abbreviated name
accept_collateral_c (Accepted as Collateral)	
Datatype	UINT8_T
Description	Accepted as collateral?.
Value Set	**name** / **value** Yes / 1 No / 2 Default / 0
account_alias_s (Account alias)	
Datatype	char[32]
Description	Defines the account name alias for an account.
account_field_no_n (Account Field Number)	
Datatype	UINT16_T
Description	The actual account attribute number.
account_id_s (Account, Identity)	
Datatype	char[10]
Description	The account identification part of an ACCOUNT structure; the part after the member identification.
account_text_s (Account Text)	
Datatype	char[20]
Description	Free text, 20 characters
account_type_c (Account Type)	
Datatype	UINT8_T
Description	The account type for a trade.
Value Set	**name** / **value** Customer / 1 Firm / 2 Market Maker / 3

account_type_s (Account Type)	
Datatype	char[12]
Description	Tells what type of account it is.
account_validation_c (Account Validation)	
Datatype	UINT8_T
Description	Account Validation
acc_allow_nov_c (Novation Allowed)	
Datatype	UINT8_T
Description	Defines if novation is allowed on an account or not.None indicates that novation is not applicable on the account.
Value Set	**name** / **value** None / 0 Yes / 1 No / 2
acc_as_pay_c (Accepted As Payment)	
Datatype	UINT8_T
Description	Accepted as payment
Value Set	**value** / **description** 1 / Yes 2 / No
acc_risk_type_c (Account Risk Type)	
Datatype	UINT8_T
Description	Defines account properties for margin requirements.
Value Set	**name** / **value** Plain Account / 1 Market Maker or Own Inventory Account / 2 Direct Pledging Account / 3
acc_state_c (Account State)	
Datatype	UINT8_T
Description	Defines the state that the account is in.
Value Set	**value** / **description** 0 / None 1 / Registered Account has been registered but not validated.

value	description
2	Inactive Account has been active and then inactivated.
3	Active Account is validated and open for position or trade.
4	Deleted Account is deleted.

acc_type_s (Account Type)	
Datatype	char[12]
Description	Tells what type of account it is

action_odd_lot_c (Odd Lot, Action)	
Datatype	UINT8_T
Description	Action to take for existing odd lot orders when entering the state.

Value Set

value	description
1	No Action
2	Delete

activate_at_reg_c (Activate At Registration)	
Datatype	UINT8_T
Description	Activate the account at the same time as registration:

Value Set

value	description
1	Yes
2	No

actual_start_date_s (Actual Start Date)	
Datatype	char[8]
Description	Defines actual start date. Distributed in UTC together with Actual Start Time. Format: YYYYM-MDD.

actual_start_time_s (Actual Start Time)	
Datatype	char[6]
Description	Defines actual start time. Distributed in UTC together with Actual Start Date. Format: HHMMSS.

added_trade_sim_c (Added Trades Simulated)	
Datatype	UINT8_T
Description	Defines how trades added in a simulation should be handled.

Value Set

value	description
0	No special action.

adjusted_c (Adjusted Series)	
Datatype	UINT8_T
Description	Is the actual adjustment containing new adjusted series?
Value Set	**value** / **description** 1 / Yes 2 / No
adjust_ident_n (Adjustment Identifier)	
Datatype	UINT16_T
Description	A number that uniquely identifies an adjustment for series with the same adjustment conditions.
aggressive_c (Aggressive)	
Datatype	UINT8_T
Description	Specifies whether the order from which a trade originates was the passive or aggressive part when the deal was matched, i.e. whether the order was stored in the order book before being eligible for a match with an order arriving later on.
Value Set	**name** / **value** Passive part / 0 Aggressive part / 1 Aggressive/passive part unknown or not applicable / 2
allow_interbank_c (Allow interbank)	
Datatype	UINT8_T
Description	The trade report type is allowed to report between different participant.
Value Set	**name** / **value** Yes / 1 No / 2
allow_non_std_settlement_c (Allow non standard settlement)	
Datatype	UINT8_T
Description	Allow a non standard settlement date in the trade report.
Value Set	**name** / **value** Yes / 1 No / 2
allow_within_participant_c (Allow within participant)	
Datatype	UINT8_T
Description	The trade report type is allowed to report within the same participant.

Value Set	<table><tr><th>name</th><th>value</th></tr><tr><td>Yes</td><td>1</td></tr><tr><td>No</td><td>2</td></tr></table>
allwd_price_move_i (Price Movement Max)	
Datatype	INT32_T
Description	Allowed Price Movements.
all_or_none_c (All Or None)	
Datatype	UINT8_T
Description	Specifies whether the information relates to the All or None Orderbook.
Value Set	<table><tr><th>value</th><th>description</th></tr><tr><td>1</td><td>Yes</td></tr><tr><td>2</td><td>No</td></tr></table>
application_status_i (Status, Application)	
Datatype	INT32_T
Description	The status indicates that a trading application has logged on and that all initializations needed are ready. The value is always equal to one.
ascii_bin_c (ASCII or Binary)	
Datatype	UINT8_T
Description	ASCII or Binary?
Value Set	<table><tr><th>value</th><th>description</th></tr><tr><td>1</td><td>ASCII</td></tr><tr><td>2</td><td>Binary</td></tr></table>
ask_marg_vol_i (Margin, Volatility Ask)	
Datatype	INT32_T
Description	Defines the latest volatility used for the series. For other instruments than options, the value is always zero. For series without positions, the volatility is calculated in the same way as if the series had positions. If it is impossible to calculate volatilities due to missing prices, the risk parameter imposed by the clearinghouse is returned. Expressed in percent, 4 implicit decimals.
ask_mask_n (Mask, Ask)	
Datatype	UINT16_T
Description	Bit mask.
ask_premium_i (Ask Premium)	
Datatype	INT32_T
Description	The price of one Series (excluding transaction cost) a user is prepared to pay - or wants to receive. This is always an integer.

	In the distribution of data from the exchange these fields may hold a value where bit 31 (highest bit) is set while all other bits are cleared. This indicates that there is no premium available. This differs from the value of zero (all bits cleared) indicating a premium prize of zero.
Value Set	**value** / **description** >0 / Price = 0 / Market price <0 / Combo price (may be neg).
ask_price_i (Ask Price)	
Datatype	UINT32_T
Description	Price for ask requests (orders selling the given Series). Statistics information.
ask_quantity_i (Quantity, Ask)	
Datatype	INT64_T
Description	Number of units (options, futures, forwards and so on) in an double price order related transaction.
ask_theo_c (Ask, Theoretical Mark)	
Datatype	UINT8_T
Description	The field indicates the origin of the price:
Value Set	**value** / **description** 0 / Missing 1 / Theoretically calculated 2 / From the Orderbook 3 / Manually updated 4 / Artificial
ask_total_volume_i (Total Volume, Ask)	
Datatype	INT64_T
Description	Total number of units (options, futures, forwards and so on) for ask side in an order related transaction.
asof_date_s (Date, As Of)	
Datatype	char[8]
Description	The date an object is valid for. Format: YYYYMMDD.
asof_time_s (Time, As Of)	
Datatype	char[6]
Description	The time an object is valid for. Format: HHMMSS.
atr_id_s (Account Type Rule)	
Datatype	char[12]
Description	The identity of Account Type Rule.
attention_c (Attention)	

Datatype	UINT8_T
Description	This field gives information about the trade. The field is retained for compatibility with earlier versions of the API. It contains the same information as in the first 8 bits of BIG ATTENTION. Please note that all bits but Bit1 and Bit2 are masked in full clearing installations. This does not apply to deal capture solutions.
attribute_rule_c (Attribute Rule)	
Datatype	UINT8_T
Description	The attribute rule associated with the account attribute:
Value Set	**value** / **description** 1 / Mandatory 2 / Inherit 3 / Not Specified 4 / Within Participant 5 / Within Organization 6 / Optional 7 / Not Applicable
authorized_c (Authorized)	
Datatype	UINT8_T
Description	Defines if the user sending the query is authorized to use the Trade Report Type.
Value Set	**value** / **description** 1 / Yes The trade report type is allowed for the user. 2 / No The trade report type is not allowed for the user.
auto_net_c (Auto Netting)	
Datatype	UINT8_T
Description	If position on this account will be netted automatically in after business operation.
Value Set	**value** / **description** 0 / Not netted 1 / Netted
average_c (Average)	
Datatype	UINT8_T
Description	Not applicable.

Value Set	**value** / **description** 1 / Yes 2 / No
average_period_c (Average Period)	
Datatype	UINT8_T
Description	Not applicable.
Value Set	**value** / **description** 0 / Not applicable 1 / Quarterly 2 / Half Year 3 / Year
balance_quantity_i (Balance Quantity)	
Datatype	INT64_T
Description	0, no balance check is performed. More than 0, the remaining quantity must be the same as the balance quantity otherwise the transaction will be rejected. Less than 0, the transaction is rejected, a negative value is not allowed.
base_cur_s (Currency, Trading)	
Datatype	char[3]
Description	Defines the trading currency for the instrument or the currency for the underlying. The representation of the currency follows the S.W.I.F.T. handbook and ISO 3166 standard, e.g. SEK, GBP, USD and ATS.
base_offset_days_c (Offset Days For Settlement Margin Base)	
Datatype	UINT8_T
Description	Specifies whether offset days should be based upon calendar days or settlement days.
Value Set	**name** / **value** Settlement Days / 0 Calendar Days / 1
base_srs_cutoff_time_i (Cutoff time base series)	
Datatype	INT32_T
Description	Cutoff time in minutes for base series prices when using spread difference spot month. 0 means no limitation.
best_ask_premium_i (Best Ask Price, Pre-opening)	
Datatype	INT32_T
Description	The best ask price that will be in the orderbook when the market goes into a trading state where order matching is enabled.

best_ask_quantity_i (Best Ask Volume, Pre-opening)	
Datatype	INT64_T
Description	The volume for the best ask price that will be in the order book when the market goes into a trading state where order matching is enabled.
best_ask_volume_u (Best Ask Volume)	
Datatype	INT64_T
Description	Total volume of orders in the market on best ask.
best_bid_premium_i (Best Bid Price, Preopening)	
Datatype	INT32_T
Description	The best bid price that will be in the order book when the market goes into a trading state where order matching is enabled.
best_bid_quantity_i (Best Bid Volume, Preopening)	
Datatype	INT64_T
Description	The volume for the best bid price that will be in the order book when the market goes into a trading state where order matching is enabled.
best_bid_volume_u (Best Bid Volume)	
Datatype	INT64_T
Description	Total volume of orders in the market on best bid.
bic_code_s (BIC Code)	
Datatype	char[15]
Description	The BIC consists of four parts and is usually written as BANKCCLLMAR. The parts are interpreted as explained in the table:
Value Set	**value** / **description** BANK — The first four characters is the Bank Code. It is unique to each financial institution and can only be made up of letters. [4 bytes] CC — CC is the ISO country code. The country code identifies the country in which the financial institution is located. [2 bytes] LL — LL is the Location Code. This 2-character code may be alphabetical or numerical. The location code provides geographical distinction within a country, e.g., cities, states, provinces and time zones. [2 bytes] MAR — MAR is the Branch Code. This 3-character code is called the Branch Code. It identifies a specific branch, or, for example, a department in a bank within the same country as the 8-character SWIFT BIC. This code may be alphabetical or numerical. The Branch code is optional for SWIFT users. [3 bytes]
bid_marg_vol_i (Margin, Volatility Bid)	
Datatype	INT32_T

Description	Defines the latest volatility used for the series. For other instruments than options, the value is always zero. For series without positions, the volatility is calculated in the same way as if the series had positions. If it is impossible to calculate volatilities due to missing prices, the risk parameter imposed by the clearinghouse is returned. Expressed in percent, 4 implicit decimals.
bid_mask_n (Mask, Bid)	
Datatype	UINT16_T
Description	Bit mask.
bid_or_ask_c (Bid or Ask)	
Datatype	UINT8_T
Description	Specifies what quotation side is requested.
Value Set	**value** / **description** 0 / Bid and Ask 1 / Bid 2 / Ask
bid_premium_i (Bid Premium)	
Datatype	INT32_T
Description	Premium for bid orders. The price of one Series (excluding transaction cost) a user is prepared to pay - or wants to receive. This is always an integer. In the distribution of data from the exchange these fields may hold a value where bit 31 (highest bit) is set while all other bits are cleared. This indicates that there is no premium available. This differs from the value of zero (all bits cleared) indicating a premium prize of zero.
Value Set	**value** / **description** >0 / Price = 0 / Market price <0 / Combo price (may be neg).
bid_price_i (Bid Price)	
Datatype	UINT32_T
Description	Price for bid requests (orders buying the given Series). Statistics information.
bid_quantity_i (Quantity, Bid)	
Datatype	INT64_T
Description	Number of units (options, futures, forwards and so on) in an double price order related transaction.
bid_theo_c (Bid, Theoretical Mark)	
Datatype	UINT8_T
Description	The field indicates the origin of the price:
Value Set	**value** / **description** 0 / Missing.

value	description
1	Theoretically calculated.
2	From the Orderbook.
3	Manually updated.
4	Artificial.

bid_total_volume_i (Total Volume, Bid)	
Datatype	INT64_T
Description	Total number of units (options, futures, forwards and so on) for bid side in an order related transaction.

big_attention_u (Big Attention)	
Datatype	UINT32_T
Description	The field big_attention gives information about the trade. This is a bit field that gives the following information, where the first bit is bit 0, and the value column represents each bit's numerical value. Note that not every value is applicable for every installation.

Value Set

name	value	description
resent	1	Resent (bit 0) The trade might have been subject to a retransition from the matching system to deal capture.
error_log	2	Error Log (bit 1) The trade has an entry in the error log, retrievable with CQ22 with error identity as trade number.
date_phase	4	Date Phase (bit 2) The trade date and the business date are not the same, menaing trades are created later than 24:00. Or in other words; as_of and created times contains a business_date that does not correspond to the site's date.
trd_prv_bus_dat	16	Previous Business Date (bit 4) The trade was made the previous business date for clearing next day.
aggressive	32	Aggressive Order (bit 5) The trade is created from an aggressive order that is, the trade (part of a deal) is the part created by an incoming order (as opposed to the part - one or more - that was al-

name	value	description
		ready stored in the order book).
clone_from_split	256	Split Clone (bit 8) The trade is a clone created in a split.
rev_old_trd	512	Reversing Previous (bit 9) The trade reverses a trade from previous date.
ovr_old_trd	512	Overtaking Previous (bit 9) The trade replaces a trade from previous date.
deal_rectified	1024	Rectification (bit 10) The trade is created or nullified in a deal rectification.
pure_position_txfr	16384	Position Transfer (bit 14) The trade represents a pure position transfer operation.
auto_netting_txn	32768	Position Transfer (bit 15) The trade results from an auto-netting operation.
rct_deal	131072	Overtaking (bit 17) The overtaking trade is created by a rectify deal operation.
deal_cancelled	262144	Deal Cancellation (bit 18) The trade is created by a cancel/annul deal operation.
force_flag	1048576	Force Order (bit 20) Force Order flag from Marketplace.
day2_correction	8388608	Day 2 correction (bit 23) Trade created during correction of an old deal.
rct_price_change	67108864	Rectify deal, price change (bit 26) Trade belongs to a deal subject to price correction.
rct_qty_change	134217728	Rectify deal, quantity change (bit 27) Trade belongs to a deal subject to correction of quantity.
rct_buy_sell_change	268435456	Rectify deal, buy/sell change (bit 28) Trade belongs to a deal subject to correction of buy and sell side.

name	value	description
excluded_from_stat	536870912	Excluded from trade statistics (bit 29) Trade belongs to a deal that has been excluded from trade statistics.
orig_NTD_deal	2147483648	(bit 31) Trade belongs to or derives from a deal that was executed for T+1 clearing.

binary_variant_c (Option, Binary Variant)	
Datatype	UINT8_T
Description	Defines the Option Binary Variants.

Value Set

value	description
0	Not applicable
1	Cash-or-nothing Pays out a predefined cash amount in case the option is in the money. Otherwise (out of the money), no money at all is paid out.
2	Asset-or-nothing Two different assets with corresponding dependencies on strike price determine whether a predefined amount of cash shall be paid out. There exists four different types of Asset-or-Nothing options: Call, Put, Down-up and Up-down.

bin_val_time_step_i (Time Steps)	
Datatype	INT32_T
Description	Number of time steps used when pricing options with the binomial method.
block_n (Block Size)	
Datatype	UINT32_T
Description	Minimum number of units (options, futures, forwards and so on) in an order transaction.
boolean (BOOLEAN)	
Datatype	CHAR
Description	Intermediate field.
bought_or_sold_c (Bought or Sold)	
Datatype	UINT8_T
Description	Defines if the item or amount in question is bought or sold.

Value Set

value	description
1	Bought

	value / **description** 2 / Sold
broadcast_number_n (Broadcast Number)	
Datatype	UINT16_T
Description	A number used to distinguish between different broadcasts.
broker_id_s (Broker, Identity)	
Datatype	char[5]
Description	The broker id is optional and may be used to identify brokers on a firm.
business_date_s (Date, Business)	
Datatype	char[8]
Description	Date in ASCII. Format: YYYYMMDD
buy_or_sell_c (Buy or Sell)	
Datatype	CHAR
Description	Buy or sell?
Value Set	**value** / **description** B / Buy S / Sell N / Not Applicable
buy_price_i (Buy Price)	
Datatype	INT32_T
Description	The buy price for a quote
buy_quantity_u (Buy Quantity)	
Datatype	INT64_T
Description	Number of units (options, futures, forwards and so on) in an double price order related transaction.
buy_sell_back_c (Buy Sell Back)	
Datatype	UINT8_T
Description	Sets if the REPO is a buy sell back or not.
Value Set	**value** / **description** 1 / Yes 2 / No
cabinet_format_c (Cabinet Format)	
Datatype	UINT8_T
Description	Not applicable.

cab_price_ind_c (Cabinet Price Indicator)	
Datatype	UINT8_T
Description	Specifies whether the price in a trade is a cabinet price or not.
Value Set	**value** / **description** 1 / Yes 2 / No
calculate_quantity_method_c (Calculate Quantity Method)	
Datatype	UINT8_T
Description	Method for calculating the quantity of a multi leg.
Value Set	**name** / **value** / **description** calc_quantity_method_none / 0 / Calculation Quantity Method None duration_neutral / 1 / Duration Neutral delta_neutral / 2 / Delta Neutral quantity_neutral / 3 / Quantity Neutral
calc_delta_protection_q (Calculated Delta Protection quantity)	
Datatype	INT64_T
Description	Calculated delta value for market maker protection
calc_quantity_protection_q (Calculated Quantity Protection)	
Datatype	INT64_T
Description	Calculated quantity value for market maker protection
cash_currency_s (Currency, Cash)	
Datatype	char[3]
Description	Currency for cash margin.
cash_margin_q (Cash Margin)	
Datatype	INT64_T
Description	Defines the cash margin.
cbo_trade_report_c (Combo Trade Report)	
Datatype	UINT8_T
Description	Describes if the Trade Report Type is used to do a combo trade report.
Value Set	**name** / **value** Yes / 1 No / 2
cbs_id_s (Combo Series, Identity)	

Datatype	char[32]
Description	ASCII representation of the standard combination series.
central_group_s (Central Group Name)	
Datatype	char[12]
Description	The ASCII representation of a centrally defined group.
central_module_c (Central Module)	
Datatype	CHAR
Description	Denotes essentially what subsystem is associated with the message. ISO Latin-1 representation is used. Central module:

Value Set

value	description
M	Market Place (MP/IMP)
C	Clearing (CL)
I	Information (IN)
S	Settlement (SE)
D	Common Database (CDB)
O	Operation (OP)
L	List Module (LM)
V	Settlement and Risk
R	Risk Valuation (RIVA)
U	Supervision (SU)
X	Deal Capture (DC)

change_previous_i (Change, Since Previous)	
Datatype	INT32_T
Description	Change in percent since previous corresponding information dissemination.
change_previous_s (Change, Since Previous)	
Datatype	char[8]
Description	Changes in percent between two values with sign if negative. Two decimals and a decimal point are included.
change_reason_c (Change Reason)	
Datatype	UINT8_T
Description	Defines why the order was changed.

Value Set

name	value	description
change_reason_delete	1	Order deleted
change_reason_deal	3	Deal
change_reason_inactive	4	Order inactivated
change_reason_change	5	Order altered

name	value	description
change_reason_add	6	Order added or activated
change_reason_mod_mkt	7	Market order converted Modified to EP during auction if an auction (market) order is modified during auction
change_reason_mod_price	8	Order price changed Order with undefined price converted to limit price, match price if a Market-to-limit order is stored
change_reason_system_delete	9	Order deleted by central system Deleted by system if the order is deleted by the central system
change_reason_proxy_delete	10	Order deleted by proxy Deleted by proxy if the order is deleted by proxy transaction
change_reason_recalculated	11	Bait order recalculated
change_reason_activated_stop	12	Stop order activated
change_reason_hv_recalc	13	Hidden volume order recalculated
change_reason_limit_change_del	15	Order deleted due to changed price limits Order deleted due to new price limits and the order premium is outside the new limits
change_reason_system_del_day	19	Order deleted by central system Order removed or changed by remove day or date orders flag
change_reason_iss_inactivate	21	Inactivated by system due to Instrument Session change.
change_reason_reload	30	Order reload at normal system start
change_reason_reload_intraday	31	Order reload at intraday Market Place restart
change_reason_no_longer_valid	33	Time validity specified in order has run out
change_reason_auction_delete	34	Market (Auction) order deleted during auction
change_reason_limit_change_inactivate	35	Order inactivated due to changed price limits

name	value	description
		Order inactivated due to new price limits and the order premium is outside the new limits
change_reason_limit_change_activate	36	Order activated due to changed price limits Order ctivated due to new price limits and the order premium is inside the new limits
change_reason_system_del_delta_protection	41	Order delete at market maker Delta Protection limit crossed.
change_reason_system_del_quantity_protection	42	Order delete at market maker Quantity Protection limit crossed.
change_reason_internal_crossing_delete	43	Order deleted because trader is not allowed to trade with himself
change_reason_t_plus_one_inactivate	52	Order Inactivated in T plus one
change_reason_t_plus_one_activate	53	Order is activated after it had been inactivated in T plus one

change_yesterday_i (Change, Since Yesterday)	
Datatype	INT32_T
Description	Change in percent since yesterday's values.

change_yesterday_s (Change, Since Yesterday)	
Datatype	char[8]
Description	Changes in % between two values with sign if negative. Two decimals and a decimal point are included.

chg_type_n (Change Type)	
Datatype	UINT16_T
Description	Information about the type of update performed on permanent information: Note: An Add might come for an already existing item in the front-end. A Change might come for a not yet existing item in the front-end. Some modifications that one might think of as a deletion are in fact changes, delistings for example.

Value Set

name	value	description
add	1	Addition The item is added.
delete	2	Deletion The item is deleted.
change	3	Modification

name	value	description
		The item is modified. Examples of modifications would be delistings and change of last trading time.

class_no_i (Class Number)	
Datatype	INT32_T
Description	Defines the type of settlement.

Value Set

name	value	description
	1	Marketplace fixed fee
	2	Clearing variable fee
	3	Tax
	4	Rebate
	5	Settlement Premium, MTM, etc.
Settlement_dvp	6	Delivery versus payment
New_contract	7	Create a new trade
Settlement_odvp	8	The other qty and base
	9	Internal information, API application should ignore this.
	10	Variation margin
Settlement_dvp_cvr	16	Quantity of underlying used as cover to be delivered
Settlement_odvp_cvr	18	Payment for delivery of cover quantity
	20	Rounding
	23	Fee 3
	24	Fee 4
	25	Fee 5
	26	Fee 6
	27	Fee 7
	28	Fee 8
	29	Fee 9
	30	Fair value

cleared_dec_in_qty_n (Decimals, Quantity)	
Datatype	UINT16_T

Description	Defines decimals in quantity in clearing related quantities.
clearing_date_s (Clearing Date)	
Datatype	char[8]
Description	Date in ASCII for clearing trade, format is YYYYMMDD.
clh_id_s (Clearinghouse)	
Datatype	char[12]
Description	Clearinghouse identity.
clh_or_cst_c (Clearing house or customer value)	
Datatype	UINT8_T
Description	Get values for clearing house or for customer
Value Set	**name** / **value** Clearing House / 0 Customer / 1
client_category_c (Client Category)	
Datatype	UINT8_T
Description	Type of client
Value Set	**name** / **value** NA / 1
closed_for_clearing_c (Closed, clearing)	
Datatype	UINT8_T
Description	Specifies if the date is closed for clearing.
Value Set	**name** / **value** Yes / 1 No / 2
closed_for_settlement_c (Closed, settlement)	
Datatype	UINT8_T
Description	Specifies if the date is closed for settlement.
Value Set	**name** / **value** Yes / 1 No / 2
closed_for_trading_c (Closed, trading)	
Datatype	UINT8_T
Description	Specifies if the date is closed for trading.

Value Set	**name** / **value** Yes / 1 No / 2
closing_date_s (Date, Closing)	
Datatype	char[8]
Description	Date in ASCII, format is YYYYMMDD. For deliveries, this field is the creation date of the delivery. For other instruments, this field is blank.
closing_price_i (Price, Closing)	
Datatype	INT32_T
Description	Defines the last traded price for the previous day.
cl_quantity_i (CL Quantity)	
Datatype	INT64_T
Description	Number of units (options, futures, forwards and so on) in an order related transaction.
cl_status_c (CL, Status)	
Datatype	CHAR
Description	Defines the clearing status for the participant.
Value Set	**value** / **description** S / Suspended from Clearing A / Active
collateral_type_c (Collateral types)	
Datatype	UINT8_T
Description	Defines the type of collateral.
Value Set	**name** / **value** Cash Collateral / 1 Guarantee / 2 Member Deposit / 3 Certificate / 4 Fixed Income / 5 Equity / 6
combo_deal_price_i (Combo deal price)	
Datatype	INT32_T
Description	Combo deal price.
combo_mark_c (Combination Order Mark)	

Datatype	UINT8_T
Description	If the order is an order with an implicit Premium, this is marked here. Note: For bulletin board orders this field is used as an index on each leg in the order.
Value Set	**value** / **description** 0 / Order with explicit Premium. Order has been entered via order or quote transaction to the system. 1 / Order with implicit Premium. Order has an implicit premium calculated by the marketplace, i.e baits generated by the system from standard combination series. This field will in this case always get the value from the corresponding instrument group defined in the CDB.
combo_source_c (Combination matching source)	
Datatype	UINT8_T
Description	This indicates if match was connected with any combination
Value Set	**name** / **value** / **description** combo_source_none / 0 / No combination involved combo_source_combo2combo / 3 / Combination matched combination combo_source_combo2single / 4 / Combination matched outright legs
combo_trade_seq_c (Combo Trades Sequence Number)	
Datatype	UINT8_T
Description	Holds a counter for combo trades
commission_i (Commission)	
Datatype	INT32_T
Description	The commission to pay for the operation.
commodity_n (Commodity Code)	
Datatype	UINT16_T
Description	Underlying definitions are defined by each exchange. Commodity Code is a part of the Series definition.
comp_delta_i (Composite Delta)	
Datatype	INT32_T
Description	Weighted average of delta in the different scenario points. Given with 2 decimals.
com_id (COM_ID)	
Datatype	char[6]
Description	Intermediate field.

com_id_s (Underlying Identity)	
Datatype	char[6]
Description	The ASCII representation of the underlying.
condition_s (Trade Report Description)	
Datatype	char[32]
Description	The description of the trade report type.
confirm_reject_c (Confirm or Reject)	
Datatype	UINT8_T
Description	The field states whether a holding item should be confirmed or rejected.
Value Set	**name** / **value**: Rejected / 0; Confirmed / 1
continues_matching_c (Matching, Open)	
Datatype	UINT8_T
Description	Automatic matching open in the state.
Value Set	**value** / **description**: 1 / Yes; 2 / No
contracts_modifier_c (Modifier, Number of Contracts)	
Datatype	UINT8_T
Description	The modifier is used to recalculate the item after an underlying adjustment. The field is stored with 3 implicit decimals.
Value Set	**value** / **description**: 1 / Modifier is added to the item; 2 / Modifier is subtracted from the item; 3 / Modifier is multiplied with the item; 4 / The item is divided by the modifier factor
contracts_mod_factor_i (Modifier Factor, Number of Contracts)	
Datatype	INT32_T
Description	The modifier is used to recalculate the item after an underlying adjustment. The field is stored with 3 implicit decimals.
contract_share_i (Contract Share)	
Datatype	INT32_T
Description	The number of contracts in the delivery, including decimals, as defined for the instrument class.
contract_size_i (Contract Size)	

Datatype	INT32_T
Description	Number of Underlying entities per contract.
contract_size_modifier_c (Modifier, Contract Size)	
Datatype	UINT8_T
Description	The modifier is used to recalculate the item after an underlying adjustment. The field is stored with 3 implicit decimals.
Value Set	**value** / **description** 1 / Modifier is added to the item 2 / Modifier is subtracted from the item 3 / Modifier is multiplied with the item 4 / The item is divided by the modifier factor
contr_size_mod_factor_i (Modifier Factor, Contract Size)	
Datatype	INT32_T
Description	The modifier is used to recalculate the item after an underlying adjustment. The field is stored with 5 implicit decimals.
corr_high_price_i (Price, Corresponding High)	
Datatype	INT32_T
Description	Defines the corresponding highest traded price during the day. If the instrument is traded in price this is the corresponding yield and if the instrument is traded in yield this is the corresponding price.
corr_last_price_i (Price, Corresponding Last)	
Datatype	INT32_T
Description	Defines the corresponding last traded price during the day. If instrument is traded in price this is the corresponding yield and if the instrument is traded in yield this is the corresponding price.
corr_low_price_i (Price, Corresponding Low)	
Datatype	INT32_T
Description	Defines the corresponding lowest traded price during the day. If the instrument is traded in price this is the corresponding yield and if the instrument is traded in yield this is the corresponding price.
corr_method_c (Margining Method)	
Datatype	CHAR
Description	Defines the margining method used.
Value Set	**name** / **value** Window method / W Hourly reduction method / H Delta hedging method / M Cardinal method / C Power Delta hedging method / P Norwegian Delta hedging method / N

corr_opening_price_i (Price, Corresponding First)	
Datatype	INT32_T
Description	Defines the corresponding first traded price for the day. If the instrument is traded in price this is the corresponding yield and if the instrument is traded in yield this is the corresponding price.
country_c (Country Number)	
Datatype	UINT8_T
Description	Country and exchange identity. Country Number is a part of the Series definition.
country_id_s (Name, Country)	
Datatype	char[2]
Description	The exchange code represented as ASCII, also known as COUNTRY. Since there may well be more than one exchange in one country, it's role is to specify the actual exchange at hand. It is the first component in the ACCOUNT and MEMBER structures.
country_s (Country)	
Datatype	char[2]
Description	The country ID where the exchange is located.
coupon_frequency_n (Coupon Frequency)	
Datatype	UINT16_T
Description	Number of coupons per year for bond underlying.
coupon_interest_i (Coupon Interest)	
Datatype	UINT32_T
Description	Coupon interest, decimal value stored with 6 decimals, e.g. 11% is stored as 110000.
coupon_settlement_days_n (Coupon Settlement Days)	
Datatype	UINT16_T
Description	Number of settlement days at coupon.
cover_margin_i (Cover Margin)	
Datatype	INT32_T
Description	Not applicable.
cover_quantity_i (Cover Quantity)	
Datatype	INT32_T
Description	Defines the quantity in the Cover Request.
cover_request_nbr_u (Cover Request Number)	
Datatype	UINT32_T
Description	Identifies a specific cover request.
created_date_s (Date, Created)	
Datatype	char[8]
Description	Date in ASCII. Format: YYYYMMDD. Defines the creation date of the item.
created_time_s (Time, Created)	
Datatype	char[6]

Description	Defines the creation time of the item. Format: HHMMSS.
create_over_api_c (Create Over API)	
Datatype	UINT8_T
Description	Allowed to create account over API?
Value Set	**value** / **description** 1 / Yes 2 / No
credit_class_s (Credit Class)	
Datatype	char[3]
Description	Exchange specific contents and interpretation.
csd_id_s (CSD, Identity)	
Datatype	char[12]
Description	Specifies the clearance system that is connected to instrument class.
cst_id_n (Customer Number)	
Datatype	UINT16_T
Description	A unique number that identified the member, used when subscribing for directed broadcast information.
currency_code (CURRENCY_CODE)	
Datatype	char[3]
Description	Intermediate field.
currency_format_c (Currency Format)	
Datatype	UINT8_T
Description	Not applicable.
currency_s (Currency)	
Datatype	char[3]
Description	Defines the type of currency. The representation of the currency follows the S.W.I.F.T. handbook and ISO 3166 standard, e.g. SEK, GBP, USD and ATS.
cur_unit_c (Currency Unit)	
Datatype	UINT8_T
Description	Specifies the currency unit for underlying prices.
Value Set	**name** / **value** Primary Unit / 1 Secondary Unit / 2 Tertiary Unit / 3
customer_info_s (Customer, Information)	
Datatype	char[15]

Description	This is a free text field a customer may fill in when entering orders. If the order is traded, the customer information is returned in the trade record.
cust_bank_id_s (Custodian Bank)	
Datatype	char[12]
Description	Identity of custodian bank
date_adjust_s (Date, Adjust)	
Datatype	char[8]
Description	Date of the adjustment. In ASCII format: YYYYMMDD
date_booksclose_s (Booksclose Date)	
Datatype	char[8]
Description	Customer Specific field. Booksclose date for bond underlying, YYYYMMDD.
date_closing_s (Date, Closing)	
Datatype	char[8]
Description	Closing date YYYYMMDD.
date_conversion_s (Date, Conversion)	
Datatype	char[8]
Description	Date in ASCII. Format: YYYYMMDD
date_coupdiv_s (Coupon/Dividend Date)	
Datatype	char[8]
Description	Coupon date for bond underlying or dividend date for stock underlying, YYYYMMDD.
date_dated_s (Date, Dated)	
Datatype	char[8]
Description	Dated date for bond underlying, YYYYMMDD.
date_delivery_start_s (Date, Delivery Start)	
Datatype	char[8]
Description	Delivery start date. Format: YYYYMMDD.
date_delivery_stop_s (Date, Delivery Stop)	
Datatype	char[8]
Description	Delivery stop date. Format: YYYYMMDD.
date_first_trading_s (Date, First Trading)	
Datatype	char[8]
Description	The first valid trading date of the series. The date is together with TIME, FIRST TRADING distributed as UTC. Format: YYYYMMDD.
date_implementation_s (Date, Implementation)	
Datatype	char[8]
Description	Implementation date. Format: YYYYMMDD.
date_last_s (Date, Last)	

Datatype	char[8]
Description	Last trading date YYYYMMDD.
date_last_trading_s (Date, Last Trading)	
Datatype	char[8]
Description	The last valid trading date of the series. The date is together with TIME, LAST TRADING distributed as UTC. Format: YYYYMMDD.
date_non_trading_s (Date, Non Trading)	
Datatype	char[8]
Description	Non trading date in format YYYYMMDD.
date_notation_s (Date, Notation)	
Datatype	char[8]
Description	Notation date YYYYMMDD
date_proceed_s (Date, Proceed)	
Datatype	char[8]
Description	Proceed date for fixed income underlying, YYYYMMDD If the last payment falls on a non-business day, the payment and the maturity is pushed forward to the next business day, the so called Proceeds Date.
date_release_s (Date, Issue)	
Datatype	char[8]
Description	Issue date for fixed income underlying. Format: YYYYMMDD.
date_s (Date)	
Datatype	char[8]
Description	Date in ASCII. Format: YYYYMMDD
date_settlement_s (Date, Settlement)	
Datatype	char[8]
Description	Settlement date for delivery or payment. Format YYYYMMDD.
date_termination_s (Date, Maturity)	
Datatype	char[8]
Description	Maturity date for fixed income underlying, YYYYMMDD.
date_trading_s (Date, Trading)	
Datatype	char[8]
Description	Date in ASCII. Format: YYYYMMDD.
days_in_interest_year_n (Days In Interest Year)	
Datatype	UINT16_T
Description	Number of days in coupon period used for interest rate calculations.
day_calc_rule_c (Day Calculation Rule)	

Datatype	UINT8_T
Description	Day Calculation Rule
Value Set	**name** / **value** ACTACT / 1 ACTAFB / 2 EU30360 / 3 US30360 / 4 ACT365 / 5 ACT360 / 6
day_count_n (Day Count)	
Datatype	UINT16_T
Description	Number of days in the year when calculating interest.
db_operation_c (Operation)	
Datatype	UINT8_T
Description	Operation to do for the item. Note:An insert might come for an existing item in the front-end. An update might come for a non-existing item in the front-end.
Value Set	**name** / **value** Insert / 1 Update / 2 Remove / 3
dc_deal_state_c (State, Deal)	
Datatype	UINT8_T
Description	Defines the state of the deal.
Value Set	**name** / **value** / **description** Normal / 1 / The TM deal has been accepted by the Clearinghouse. Holding_matched / 15 / The trade report is not yet accepted by the Clearinghouse.
deal_info_n (Deal Information)	
Datatype	UINT16_T
Description	Information of a deal.
Value Set	**name** / **value** / **description** deal_info_no_info / 0 / No info

name	value	description
reported_trade	1	Reported Trade
aon	2	All or none
part_of_combo_match	4	Part of combo match

deal_number_i (Deal Number)	
Datatype	INT32_T
Description	A number that identifies a specific deal. Deal number is unique within Instrument type
deal_price_i (Price, Deal)	
Datatype	INT32_T
Description	Defines the deal price.
deal_price_modifier_c (Modifier, Deal Price)	
Datatype	UINT8_T
Description	The modifier is used to recalculate the item after an underlying adjustment. The field is stored with 3 implicit decimals.

Value Set

value	description
1	Modifier is added to the item
2	Modifier is subtracted from the item
3	Modifier is multiplied with the item
4	The item is divided by the modifier factor

deal_price_mod_factor_i (Modifier Factor, Deal Price)	
Datatype	INT32_T
Description	The modifier is used to recalculate the item after an underlying adjustment. The field is stored with 7 implicit decimals
deal_quantity_i (Quantity, Deal)	
Datatype	INT64_T
Description	Defines number of contracts in a deal.
deal_source_c (Deal Source)	
Datatype	UINT8_T
Description	Refers to where the deal is created during the day :

Value Set

name	value	description
deal_source_none	0	Internal use. Trades reported directly to the clearing subsystem.
deal_source_auto	1	Matched by system, automatically.
deal_source_manually	2	Matched by system, manually.

name	value	description
deal_source_outside_different	3	Matched Outside Exchange, Different participants
deal_source_outside_different_om	4	Matched outside exchange, different brokers, reg. by exchange.
deal_source_outside_same	5	Matched Outside Exchange, One participant
deal_source_outside_same_om	6	Matched outside exchange, one broker, reg. by exchange.
deal_source_auto_combo	7	Combination order matched against another combination order when matched by the Exchange, electronically.
deal_source_auto_internal	9	Matched electronically, member internal.
deal_source_manual_reversing	16	Reversing deal made by the exchange manually.
deal_source_basis_trade	17	Basis trade.
deal_source_correction	18	Correction of trade.
deal_source_internally_created	19	Internally created.
deal_source_open_allocation	20	Deal made at the end of an auction.
deal_source_tm_combo	36	Tailor-made combination.
deal_source_non_std_combo	37	Non-standard combination.
deal_source_block_trade_fac	38	Outside the Exchange, block trade facility.
deal_source_outside_combo	39	Matched outside the Exchange, combinations.
deal_source_external_vendor	40	Outside the Exchange, block trade facility.
deal_source_combo_vs_outright	43	Combination matched outright legs.
deal_source_av_price_trade	128	Trade resulting from an Average Price Trade transaction.
deal_source_intermediate_apt	129	Intermediate trade created in an Average Price Trade transaction.
deal_source_transfer_with_price	131	Trade transfer.
deal_source_transfer_misclear	132	Misclear.
deal_source_efp	133	Exchange for physical (EFP).
deal_source_spread	134	Spread trade.

name	value	description
deal_source_aps	135	Average price system (APS).
deal_source_ad-just_wo_price	136	Adjustment without price.
deal_source_ad-just_with_price	137	Adjustment with price.
deal_source_cross_prod-uct_netting	139	Cross product netting.

deal_source_n (Deal Source)	
Datatype	INT16_T
Description	This is used when retrieving translations of deal source values (see DEAL_SOURCE_C).
dec_in_contr_size_n (Decimals, Contract Size)	
Datatype	UINT16_T
Description	Number of implicit decimals in the Contract Size and the Price Quotation Factor fields.
dec_in_deliv_n (Decimals, Delivery)	
Datatype	UINT16_T
Description	Number of implicit decimals used in the delivery quantity.
dec_in_fixing_n (Decimals, Fixing)	
Datatype	UINT16_T
Description	Number of implicit decimals in Fixing.
dec_in_nominal_n (Decimals, Nominal)	
Datatype	UINT16_T
Description	Number of implicit decimals in the Nominal Value.
dec_in_premium_n (Decimals, Premium)	
Datatype	UINT16_T
Description	Number of implicit decimals in the premium/price.
dec_in_price_n (Decimals, Price)	
Datatype	UINT16_T
Description	Number of implicit decimals in the underlying price received from external sources.
dec_in_rate_n (Decimals, Rate)	
Datatype	UINT16_T
Description	Number of implicit decimals in Rate.
dec_in_strike_price_n (Decimals, Strike Price)	
Datatype	UINT16_T
Description	Number of implicit decimals in the strike price.
deferred_publication_c (Deferred Publication)	
Datatype	UINT8_T

Description	Defines if the publication of a trade report should be deferred or not
Value Set	**name** / **value** Yes / 1 No / 2
deliverable_c (Deliverable)	
Datatype	UINT8_T
Description	Defines if a series can be delivered or not (Cash settlement):
Value Set	**value** / **description** 1 / Yes 2 / No
delivery_number_i (Delivery, Number)	
Datatype	INT32_T
Description	The delivery number for this delivery. Together with key number and series it is a unique number.
delivery_origin_i (Delivery Origin)	
Datatype	INT32_T
Description	The trade number for the trade that this delivery originates from. Together with Series it forms a unique trade identification.
delivery_properties_u (Delivery Properties)	
Datatype	UINT32_T
Description	Bit mask provides specific information about the delivery.
Value Set	**value** / **description** 0 / No information 1 / DvP (Create DvP instruction) 2 / SWIFT (Entered by SWIFT) 4 / Transfer (Other quantity is zero) 8 / Reversing (Reversing BD18) 16 / Overtaking (Overtaking BD18) 32 / Confirm (Holding DvP instruction needs confirmation) 64 / Settled Ext (Don't create DvP instruction - the delivery will be settled externally) 128 / Bill Delivery 256 / Cross Clearinghouse Balance 512 / Cross Border Give-up 1024 / Additional Basket 8192 / Do not reverse the sign of this delivery item

<table>
<tr><td colspan="2">delivery_quantity_q (Quantity, Delivery)</td></tr>
<tr><td>Datatype</td><td>INT64_T</td></tr>
<tr><td>Description</td><td>Defines the quantity the delivery is based on.</td></tr>
<tr><td colspan="2">delivery_state_c (Delivery, State)</td></tr>
<tr><td>Datatype</td><td>UINT8_T</td></tr>
<tr><td>Description</td><td>Defines what state the delivery is in.</td></tr>
<tr><td>Value Set</td><td><table>
<tr><th>value</th><th>description</th></tr>
<tr><td>1</td><td>Normal</td></tr>
<tr><td>2</td><td>Rectified
The delivery is rolled back. There exists another rollback delivery that points to this delivery.</td></tr>
</table></td></tr>
<tr><td colspan="2">delivery_type_c (Delivery, Type)</td></tr>
<tr><td>Datatype</td><td>UINT8_T</td></tr>
<tr><td>Description</td><td>Defines what type the delivery is.</td></tr>
<tr><td>Value Set</td><td><table>
<tr><th>value</th><th>description</th></tr>
<tr><td>1</td><td>Normal</td></tr>
<tr><td>2</td><td>Rollback
The delivery offsets a previous delivery that is no longer valid. The original delivery is identified by the instrument type of the series in combination with original delivery number and original key number. The quantity delivery base reverse the original delivery.</td></tr>
<tr><td>3</td><td>Overtaking
The delivery superseeds a previous delivery that is no longer valid. The original delivery is identified by the instrument type of the series in combination with original delivery number and original key number.</td></tr>
<tr><td>4</td><td>Backdated
The delivery is backdated which entails that it concerns an event occuring on a previous clearing date.</td></tr>
</table></td></tr>
<tr><td colspan="2">delivery_unit_u (Delivery Unit)</td></tr>
<tr><td>Datatype</td><td>UINT32_T</td></tr>
<tr><td>Description</td><td>Trade reports, delivery items and dvp-instructions belong to a delivery unit.</td></tr>
<tr><td colspan="2">deliv_base_quantity_q (Quantity, Delivery Base)</td></tr>
<tr><td>Datatype</td><td>INT64_T</td></tr>
<tr><td>Description</td><td>Defines the quantity of the delivery base that is delivered.</td></tr>
<tr><td colspan="2">delta_i (Delta)</td></tr>
</table>

Datatype	INT32_T
Description	The rate of change in an options value, due to a change in the price of the underlying. Given with 4 decimals.
delta_protection_q (Delta protection)	
Datatype	INT64_T
Description	Specifies the limit of the delta value per underlying within the exposure time interval when market maker protection is triggered. When this value is exceeded the system automatically removes the quotes for the instruments connected to the underlying. A value of 0 means that no delta protection exists.
delta_quantity_c (Delta Quantity)	
Datatype	UINT8_T
Description	When changing quantities there are two options: delta and absolute. Delta changes amend the quantity/total volume of an order by the given amount, positive to increase the quantity, negative to reduce the quantity. Absolute changes means that the quantity/total volume should be set to the value in the quantity/total volume field.
Value Set	**value** / **description** 1 / Absolute quantity 2 / Delta quantity
demands_populated_c (Demands, Populated)	
Datatype	UINT8_T
Description	Defines if demands are populated or not.
Value Set	**value** / **description** 1 / Yes 2 / No
demand_u (Demand)	
Datatype	INT64_T
Description	Total volume of contracts.
deny_exercise_q (Deny Exercise)	
Datatype	INT64_T
Description	The number of held position that will NOT participate in exercise.
derivate_level_n (Derivate Level)	
Datatype	UINT16_T
Description	The derivate level of the instrument:
Value Set	**name** / **value** Spot / 0 Derivate based on spot. / 1 Derivative based on instrument level 1. / 2

<table>
<tr><td colspan="2">description_s (Description)</td></tr>
<tr><td>Datatype</td><td>char[40]</td></tr>
<tr><td>Description</td><td>Description field.</td></tr>
<tr><td colspan="2">desc_abbreviated_s (Description, Abbreviated)</td></tr>
<tr><td>Datatype</td><td>char[32]</td></tr>
<tr><td>Description</td><td>An abbreviated textual description.</td></tr>
<tr><td colspan="2">desc_long_s (Description, Long)</td></tr>
<tr><td>Datatype</td><td>char[128]</td></tr>
<tr><td>Description</td><td>A textual description.</td></tr>
<tr><td colspan="2">destination_level_c (Destination, Level)</td></tr>
<tr><td>Datatype</td><td>UINT8_T</td></tr>
<tr><td>Description</td><td>Defines the destination level.</td></tr>
<tr><td>Value Set</td><td><table>
<tr><th>name</th><th>value</th><th>description</th></tr>
<tr><td>DESTINATION_LEVEL_MARKET</td><td>1</td><td>Market level</td></tr>
<tr><td>DESTINATION_LEVEL_UNDERLYING</td><td>2</td><td>Underlying level</td></tr>
<tr><td>DESTINATION_LEVEL_SERIES</td><td>3</td><td>Series level</td></tr>
</table></td></tr>
<tr><td colspan="2">diary_number_s (Diary Number)</td></tr>
<tr><td>Datatype</td><td>char[15]</td></tr>
<tr><td>Description</td><td>The diary number for this account.</td></tr>
<tr><td colspan="2">difflen (DIFFLEN)</td></tr>
<tr><td>Datatype</td><td>char[8]</td></tr>
<tr><td>Description</td><td>intermediate field.</td></tr>
<tr><td colspan="2">directed_trade_information_c (Directed Trade Information)</td></tr>
<tr><td>Datatype</td><td>UINT8_T</td></tr>
<tr><td>Description</td><td>Specifies how the directed trade broadcast is distributed.</td></tr>
<tr><td>Value Set</td><td><table>
<tr><th>name</th><th>value</th></tr>
<tr><td>Without Counterparty</td><td>1</td></tr>
<tr><td>With Counterparty</td><td>2</td></tr>
</table></td></tr>
<tr><td colspan="2">display_quantity_i (Quantity, Display)</td></tr>
<tr><td>Datatype</td><td>INT64_T</td></tr>
<tr><td>Description</td><td>The quantity that is originally displayed in the field mp_quantity_i for orders using the hidden volume order concept. This is the maximum quantity that the mp_quantity_i field will be repopulated with when the quantity reaches zero.</td></tr>
<tr><td colspan="2">dividend_i (Dividend)</td></tr>
</table>

Datatype	UINT32_T
Description	The dividend for the stock.
dividend_yield_i (Dividend, Yield)	
Datatype	INT32_T
Description	The dividend yield used in evaluations. Expressed in percent with 4 implicit decimals.
download_ref_number_q (Download Reference Number)	
Datatype	INT64_T
Description	Reference number to use in delta queries and answers. To receive the delta use the latest received number from the answer of this query or the latest broadcast related to the query. To enforce a full answer use "no value" in the query to indicate this. This number is always increasing, but may contain gaps.
down_int_i (Valuation Interval, Down)	
Datatype	INT32_T
Description	Valuation interval down in margin calculations. Expressed in percent of underlying price. Represented with 4 implicit decimals.
ds_attribute_q (Deal Source Attribute)	
Datatype	INT64_T
Description	Defines the attribute of the deal source, different behaviors may be controlled by the attribute. 0 = Unassigned Bit 1 = Trade Report Bit 2 = Bulletin board Bit 3 = Excluded from Trade Statistics Bit 4 = Outside exchange
dvp_account_s (DVP Account)	
Datatype	char[24]
Description	Sub account/Security account or Cash record/Cash account identification designated for deliveries.
edited_ob_changes_avail_c (Edited Price Information Available)	
Datatype	UINT8_T
Description	Price Information broadcasts available during the state.
Value Set	**value** / **description** 1 / Yes 2 / No
edited_price_info_reason_c (Reason for Edited Price Information update)	
Datatype	UINT8_T
Description	Reason why the Edited Price Information broadcast was distributed

Value Set	<table><tr><th>name</th><th>value</th><th>description</th></tr><tr><td>edited_price_reason_none</td><td>0</td><td>Void and not used</td></tr><tr><td>edited_price_reason_refresh</td><td>1</td><td>Sent due to refresh of data</td></tr><tr><td>edited_price_reason_deal</td><td>2</td><td>Sent due to execution of deal</td></tr><tr><td>edited_price_reason_cor</td><td>3</td><td>Sent due to correction of data</td></tr><tr><td>edited_price_reason_delete</td><td>4</td><td>Sent due to deletion of deal</td></tr><tr><td>edited_price_reason_exclude</td><td>5</td><td>Sent due to exclusion of deal in trade statistics</td></tr><tr><td>edited_price_reason_include</td><td>6</td><td>Sent due to reinclusion of deal in trade statstics</td></tr><tr><td>edited_price_reason_reset</td><td>7</td><td>Sent due to reset of trade statstics</td></tr></table>
effective_exp_date_s (Effective Expiration Date)	
Datatype	char[8]
Description	The effective expiration date is the actual expiration date of the series and will normally be the same as expiration_date_n in the series binary code. The effective expiration date can be changed during the lifetime of the series whereas expiration_date_n will continue to hold the original expiration date. Format: YYYYMMDD.
end_date_s (Date, End)	
Datatype	char[8]
Description	End date. Format: YYYYMMDD.
end_of_clearing_day_c (End of Clearing Day)	
Datatype	UINT8_T
Description	Indicates if this state is the start for trading on T+1 basis, implying that such trades will be subject to After Business processing the following clearing day.
Value Set	<table><tr><th>value</th><th>description</th></tr><tr><td>1</td><td>Yes</td></tr><tr><td>2</td><td>No</td></tr></table>
equilibrium_price_i (Price, Equilibrium)	
Datatype	INT32_T
Description	The equilibrium price is calculated by the exchange during trading phases where order matching is disabled. The exact rules for its calculation is exchange specific.
equilibrium_quantity_i (Equilibrium Volume)	
Datatype	INT64_T
Description	The quantity possible to match if an uncrossing of the market should occur.The equilibrium volume is calculated by the exchange during trading phases where order matching is disabled.
eqy_combo_trade_pos_n (Equity Combo Trade, Trade Position)	
Datatype	UINT16_T

Description	Holds current trade position within an equity combo deal.
eqy_combo_trade_seq_n (Equity Combo Trade, Counter)	
Datatype	UINT16_T
Description	Holds a counter for equity combo trades.
eqy_combo_trade_tot_n (Equity Combo Trade, Total Value)	
Datatype	UINT16_T
Description	Holds a total value of trades for an equity combo deal.
erosion_i (Erosion Adjustment)	
Datatype	INT32_T
Description	In margin calculations, number of days to maturity will be reduced by this number for held options. Represented with 1 implicit decimal.
error_id_u (Error Identity)	
Datatype	UINT32_T
Description	An identity that refers to the source for error. For trade errors, this is the trade number.
error_operation_s (Error, Operation)	
Datatype	char[10]
Description	Defines what type of operation caused the error message.
error_problem_s (Error, Problem)	
Datatype	char[40]
Description	The error message.
er_trd_days_in_year_i (Trading Days Erosion)	
Datatype	INT32_T
Description	Trading days per year in margin calculations for erosion adjustment for held options.
event_type_i (Stimuli Event)	
Datatype	INT32_T
Description	Defines the reason that caused the contractual event.

Value Set

value	description
1	Trade
2	Transfer
3	Rectify
4	Mark to Market
5	Closing
6	Exercise
7	Assign
8	Dividend
9	New Contract Trade
10	Give Up

value	description
11	Closing Trade
12	Delivery Flow
13	DVP Settled

exchange_info_cl_s (Exchange Information)	
Datatype	char[32]
Description	This is an exchange specific field and may be used as convenient, as a free text field, for example.

exchange_info_s (Exchange, Information)	
Datatype	CHAR[32]
Description	This is an exchange specific field and can be used for different purposes, e.g. as a free text field.

exchange_short_s (Exchange, Short Name)	
Datatype	char[4]
Description	Short name for exchange

exch_order_type_n (Order Type, Exchange)	
Datatype	UINT16_T
Description	This is bit-coded field for exchange specific order types and attributes.

Value Set

name	value	description
EXCH_ORDER_TYPE_NOT_DEFINED	0	Not applicable.
EXCH_ORDER_TYPE_FORCE	1	Force
EXCH_ORDER_TYPE_SHORT_SELL	2	Short Sell Short sell order condition.
EXCH_ORDER_TYPE_MARKET_BID	4	Market Bid Market bid order condition(exchange specific).
EXCH_ORDER_TYPE_PRICE_STAB	8	Price Stabilization Price stabilization order condition (exchange specific).
EXCH_ORDER_TYPE_OVERRIDE_CRS	16	Override Crossing Override crossing condition (exchange specific).
EXCH_ORDER_TYPE_UNDISCLOSED	32	Undisclosed
EXCH_ORDER_TYPE_CENTRE_POINT	64	Centre Point
EXCH_ORDER_TYPE_ALWAYS_INACTIVE	128	Always Inactive Always centrally inactive order, not possible to activate.

name	value	description
		Only valid for transactions to enter inactive orders (exchange specific).
EXCH_ORDER_TYPE_CPPX	256	Centre Point Priority Crossing
EXCH_ORDER_TYPE_SESSION_STATE	512	Sleeping order on entry When the active Session State is changed to the one given in the order, the order is triggered and entered into the order book
EXCH_ORDER_TYPE_T_PLUS_ONE	2048	T plus one order The order is valid also in T plus one trading.

exclusive_opening_sell_c (Exclusive Opening Sell)	
Datatype	UINT8_T
Description	Is the account allowed to exclusive opening sell?

Value Set

value	description
1	Yes
2	No

execution_event_nbr_u (Execution number)	
Datatype	UINT64_T
Description	An ever increasing number per partition, assigned to an execution event.

exercisenumber (EXERCISENUMBER)	
Datatype	INT32_T
Description	intermediate field.

exercise_number_i (Exercise, Request Number)	
Datatype	INT32_T
Description	Identifies each part in an exercise request.

exerc_limit_i (Exercise, Limit)	
Datatype	INT32_T
Description	The limit from the at-the-money value when an automatic exercise is done. If the Unit is Percent, this value is stored with 6 implicit decimals. E.g. 10 % is stored as 10000. If the unit is an absolute value this value is stored with 3 implicit decimals.

exerc_limit_unit_c (Exercise, Limit Unit)	
Datatype	UINT8_T
Description	What type is the Exercise Limit Unit?

Value Set	**value** / **description** 1 / Absolute Value 2 / Percentage (%)
expiration_date_n (Date, Expiration)	
Datatype	UINT16_T
Description	Expiration date of financial instrument. A bit pattern is used. The seven most significant bits are used for year, the next four for month and the five least significant bits for day. All these bits make up an unsigned word. The year-field starts counting from 1990. Thus, 1990=1, 1991=2 ... 2001=12. Example: January 1, 1990: Binary: 0000001 0001 00001 year month day 7 bits 4 bits 5 bits Decimal: 545
exposure_time_interval_i (Exposure Time Interval)	
Datatype	INT32_T
Description	Specifies the rolling time interval in seconds used in quantity/delta protection calculations.
extended_info_n (Extended Information)	
Datatype	UINT16_T
Description	Not applicable.
Value Set	**value** / **description** 0 / Not Applicable
external_fee_type_c (External Fee Type)	
Datatype	UINT8_T
Description	The external fee type is used to look up the fee table that will be used to calculate the fee for the trade
external_full_depth_c (Full Depth, External)	
Datatype	UINT8_T
Description	Not applicable.
Value Set	**value** / **description** 2 / No
external_id_s (External Price Feed Identity)	
Datatype	char[40]
Description	External Price feed identity
ext_acc_controller_s (External Account Controller)	
Datatype	char[15]
Description	External account controller. May hold BIC, CSD member id etc.
ext_acc_id_s (External Account ID)	
Datatype	char[34]

Description	External account id. A bank or CSD account number.
ext_acc_registrar_s (External Account Registrar)	
Datatype	char[12]
Description	External account registrar. May hold names like VPS, SWIFT etc.
ext_info_source_c (External Information Source)	
Datatype	UINT8_T
Description	Specifies whether or not the data source for distributed prices is sent into the system with an external transaction.
Value Set	**name** / **value** Yes / 1 No / 2
ext_or_int_c (User Type)	
Datatype	UINT8_T
Description	If the user type is external or internal:
Value Set	**value** / **description** 1 / External 2 / Internal
ext_provider_c (External Price Feed Provider)	
Datatype	CHAR
Description	External Price feed provider
Value Set	**name** / **value** NMF / N Six / S Six OMX / O Direct Feed / F Direct Feed OPRA / R Transaction / T LMIL / L Reuter SSL / E
ext_seq_nbr_i (External Clearinghouse, Sequence Number)	
Datatype	INT32_T
Description	An identity that the clearinghouse or exchange can assign to a trade. Exchange specific.
ext_status_i (Return Status)	
Datatype	INT32_T

Description	Defines return status, configuration specific.
ext_time_s (Time, External)	
Datatype	char[6]
Description	External time, given by the Stock Exchange. Format: HHMMSS
ext_trade_fee_type_c (External Trade, Fee Type)	
Datatype	CHAR
Description	The external fee type is used to look up the fee table that will be used to calculate the fee for the trade.
ext_trade_number_u (Trade Number, External)	
Datatype	UINT32_T
Description	Trade number assigned by external system
ext_t_state_c (Trade Report Type)	
Datatype	UINT8_T
Description	Defines the type of Trade Report. The available types can be retrieved by Query Trade Report.
Value Set	**value** / **description**: 0 – Not applicable.
ex_client_s (Client)	
Datatype	char[10]
Description	Exchange client is the name of the participant's client.
ex_coupon_n (Period, Ex Coupon)	
Datatype	UINT16_T
Description	Ex Coupon period
ex_customer_s (Customer, Identity)	
Datatype	char[5]
Description	This field together with Country Name, identifies a member/participant of the exchange (such as a bank or broker firm).
failure_reason_s (Failure Reason)	
Datatype	char[160]
Description	Free text describing why margin simulation has failed. Blank in case of success.
fee_type_s (Account Fee Type)	
Datatype	char[12]
Description	Defines the account fee type for an account.
file_type_s (File Type)	
Datatype	char[8]
Description	The string representing the file type, i.e. suffix.
filler_1_s (Filler)	
Datatype	CHAR

Description	Filler for alignment.
filler_2_s (Filler)	
Datatype	char[2]
Description	Filler for alignment.
filler_3_s (Filler)	
Datatype	char[3]
Description	Filler for alignment.
filler_40_s (Filler)	
Datatype	char[40]
Description	Filler for alignment
filler_4_s (Filler)	
Datatype	char[4]
Description	Filler
filler_6_s (Filler)	
Datatype	char[6]
Description	Filler for alignment
filler_7_s (Filler)	
Datatype	char[7]
Description	Filler for alignment
filler_8_s (Filler)	
Datatype	char[8]
Description	Filler for alignment.
fill_and_kill_allowed_c (Fill and Kill Allowed)	
Datatype	UINT8_T
Description	Fill and Kill allowed during the state.
Value Set	**value** / **description** 1 / Yes 2 / No
fill_or_kill_allowed_c (Fill or Kill Allowed)	
Datatype	UINT8_T
Description	Fill or Kill allowed during the state.
Value Set	**value** / **description** 1 / Yes 2 / No
first_settlement_date_s (Date, First Settlement)	

Datatype	char[8]
Description	First Settlement Date in format YYYYMMDD.
fixed_income_type_c (Fixed Income Type)	
Datatype	UINT8_T
Description	Type of fixed income security:
Value Set	**value** / **description** 0 / Not applicable 1 / Bill 2 / Bond 3 / Index Linked Bonds 4 / Bond Floating 5 / Lottery Bond 6 / Convertible Bond 7 / Structured Bond 8 / Fixing 9 / Credit Certificates 10 / Deposit 11 / RIBA
fixed_vol_i (Volatility, Fixed)	
Datatype	INT32_T
Description	For those options that use fixed volatility in margin calculations, this field is the volatility used. For other options, this is the fallback volatility when calculating theoretical prices. Expressed in percent, 4 last digits represent decimals.
fixing_req_c (FIXING_REQ_C)	
Datatype	UINT8_T
Value Set	**name** / **value** Yes / 1 No / 2
fixing_value_i (Fixing Value)	
Datatype	INT32_T
Description	A value defined for a series a given date, used for clearing purposes. The Decimals, Fixing field defines the number decimals used.
fix_rate_down_i (Fix Rate Step Down)	
Datatype	INT32_T
Description	Steps for fix rate when evaluating swaps in margin calculations. Expressed in percent with 4 implicit decimals.
fix_rate_up_i (Fix Rate Step Up)	

Datatype	INT32_T
Description	Steps for fix rate when evaluating swaps in margin calculations. Expressed in percent with 4 implicit decimals.
flat_rate_decrease_i (Flat rate decrease)	
Datatype	INT32_T
Description	Always equal zero.
flat_rate_gain_discount_i (Flat rate gain discount)	
Datatype	INT32_T
Description	Always equal zero.
flat_rate_increase_i (Flat rate increase)	
Datatype	INT32_T
Description	Always equal zero.
float_swap_steps_i (Float Rate Steps)	
Datatype	INT32_T
Description	Number of float rate steps for swaps in margin calculations.
forward_style_c (Style, Forward)	
Datatype	UINT8_T
Description	Defines if this an Instrument Group where corresponding Instrument Series are forward styled.
Value Set	**value** / **description** 0 / Not applicable 1 / Normal 2 / CfD
forw_margin_c (Forward Margining)	
Datatype	UINT8_T
Description	Defines the type of forwards margining
Value Set	**name** / **value** Forward Style / 0 Future Style / 1 Power Spot Style / 2 Zero-day forward style / 3
free_text_80_s (Text , Free)	
Datatype	char[80]
Description	Defines a free text buffer.
from_date_s (Date, From)	
Datatype	char[8]

Description	From date. Format: YYYYMMDD.
from_sequence_number_u (From Sequence Number)	
Datatype	UINT32_T
Description	From Sequence Number
from_time_s (Time, From)	
Datatype	char[6]
Description	Defines the from time. Format: HHMMSS.
frozen_time_i (Frozen Time)	
Datatype	INT32_T
Description	Specifies the time interval in seconds when quotes are rejected after Market Maker protection has been triggered.
full_answer_c (Full Answer)	
Datatype	UINT8_T
Description	A full answer is enforced in the delta query.
Value Set	**name** / **value** Yes / 1 No / 2
future_styled_c (Option, Future Styled)	
Datatype	UINT8_T
Description	If the option is a future styled option:
Value Set	**value** / **description** 1 / Yes 2 / No
fut_pl_sim_c (Futures profit/loss Simulated)	
Datatype	UINT8_T
Description	Flags if profit/loss for futures and future styled options should be included in margin simulation.
Value Set	**value** / **description** 0 / Not included. 1 / Included.
fut_spread_rate_i (Future Spread Rate)	
Datatype	INT32_T
Description	Future spread rate used in margining by the cardinal method.
gamma_i (Gamma)	
Datatype	INT32_T

Description	The rate of change in an options delta, due to a change in the price of the underlying. Given with 4 decimals.
give_up_number_i (Give Up, Number)	
Datatype	INT32_T
Description	Unique, within each instrument type (country, market, instrument group) system generated number, for a give-up.
give_up_state_c (Give Up, State)	
Datatype	UINT8_T
Description	Indicates the state of the give up the trade may be subject to. The value is a bit mask and can be one of the following:
Value Set	**value** / **description** 0 / None 1 / Holding 2 / Confirmed . 4 / Rejected . 8 / Holding Rectify Trade . 16 / Holding Rectify Deal . 32 / Deleted . 64 / Delete Holding .
give_up_text_s (Give Up, Free Text)	
Datatype	char[30]
Description	User-supplied information to a give-up request. This information is passed through the clearing system without any processing or validation.
giving_up_exchange_s (Giving Up Exchange)	
Datatype	char[2]
Description	The exchange of the owner of the trade that was given up.
global_deal_no_u (Global Deal Number)	
Datatype	UINT32_T
Description	A number that together with series identifies a specific deal. The number is used as reference from outside clearing system.
gross_open_interest_q (Gross Open Interest)	
Datatype	UINT64_T
Description	Defines gross open interest.

gross_or_net_c (Gross Or Net)	
Datatype	UINT8_T
Description	Defines if current value is gross or net calculated.
Value Set	

name	value	description
Gross	0	Gross
Net	1	Net

group_short_name_s (Short Name, Instrument Group)	
Datatype	char[15]
Description	Defines a short description of the instrument group.

group_type_c (Group, Type)	
Datatype	UINT8_T
Description	Defines the type of instrument group.
Value Set	

name	value	description
group_type_undefined	0	Undefined
group_type_option	1	Option
group_type_forward	2	Forward
group_type_future	3	Future
group_type_fra	4	FRA
group_type_cash	5	Cash
group_type_payment	6	Payment
group_type_exchange_rate	7	Exchange Rate
group_type_inter-est_rate_swap	8	Interest Rate Swap
group_type_repo	9	REPO
group_type_synth_box_leg	10	Synthetic Box Leg/Reference
group_type_standard_combo	11	Standard Combination
group_type_guarantee	12	Guarantee
group_type_otc_general	13	OTC General
group_type_equity_warrant	14	Equity Warrant
group_type_security_lending	15	Security Lending

gup_reason_i (Give Up, Broadcast Reason)	
Datatype	INT32_T
Description	Defines the reason why the Directed Give Up broadcast was sent.
Value Set	

value	description
1	Holding

value	description
2	Confirmed .
3	Rejected .
4	Delete Holding .
5	Deleted .
6	Extended .

heartbeat_interval_c (Heartbeat Interval)	
Datatype	UINT8_T
Description	The interval in seconds between heartbeats sent out.
held_for_adj_i (Future Adjustment Held)	
Datatype	INT32_T
Description	Adjustment factor for margin calculation of held futures and forwards. Expressed in percent with 4 implicit decimals.
held_marg_q (Marginables, Held)	
Datatype	INT64_T
Description	The number of held marginables in a position.
held_val_max_i (Spread Minimum)	
Datatype	INT32_T
Description	Highest value for a held option in relation to a written option. Expressed in percent with 4 implicit decimals.
held_vol_down_i (Volatility Held Down)	
Datatype	INT32_T
Description	Volatility interval down for held options in margin calculations. Expressed in percent, 4 implicit decimals.
held_vol_max_i (Volatility Held, Max)	
Datatype	INT32_T
Description	Max volatility for held options. Expressed in percent with 4 implicit decimals.
held_vol_up_i (Volatility Held Up)	
Datatype	INT32_T
Description	Volatility interval up for held options in margin calculations. Expressed in percent, 4 implicit decimals.
held_zero_limit_i (Value Held Zero Limit)	
Datatype	INT32_T

Description	Held options with a value lower than this are considered worthless in margin calculations. Expressed with 4 implicit decimals.
hhmmss_s (Time, External)	
Datatype	char[6]
Description	Time in ASCII. Format: HHMMSS.
hidden_price_c (Hidden Price)	
Datatype	UINT8_T
Description	Defines if the price is hidden:
Value Set	**value** / **description** 0 / Not applicable. 1 / The price information in the broadcast is not valid and should not be used. 2 / The price information is valid.
hidden_vol_meth_n (Method, Hidden Volume)	
Datatype	UINT16_T
Description	Hidden Volume Method:
Value Set	**value** / **description** 0 / No hidden used 1 / Normal 2 / Additional
high_index_s (Index, Highest Value)	
Datatype	char[8]
Description	Highest index value for current day in ASCII format.
high_price_i (Price, High)	
Datatype	INT32_T
Description	Defines the highest traded price during the day.
hrm_corr_i (Reduction Correlation, Hourly)	
Datatype	INT32_T
Description	Defines correlation in margin calculations when using hourly reduction method. Expressed in percent with 4 implicit decimals.
identity (IDENTITY)	
Datatype	char[5]
Description	Intermediate field.
include_futures_c (Include futures)	
Datatype	UINT8_T
Description	Specifies if futures and forwards are to be included in the delta calculation.

Value Set	**name** / **value** Yes / 1 No / 2
include_in_statistics_c (Include in statistics)	
Datatype	CHAR
inc_id (INC_ID)	
Datatype	char[14]
Description	Intermediate field.
inc_id_s (Instrument Class, Identity)	
Datatype	char[14]
Description	The ASCII representation of the instrument class.
index_market_c (Index Market)	
Datatype	UINT8_T
Description	Indicates if the market is an index market or not
Value Set	**value** / **description** 1 / Yes 2 / No
index_s (Index, Identify)	
Datatype	char[15]
Description	The ASCII representation of the index name.
indicative_prices_c (Indicative Prices)	
Datatype	UINT8_T
Description	Indicative Prices
Value Set	**name** / **value** Yes / 1 No / 2
ind_ask_marg_vol_i (Margin, Individual Volatility Ask)	
Datatype	INT32_T
Description	Defines the latest volatility calculated for the series, where the value always is calculated from data for the series itself. For other instruments than options, the value is always zero. For series without positions, the volatility is calculated in the same way as if the series had positions. If it is impossible to calculate volatilities due to missing prices, the risk parameter imposed by the clearinghouse is returned. Expressed in percent, 4 implicit decimals.
ind_bid_marg_vol_i (Margin, Individual Volatility Bid)	
Datatype	INT32_T

Description	Defines the latest volatility calculated for the series, where the value always is calculated from data for the series itself. For other instruments than options, the value is always zero. For series without positions, the volatility is calculated in the same way as if the series had positions. If it is impossible to calculate volatilities due to missing prices, the risk parameter imposed by the clearinghouse is returned. Expressed in percent, 4 implicit decimals.
info_type_i (Information Type)	
Datatype	INT32_T
Description	The type of information ready:

Value Set

value	description
0	Used in queries to get available reports
1	Trade, position and delivery item information
2	Legacy clearing reports
3	Revising trade, position and delivery item information
4	Settlement information
5	Close of business
7	After Business started
8	Margin information
9	Margin vector information
10	Intra day margin call information ready
11	Margin summary information
12	New series next day ready
13	All securities closed
14	After Business completed
15	Day-end positions established
16	Exercise/delivery information
17	Open interest ready
18	After Business phase break
19	Fixing ready
20	All securities closed
22	Extracted data for report generating are ready (Kofex)
23	NRS batch data loaded completed
24	NRS batch data loaded started
26	Stock deliveries ready
27	Reversed Stock deliveries ready
28	Bilateral Delivery Instructions ready
32	Delivery
41	Margin Evening Prices ready

value	description
42	Intra Day Margin Calculation ready
43	Intra Day Greek Calculation ready
44	Intra Day Capital Based Position Limit calculation ready
45	Intra Day Reserve Fund calculation ready
46	Recalculated margin for previous day ready
47	Margin information from Lateevening ready
48	Margin summary information from Lateevening ready
49	API data from Intra Day Margin Calculation ready
52	Margin summary information from old dateready
53	Start owl cycle
54	Intra Day Margin Calculation product area ready
64	Expiration information
98	Final Fixing value established
100	Daily Trade statistics information
101	Revised Daily Trade statistics information
128	Paynote information
200	Official price ready (LME only)
201	Evening margin file ready (KOFEX specific)
202	Intra day margin file ready (KOFEX specific)
256	Used in queries to get possible reports
257	Vector files ready
260	Settlement note
261	Trades on trading account zero days forward
263	Settlement note futures
265	Settlement note ELEX
280	Cancellation note
285	Settlement notes, overtaking trades older than 1 day
290	Settlement note (position accounts)
291	Cancellation note (position accounts)
292	Settlement notes, overtaking... (position account)
293	Settlement note futures (position accounts)

value	description
300	Daily cash settlement futures
320	Error deals
325	Dividends, security lending
340	Exercise transaction list
341	Restoration, security lending
342	Trades per clearing account
344	Monthly cash settlement, security lending
350	Cash settlement options
351	Cash settlement forwards
352	Cash settlement forwards trading accounts
353	Cash settlement swaps
355	Monthly cash settlement forwards & IMM-FRA, detailed
356	Monthly cash settlement forwards & IMM-FRA
357	Expiration cash settlement forwards & IMM-FRA
358	Expiration cash settlement forwards & IMM-FRA/summary on account
359	Expiration cash settlement forwards & IMM-FRA/sumary on member
360	Expiration settlement FX Forwards
361	Expiration Tailor-Made Bond Forward
362	Cash settlement STINA
363	Accumulated Compound Rate STINA
370	Delivery
371	Delivery instruction security lending
373	Delivery advice summary
374	Delivery instruction collect note security lending
375	Delivery summary
376	Delivery fees new contracts
377	Delivery fees new contracts, summary on customer
379	DPMON Clearing Mgr Total Margin Req Summary
380	DPMON Product Area Collateral Summary
381	Margin and position listing
382	Margin requirement summary

value	description
383	Data used for margin calculation
384	Product area total collateral summary
385	Product area collateral summary
386	Security bank summary
387	Clearing manager summary
388	Clearing manager product area margin requirement summary
389	Clearing manager total margin requirement summary
390	Position and position overview
391	Non-propagated Margin and position listing
392	Member product area collateral summary
393	Evening Risk Parameter File, Central, Exchange 1
394	Evening Risk Parameter File, Central, Exchange 2
395	Intra Day Risk Parameter File, Central, Exchange 1
396	Intra Day Risk Parameter File, Central, Exchange 2
397	Preliminary Risk Parameter File, Central, Exchange 1
398	Preliminary Risk Parameter File, Central, Exchange 2
400	Delivery instruction stocks (net)
401	Delivery instruction bonds
403	Evening Risk Parameter File, Member, Exchange 1
404	Evening Risk Parameter File, Member, Exchange 2
405	Intra Day Risk Parameter File, Member, Exchange 1
406	Intra Day Risk Parameter File, Member, Exchange 2
407	Preliminary Risk Parameter File, Member, Exchange 1
408	Preliminary Risk Parameter File, Member, Exchange 2
410	Payment notes
411	Settlement amounts, customer
412	Separate fees
420	Changes of position

value	description
421	Accumulated amounts clearing accounts
422	In the money
423	Out of the money
424	Open Balance
426	Valid accounts
429	Accumulated amounts trading accounts
430	Trades/daily account
431	Rectified trades during the day
432	Position transfer trades during the day
433	Forecast closing
434	Forecast closing, summary
436	After hours trades
437	Customer Position Exceeding the Limits
438	Rebate per customer
439	FX clearing
440	FX expiration
441	Total margin requirements
442	Total settlement amounts
443	Power positions
444	Cascade options
445	Cascade forwards
446	Trades with counterparts
447	Trades per customer account with fees
448	Position not assign in exercise
449	FX Clearing, sorted by counterparts
450	Nord pool daily trade list
451	Nord pool clearing list summary for brokers
452	Nord pool clearing list
453	Pulpex option exercise note
454	Pulpex future expiration note
455	Clearing information on exercise, closing & markto-market
456	Discount per customer, rule and account
457	NOS fee list
458	Delivery note, zero-day forwards
459	Delivery note, summary

value	description
460	Trade counterparty report
501	Collateral held and activity
502	Option open positions
503	Futures open positions
504	Intra day risk - upside (Net)
505	Intra day risk - downside (Net)
506	Daily settlement reports (general clearing members)
507	Daily settlement reports
508	Margin activity reports
509	Cash transfer instructions (credit)
510	Cash transfer instructions (debit)
511	Options exercised and assigns
512	Consolidated positions activity (options)
513	Final contract reports (options)
514	Consolidated positions activity (futures)
515	Final contract reports (futures)
516	Monthly interest and accommodation
517	Monthly fees reports
518	Unsettled delivery report
519	Deliver/Receive reports
520	Exercise by exceptions
521	Options expired positions
522	Intra day margin activity reports
523	Give-up trades for executor
524	Give-up trades for clearing broker
525	Exercised/Expired options to be settled
541	DPMON margin and position
542	DPMON margin requirement summary
543	DPMON data used for margin calc
544	DPMON data used for margin calc CO
545	DPMON security bank summary
546	DPMON clearing manager summary
547	DPMON non-prop margin and position
548	DPMON margins
549	DPMON price alarm limit

value	description
550	DPMON price dump
551	SIMSRV price dump
552	IDMON margin and position
553	IDMON margin requirement summary
554	IDMON data used for margin calc
555	IDMON data used for margin calc CO
556	IDMON security bank summary
557	IDMON clearing manager summary
558	IDMON non-prop margin and position
559	IDMON margin report
560	IDMON price dump
561	RCAR worst
562	RCAR final scenario
563	RCAR top 10
564	RCAR detailed
566	DPMON Margin alarm limits
567	IDMON Margin alarm report
568	Risk parameter report
566	DPMON Margin alarm limits
590	DPMON Margin and position external
591	DPMON Data used for margin calc external
592	Data used for margin calc CO
593	Margin evening prices
594	Intray Param Change Report
595	Parameter Value Report
596	Window class Value Report
597	DPMON Parameter Value Report
598	DPMON Window class Value Report
600	Member order list report (CED only)
601	Member trade list report (CED only)
602	Market trades
603	Option Give up (for the executor member)
604	Option Give up (for the clearing broker member)
605	MS33 (CASSA report id)
606	MS59 (CASSA report id)

value	description
607	MS60 (CASSA report id)
608	Member stop order list report (CED only)
701	Assign ready (CED)
702	Theoretical ready (CED)
703	Class file ready (CED)
1381	Margin and position listing for Late Evening1
1382	Margin requirement summary for Late Evening1
1383	Data used for margin calculation for Late Evening1
1384	Product area total collateral summary for Late Evening1
1385	Product area collateral summary for Late Evening1
1386	Security bank summary for Late Evening1
1387	Clearing manager summary for Late Evening1
1388	Clearing manager product area margin re-quirement summary for Late Evening1
1389	Clearing manager total margin requirement summary for Late Evening1
1390	Position and position overview for Late Evening1
1391	Non-propagated Margin and position listing for Late Evening1
1392	Member product area collateral summary for Late Evening1
1561	RCAR worst for Late Evening1
1562	RCAR final scenario for Late Evening1
1563	RCAR top 10 for Late Evening1
1564	RCAR detailed for Late Evening1
1592	Data used for margin calc CO for Late Evening1

ing_id_s (Instrument Group Identity)	
Datatype	char[3]
Description	The ASCII representation of the instrument group.
initial_trr_min_value_u (Initial Trade Report, Minimum Order Value.)	
Datatype	INT64_T
Description	Not applicable.
instance_c (Instance, Number)	

Datatype	UINT8_T
Description	Defines one specific instance for multiple processes.
instance_next_c (Next Instance Number)	
Datatype	UINT8_T
Description	Next instance number for multiple processes.
instigant_c (Instigant)	
Datatype	UINT8_T
Description	Specifies whether a trade in a deal is the instigating party. A trade is considered instigant in the following cases: - Active/aggressive part in deal matched in electronic order book. - Source side in position transfer. - Source side in APS (average price system) deal.
Value Set	**value** / **description** 0 / Not instigating part 1 / Instigating part 2 / Instigating part unknown or N/A
instrument_group_c (Instrument Group)	
Datatype	UINT8_T
Description	A unique binary representation of the instrument group.
ins_id (INS_ID)	
Datatype	char[32]
Description	Intermediate field.
ins_id_s (Series, Identity)	
Datatype	char[32]
Description	Instrument Series name is ASCII.
interest_rate_i (Interest Rate)	
Datatype	INT32_T
Description	Defines the Interest Rate for the underlying. Decimal value stored with 6 implicit decimal, e.g. 11% is stored as 110000.
interest_rate_type_c (Interest Rate Type)	
Datatype	UINT8_T
Description	Defines the type of interest rate.
Value Set	**value** / **description** 0 / Simple 1 / Continuous
internal_full_depth_c (Full Depth, Internal)	

Datatype	UINT8_T
Description	Not applicable.
Value Set	**value** / **description** 2 / No

intraday_ind_c (Intra-day Indicator)	
Datatype	UINT8_T
Description	Defines if a covered call request is issued intra-day.
Value Set	**name** / **value** Not intra-day / 0 Intra-day / 1 Capital adjustment / 2

intra_day2_c (Intra Day2)		
Datatype	UINT8_T	
Description	Defines from which margin calculation result should be fetched.	
Value Set	**name**	**value** / **description**
	intra_day2_evening_data	0 / evening data Use results from evening margin calculations N/A for RQ2073
	intra_day2_intra_day_data	1 / intra day data Use results from latest available intra day margin calculations
	intra_day2_intra_call_data	2 / intra day margin call data Use results from latest available intra day margin call
	intra_day2_intra_calc_nbr	101 / Specific intra day margin data Use results from specific intra day calculation, as specifed in field Margin run number Applicable for RQ2, RQ3, RQ35, RQ36, RQ122, RQ2055, RQ2057, RQ2070, RQ2073, RQ2074 and RQ2078 only
	intra_day2_intra_call_nbr	102 / Specific intra day call data Use results from specific intra day margin call, as specifed in field Margin call number Applicable for RQ2, RQ3, RQ35, RQ36, RQ122, RQ2055, RQ2057, RQ2070,

name	value	description
		RQ2073, RQ2074 and RQ2078 only
intra_day2_intra_calc_nbr_non_prop	111	Specific non-propagated intra day call data Use results from specific non-propagated intra day calculation, as specifed in field Margin run number Applicable for RQ2, RQ3, RQ35, RQ36, RQ122, RQ2055, RQ2057, RQ2070 and RQ2073 only

intra_day4_c (Intra Day4)	
Datatype	UINT8_T
Description	Defines from which margin calculation result should be fetched.

Value Set

name	value	description
intra_day4_evening_data	0	Evening data Use results from evening margin calculations.
intra_day4_intra_day_data	1	intra day data Use results from latest available intra day margin calculations.
intra_day4_intra_call_data	2	intra day margin call data Use results from latest available intra day margin call
intra_day4_prel_evening_data	3	preliminary evening data Use results from calculation of preliminary evening prices.
intra_day4_intra_calc_nbr	101	Specific intra day margin data Use results from specific intra day calculation, as specifed in field Margin run number
intra_day4_intra_call_nbr	102	Specific intra day call data Use results from specific intra day margin call, as specifed in field Margin call number

int_id (INT_ID)	
Datatype	char[8]
Description	Intermediate field.

int_id_s (Instrument, Identity)	
Datatype	char[8]

Description	The ASCII representation of the instrument type.
investor_type_s (Investor Type)	
Datatype	char[4]
Description	Defines the investor type for the account.
inv_scheme_c (Investment Scheme)	
Datatype	CHAR
Description	Not applicable.
Value Set	**value** / **description** Blank / Not Applicable
isin_code_old_s (ISIN Code, Old Series)	
Datatype	char[12]
Description	This is the old ISIN Code if a new code was assigned to the series after a recapitalization.
isin_code_s (ISIN Code)	
Datatype	char[12]
Description	A code which uniquely identifies a specific securities issue (International Securities Identification Number). The ISIN shall consist of: a) A prefix, which is the alpha-2 country code b) The basic number, which is nine characters c) A check digit For more information about ISIN code, see the international standard ISO 3166.
iss_def_num_of_warnings_n (Number of Warnings, Default for ISS)	
Datatype	UINT16_T
Description	The default number of warnings if using the state as an Instrument Session State.
iss_def_warning_interval_n (Warning Interval, Default for ISS)	
Datatype	UINT16_T
Description	The default warning interval in seconds when using the state as an Instrument Session State.
is_exclusive_opening_sell_c (Exclusive Open Sell)	
Datatype	UINT8_T
Description	Defines if this is an Instrument Group where corresponding Instrument Series has Exclusive Open-Sell. If Exclusive Open-Sell, then it is only possible to do buy-open or sell-close.
Value Set	**value** / **description** 1 / Yes 2 / No
is_fractions_c (Fraction, Premium)	
Datatype	CHAR
Description	Is the premium internally represented as fractions?

Value Set	**name** / **value** Yes / Y No / N
is_trader_c (Trader)	
Datatype	UINT8_T
Description	Indicates if a certain user connected to the user type is a trader or not.
Value Set	**name** / **value** Trader / 1 Not trader / 2
items_block_n (Item, Block)	
Datatype	UINT16_T
Description	Number of items.
items_c (Item)	
Datatype	UINT8_T
Description	Number of items.
items_n (Items)	
Datatype	UINT16_T
Description	Number of items. This field used in a variable message counts the number of sub items provided in the variable message.
item_number_c (Item Number)	
Datatype	UINT8_T
Description	A common field holding a number.
item_type_c (Item Type)	
Datatype	UINT8_T
Description	Flags type of item in simulation query.

Value Set

name	value	description
item_type_market_data	1	Market data Market to use
item_type_bought_trade	2	Bought trade Item is a bought trade
item_type_sold_trade	3	Sold trade Item is a sold trade
item_type_payment	4	Payment Item is a payment
item_type_bought_delivery	5	Bought delivery

name	value	description
		Item is a bought delivery
item_type_sold_delivery	6	Sold delivery Item is a sold delivery

iter_accuracy_q (Iteration, Accuracy)	
Datatype	INT64_T
Description	Accuracy used when calculating implied volatility. Expressed with 8 implicit decimals.
iter_high_bound_i (Iteration, High Bound)	
Datatype	INT32_T
Description	High bound used when calculating implied volatility. Expressed in percent, no decimals.
iter_low_bound_i (Iteration, Low Bound)	
Datatype	INT32_T
Description	Low bound used when calculating implied volatility. Expressed in percent, no decimals.
iter_max_no_i (Iteration, Max)	
Datatype	INT32_T
Description	Max number of iterations when calculating implied volatility.
key_number_i (Key Number)	
Datatype	INT32_T
Description	The key number within one delivery number.
knock_variant_c (Knock Variant)	
Datatype	UINT8_T
Description	Knock in/out variant. A Knock In option is an option that comes alive, i.e. Knocks In, when a certain barrier is reached. If the barrier is never reached, the option will automatically expire worthless, as without reaching the barrier, it never exists. If the barrier is reached, the option knocks in and its final value will depend on where the spot rate settles in relation to the strike. They are therefore substantially cheaper than ordinary options. With the Knockout feature, if at any time up to and including the maturity, the Knockout level is reached the option will expire worthless.

Value Set

value	description
0	Not applicable
1	Down
2	Up

last_index_s (Index, Last Value)	
Datatype	char[8]
Description	Last index value for current day in ASCII format.
last_paid_i (Last, Paid)	

Datatype	INT32_T
Description	Last paid for the Instrument Series.
last_price_i (Price, Last)	
Datatype	INT32_T
Description	Defines the last traded price during the day.
last_qry_segment_c (Last, Query Segment)	
Datatype	UINT8_T
Description	Flags if this segment is the last query segment. 1 = Yes (Must be set to 1)
last_theo_c (Last Paid, Theoretical Mark)	
Datatype	UINT8_T
Description	Defines the origin of the price.
Value Set	**value** / **description** 0 / Missing 1 / Theoretically calculated 2 / From the order book 3 / Manually updated 4 / Artificial
leg_number_n (Leg Number)	
Datatype	UINT16_T
Description	The leg number of the central group.
level_type_i (Level Type)	
Datatype	INT32_T
Description	Position to be retrieved at what level?
Value Set	**value** / **description** 1 / Origin 2 / Margin
le_state_c (Type, Legal Event)	
Datatype	UINT8_T
Description	In principle, any object related to the clearing oriented part of the system, may be assigned a Legal Event State, or Le state for short. The field is not relevant to exchanges not using the clearing functionality; the value will in these cases always be 4, Active. Legal Event type:
Value Set	**name** / **value** / **description** None / 0 / None holding / 1 / Holding

name	value	description
		Object is holding and awaits countersign.
holding_indirectly	2	Holding Indirectly Object is awaiting a holding object.
pending	3	Pending Object is awaiting a later operation.
active	4	Active Object has been confirmed, if it was originally holding.
completed	5	Completed A pending object has been completed.
rejected	6	Rejected Object has been rejected.
business_completed	7	Business Completed Realtime events done. This value is logically between Active and Completed.
delivered	8	Delivered Object has been completed due to delivery.
rectified	9	Rectified
deleted	10	Deleted
pending_rectify	11	Pending Rectify
expired	12	Expired
pending_authorize	13	Pending Authorize

limit_premium_i (Premium, Limit)	
Datatype	INT32_T
Description	Defines the limit price.
linked_commodity_n (Linked Commodity Code)	
Datatype	UINT16_T
Description	If one or several underlying entries are linked together they are referenced to the real underlying by a pointer to the linked underlying code. If the underlyings are linked this code contains another Commodity Code distributed as another entry. 0 means that the underlyings are not linked.
list_name_s (Name, List)	
Datatype	char[40]

Description	List file name
long_adjustment_i (Long Adjustment)	
Datatype	INT32_T
Description	The number of contracts to net.
long_ins_id_s (Series Name, Long)	
Datatype	char[32]
Description	Defines an additional instrument information to an instrument series.
long_name (LONG_NAME)	
Datatype	char[32]
Description	Intermediate field.
lot_type_c (Lot, Type)	
Datatype	UINT8_T
Description	Specifies the lot type per block size.
Value Set	**value** / **description** 1 / Odd Lot 2 / Round Lot 3 / Block Lot 4 / All or None Lot Used to define which multiple of the order quantity that is allowed for All or None orders. In order transactions an All or None order is sent with block size = 0.
lower_limit_i (Premium/Price, Low Limit)	
Datatype	INT32_T
Description	The lower limit in the price interval.
low_index_s (Index, Lowest Value)	
Datatype	char[8]
Description	Lowest index value for current day in ASCII format.
low_price_i (Price, Low)	
Datatype	INT32_T
Description	Defines the lowest traded price during the day.
maintain_positions_c (Maintain Positions)	
Datatype	UINT8_T
Description	Maintain positions?
Value Set	**value** / **description** 1 / Keep Position 2 / No Keep Position

margin_class_s (Margin class)	
Datatype	char[3]
Description	Always set to blank
margin_deliv_c (Margin, Delivery)	
Datatype	UINT8_T
Description	Defines if delivery margin is used.
Value Set	**value** / **description** 1 / Forward Type 2 / None 3 / Forward Type, Reversed 4 / Compatibility 5 / Future Type 6 / Future Type, Reversed
margin_dividend_c (Margin, Dividend)	
Datatype	UINT8_T
Description	Specifies whether dividends should be used in margin calculations for equity based products.
Value Set	**value** / **description** 1 / Yes 2 / No
margin_one_writ_opt_q (Margining Requirements, One Written Option)	
Datatype	INT64_T
Description	Margin Requirements for one written option. The field contains an integer.
margin_payment_c (Payment Margin)	
Datatype	UINT8_T
Description	Defines type of payment margin used in margin calculations.
Value Set	**value** / **description** 1 / Fees 2 / Settlement 3 / Fees + Settlement 4 / None 5 / Compatibility 6 / Bought power
margin_req_u (Margin Requirements)	

Datatype	INT64_T
Description	The margining requirements needed as security.
marg_call_nbr_n (Margin call number)	
Datatype	UINT16_T
Description	Intra-day margin call number.
marg_item_type_c (Margin item type)	
Datatype	UINT8_T
Description	Always set to zero
marg_meth_inst_c (Margin method, for instrument class and instrument series)	
Datatype	UINT8_T
Description	Always set to zero
Value Set	**name** / **value**: Not set / 0
marg_param_id_s (Margin Parameter)	
Datatype	char[15]
Description	Defines name of margin parameter.
marg_price_i (Margin, Settlement Price)	
Datatype	INT32_T
Description	Defines the margin settlement price.
marg_run_nbr_n (Margin run number)	
Datatype	UINT16_T
Description	Intra-day margin calculation number.
marg_settl_days_i (Settlement Days)	
Datatype	INT32_T
Description	Number of settlement days for settlement margin (deliveries/payments) in margin calculations.
marg_theo_c (Margin, Settlement Price Theoretical Mark)	
Datatype	UINT8_T
Description	Defines the origin of the price.
Value Set	**value** / **description**: 0 / Missing; 1 / Theoretically calculated; 2 / From the order book; 3 / Manually updated; 4 / Artificial
market_c (Market Code)	

Datatype	UINT8_T
Description	Binary representation of the market. Unique together with COUNTRY_C.
market_currency_s (Currency, Market)	
Datatype	char[3]
Description	Native currency of the market (before currency conversion).
market_maker_c (Market Maker)	
Datatype	UINT8_T
Description	Is the account a market maker account?
Value Set	**value** / **description**: 1 – Yes; 2 – No
market_margin_q (Margin Requirements, Market)	
Datatype	INT64_T
Description	Margin requirement in native currency, before currency conversion.
market_orders_allowed_c (Market Orders, Allowed)	
Datatype	UINT8_T
Description	Are market orders allowed during the state:
Value Set	**name** / **value**: Yes – 1; No – 2
market_type_c (Market, Type)	
Datatype	UINT8_T
Description	Defines the type of market.
Value Set	**value** / **description**: 0 – Generic; 1 – Stock; 2 – Fixed Income; 3 – Currency; 4 – Power/Energy; 5 – Commodity; 6 – Payment; 7 – Index; 8 – General
market_value_q (Market Value)	

Datatype	INT64_T
Description	Calculated market value for the position. When used in F*-messages, the number of decimals equals decimals in premium price of series.
mar_id_s (Market, Identity)	
Datatype	char[5]
Description	The ASCII representation of the market.
master_clh_id_s (Master CLH, Identity)	
Datatype	char[12]
Description	The master clearinghouse for the exchange.
matching_price_type_c (Matching Price Type)	
Datatype	UINT8_T
Description	Different type of prices distributed as equilibrium price
Value Set	**name** / **value** / **description** matching_price_type_equilibrium / 1 / matching_price_type_equilibrium Normal indicative Equilibrium Price matching_price_type_fixed / 2 / matching_price_type_fixed Fixed price matching
match_group_nbr_u (Match group number, group inside an execution)	
Datatype	UINT32_T
Description	A sequential number of an execution sequence number.
match_item_nbr_u (Match Item Number)	
Datatype	UINT32_T
Description	Match item number inside a match group number.
maturity_c (Maturity)	
Datatype	UINT8_T
Description	Defines if this an Instrument Group where corresponding Instrument Series has an Expiration Date defined.
Value Set	**name** / **value** Yes / 1 No / 2
maximum_size_u (Block Size, Maximum Volume)	
Datatype	INT64_T
Description	The maximum volume allowed for the order per block size. Note! A value of 0 means no limit.
max_block_order_size_i (Order Size, Max Block)	

Datatype	INT32_T
Description	Max items in a Block Order Entry transaction.
max_block_price_size_i (Order Price, Max Block)	
Datatype	INT32_T
Description	Max items in a Two-sided Price Quotation Block transaction.
mbs_id_s (Minimum Bid Schedule)	
Datatype	CHAR[2]
Description	Not applicable.
member_circ_numb_s (Member, Circular Number)	
Datatype	char[4]
Description	Not applicable.
member_net_open_interest_q (Net Open interest, Member)	
Datatype	UINT64_T
Description	Defines the member net open interest.
message_header_s (Message, Header)	
Datatype	char[80]
Description	Header of message. Used to specify a short description of a message.
message_information_type_c (Message Information, Type)	
Datatype	UINT8_T
Description	Kind of message sent in announcement.

Value Set

name	value	description
MESSAGE_INFO_TYPE_COMPANY_ANNOUNCEMENT	1	Company Announcement
MESSAGE_INFO_TYPE_MARKET_MESSAGE	2	Market Message
MESSAGE_INFO_TYPE_STATIC_LINE	3	Static Line
MESSAGE_INFO_TYPE_NOTICE_RECEIVED	4	Notice Received

message_priority_c (Message, Priority)	
Datatype	UINT8_T
Description	Defines the priority of the message.

Value Set

name	value	description
MESSAGE_PRIORITY_LOW	1	Low priority
MESSAGE_PRIORITY_MEDIUM	2	Medium priority

name	value	description
MESSAGE_PRIORITY_HIGH	3	High priority
MESSAGE_PRIORITY_CRITICAL	4	Critical priority

message_source_s (Message, Source)	
Datatype	char[80]
Description	Source of the message, e.g. a linked exchange or the market control.
mic_code_s (MIC Code)	
Datatype	char[8]
Description	Specifies the MIC Code for the market.
minimum_size_n (Block Size, Minimum Volume)	
Datatype	UINT32_T
Description	The minimum volume required for the order per block size. Note! A value of 0 means no limit.
min_qty_increment_i (Minimum Quantity Increment)	
Datatype	INT32_T
Description	Not applicable.
min_show_vol_u (Order, Min Show Volume)	
Datatype	UINT32_T
Description	Minimum visible volume that must be specified in hidden orders.
modified_date_s (Date, Modified)	
Datatype	char[8]
Description	Date what the item was modified in ASCII. Format: YYYYMMDD.
modified_time_s (Time, Modified)	
Datatype	char[6]
Description	Defines what time the item was last changed. Format: HHMMSS.
modifier_c (Modifier)	
Datatype	UINT8_T
Description	Expiration date modifier. This value is set to zero when the instrument is new. The value is incremented by one each time the instrument is involved in an issue, split, etc. Note that the modifier value can be different for bid and ask options in the same Series.
mp_quantity_i (Quantity)	
Datatype	INT64_T
Description	Number of units (options, futures, forwards and so on) in an order related transaction.
multi_leg_price_type_c (Multi Leg Price Type)	
Datatype	UINT8_T

Description	Defines the price type used in the multi leg order.
Value Set	

name	value	description
multi_leg_price_type_none	0	Multi leg price is undefined
net_value	1	Net Value
reversed_net_value	2	Reversed Net Value
yield_difference	3	Yield Difference
individual_prices	4	Individual Prices
quantity_weighted_average	5	Quantity Weighted Average
multiplied	6	Multiplied

naked_margin_q (Margin Requirements, Naked)	
Datatype	INT64_T
Description	Margin requirement that should be present if there were no correlation effects available.
named_struct_n (Named Struct, Number)	
Datatype	UINT16_T
Description	In order to use variable messages, the structs that are potential members of such messages must have unique numbers. For detailed information refer to the "Named Structs Involved in VIMs" section.
name_s (Name)	
Datatype	char[32]
Description	The full ASCII representation.
name_short (NAME_SHORT)	
Datatype	char[10]
Description	intermediate field.
nationality_s (Nationality)	
Datatype	char[4]
Description	Defined the nationality for the account.
nbr_held_q (Held)	
Datatype	INT64_T
Description	Number of held (long) contracts
nbr_written_q (Written)	
Datatype	INT64_T
Description	Number of written (short) contracts
neg_time_adj_c (Negative Time Value Adjustment)	
Datatype	CHAR
Description	Flags if margin calculations should use adjustment for negative time value when evaluating held options.

Value Set	**value** / **description** Y / Use negative time adjustment N / Do NOT use negative time adjustment
netting_ratio_i (Netting Ratio)	
Datatype	INT32_T
Description	Defines the netting ratio between the classes.
net_open_interest_q (Net Open Interest)	
Datatype	UINT64_T
Description	Defines the net open interest.
new_commodity_n (Commodity Code, New)	
Datatype	UINT16_T
Description	Specified if the adjusted series are moved to a new underlying compared to the original series. If keeping the original underlying, the value is zero.
next_clearing_date_s (Clearing Date, Next)	
Datatype	char[8]
Description	Date in ASCII for clearing trade, format is YYYYMMDD.
next_planned_start_date_s (Planned Start Date, Next)	
Datatype	char[8]
Description	Defines planned start date for next planned state change. Distributed in UTC together with Planned Start Time, Next. Format: YYYYMMDD. If specified it is a warning and defines the next planned state. If not specified it is a state change.
next_planned_start_time_s (Planned Start Time, Next)	
Datatype	char[6]
Description	Defines planned start time for next planned state change. Distributed in UTC together with Planned Start Date, Next. Format: HHMMSS. If specified it is a warning and defines the next planned state. If not specified it is a state change.
nominal_value_q (Nominal Value)	
Datatype	INT64_T
Description	Nominal value for the underlying.
non_traded_ref_c (Non Traded Reference)	
Datatype	UINT8_T
Description	Not applicable.
Value Set	**value** / **description** 2 / No

normal_clearing_days_n (Normal Clearing Days)	
Datatype	UINT16_T
Description	This field describes the normal week days which is open for clearing. The field is a bit map, where each bit corresponds to a day in the week. If the bit is set to 1 the day is open, otherwise it is closed. The lowest bit is Monday, next Tuesday and so on.
normal_settl_days_n (Normal Settlement Days)	
Datatype	UINT16_T
Description	This field describes the normal week days which is open for settlement. The field is a bit map, where each bit corresponds to a day in the week. If the bit is set to 1 the day is open, otherwise it is closed. The lowest bit is Monday, next Tuesday and so on.
normal_trading_days_n (Normal Trading Days)	
Datatype	UINT16_T
Description	This field describes the normal week days which is open for trading. The field is a bit map, where each bit corresponds to a day in the week. If the bit is set to 1 the day is open, otherwise it is closed. The lowest bit is Monday, next Tuesday and so on.
no_of_legs_n (Legs, Number Of)	
Datatype	UINT16_T
Description	Number of legs in the combination.
no_of_orders_u (Orders, Number of)	
Datatype	UINT32_T
Description	Number of orders for one price level.
number_of_deals_u (Deals, Number)	
Datatype	UINT32_T
Description	Number of deals executed.
number_short (NUMBER_SHORT)	
Datatype	UINT16_T
Description	Intermediate field.
ob_changes_avail_c (Order Book Changes Available)	
Datatype	UINT8_T
Description	Order book changes available during the state.
Value Set	**value** / **description** 1 / Yes 2 / No
ob_command_c (Order-Book Command)	
Datatype	UINT8_T
Description	The type of change in the Order Book. Order Book command:

Value Set	<table><tr><th>name</th><th>value</th><th>description</th></tr><tr><td>ob_command_add</td><td>0</td><td>Order-Book Command Add</td></tr><tr><td>ob_command_delete</td><td>1</td><td>Order-Book Command Delete</td></tr><tr><td>ob_command_change</td><td>2</td><td>Order-Book Command Change</td></tr></table>
ob_position_u (Order Book Position)	
Datatype	UINT32_T
Description	Defines the priority or ranking position in the Order Book (l = highest priority).
odd_lot_allwd_c (Odd Lot, Allowed)	
Datatype	UINT8_T
Description	Is odd lot orders allowed during the state:
Value Set	<table><tr><th>value</th><th>description</th></tr><tr><td>1</td><td>Yes</td></tr><tr><td>2</td><td>No</td></tr></table>
offset_days_n (Offset Days for Settlement Margin)	
Datatype	UINT16_T
Description	Number of days used when determining the offset from last day when settlement margin is included in margin calculations.
old_trade_c (Old Trade Indicator)	
Datatype	UINT8_T
Description	Indicates whether the trade emanates from a deal cleared prior to the current clearing date.
Value Set	<table><tr><th>value</th><th>description</th></tr><tr><td>1</td><td>Yes</td></tr><tr><td>2</td><td>No Given up trade cleared today</td></tr></table>
omex_version_s (OMEX Version)	
Datatype	char[16]
Description	This is the current Genium INET version running on the system.
omxlen (OMXLEN)	
Datatype	char[8]
Description	intermediate field.
only_this_series_c (Series, Only this)	
Datatype	UINT8_T
Description	Only one specific series is requested.

Value Set	**value** / **description** 0 / No 1 / Yes
on_off_c (On or Off)	
Datatype	UINT8_T
Description	Status field for Suspend, Resume. Resume=On, Suspend=Off
Value Set	**value** / **description** 1 / On, keep orders 2 / Off, remove orders 3 / On, remove orders 4 / Off, keep orders
opening_price_i (Price, First)	
Datatype	INT32_T
Description	Defines the first traded price for the day.
open_balance_u (Open Interest)	
Datatype	INT64_T
Description	The number of outstanding contracts (not updated during the day).
open_close_c (Open or Closed)	
Datatype	UINT8_T
Description	Defines the position update for the account. None if positions not maintained or not applicable for instrument.
Value Set	**value** / **description** 0 / None No position update 1 / Open 2 / Closed
open_close_req_c (Open Close Request)	
Datatype	UINT8_T
Description	Describes how the requested position account should be updated:
Value Set	**name** / **value** / **description** OPEN_CLOSE_REQ_DEFAULT / 0 / Default for the account OPEN_CLOSE_REQ_OPEN / 1 / Open OPEN_CLOSE_REQ_CLOSE / 2 / Close/net

name	value	description
OPEN_CLOSE_REQ_MND_CLOSE	3	Mandatory close
OPEN_CLOSE_REQ_RE-SET	4	Set to default to the account (valid only for alter order)

opposing_deal_source_c (Opposing Deal Source)	
Datatype	UINT8_T
Description	Deal Source for the opposing order for this trade.
opposing_order_number_u (Order Number, Opposing)	
Datatype	QUAD_WORD
Description	Order number for the opposing order for this trade.
opra_indicator_c (OPRA Indicator)	
Datatype	CHAR
Description	Not applicable.
option_style_c (Option, Style)	
Datatype	UINT8_T
Description	Defines the style of the option.

Value Set

name	value	description
option_style_undefined	0	Not applicable
american	1	American
european	2	European
asian	3	Asian
bermudan	4	Bermudan
knock_in	5	Knock-in
knock_out	6	Knock-out
binary	7	Binary
ratchet	8	Ratchet

option_type_c (Option, Type)	
Datatype	UINT8_T
Description	Defines the type of the option.

Value Set

name	value	description
option_type_undefined	0	Not applicable
option_type_call	1	Call
option_type_put	2	Put

option_variant_c (Option, Variant)

Datatype	UINT8_T
Description	Defines the option variant.
Value Set	**value** / **description** 0 / Not applicable 1 / Normal 2 / Cap 3 / Floor
opt_min_ord_val_i (Optional minimum order value)	
Datatype	INT32_T
Description	Optional minimum order value. The value is always expressed in the primary currency unit. The value is defined as quantity*price*price quotation factor.
opt_min_trade_val_i (Optional minimum trade value)	
Datatype	INT32_T
Description	Optional minimum trade value. The value is always expressed in the primary currency unit. The value is defined as quantity*price*price quotation factor.
opt_price_base_1_c (Option Price Base 1)	
Datatype	CHAR
Description	Defines what to base valuation of options upon. Base 2 is a fallback method if Base 1 fails.
Value Set	**name** / **value** / **description** Last Price of Underlying spot / S / Last Price of Underlying spot Future Forward Fix / X / Future Forward Fix Future Forward Settlement Price / F / Future Forward Settlement Price Syntetic Future / Y / Syntetic Future Underlying Margin Settl Special / M / Underlying Margin Settl Special Forward Fix / I / Forward Fix Forward Settlement Price / R / Forward Settlement Price Future Fix / T / Future Fix Future Settlement Price / E / Future Settlement Price
opt_price_base_2_c (Option Price Base 2)	
Datatype	CHAR
Description	Defines what to base valuation of options upon. Base 2 is a fallback method if Base 1 fails.

Value Set	

name	value	description
Last Price of Underlying spot	S	Last Price of Underlying spot
Future Forward Fix	X	Future Forward Fix
Future Forward Settlement Price	F	Future Forward Settlement Price
Syntetic Future	Y	Syntetic Future
Underlying Margin Settl Special	M	Underlying Margin Settl Special
Forward Fix	I	Forward Fix
Forward Settlement Price	R	Forward Settlement Price
Future Fix	T	Future Fix
Future Settlement Price	E	Future Settlement Price

opt_price_model_c (Option Price Model)	
Datatype	UINT8_T
Description	Defines the option price model used for the series.
Value Set	

name	value	description
Non Option	0	Non-option
Standard Black And Scholes	1	Standard Black and Scholes
Standard Black And Scholes Dividend Yield	2	Black and Scholes extended by dividend yield
Black 76 Index Options	3	Black 76 for index options
Black 76 Interest Rate Options	4	Black 76 for interest rates
Black 76 Other Options	5	Black 76 for other options than index or interest rates
Binomial Without Dividends	6	Binomial without dividends
Binomial With Dividends	7	Binomial with one or several dividends.

opt_ulg_price_src_c (Option Underlying Price Source)	
Datatype	UINT8_T
Description	This field tells what type of underlying that is used as source of the underlying price.
Value Set	

name	value	description
Non Option	0	Non-option
Underlying	1	Underlying
Upper Level Series	2	Upper level series
Future Or Forward	3	Corresponding future/forward Comment: This is for instance used for OMX options.

name	value	description
		This is the future/forward with the same country, market, underlying and expiration as the option.
Synthetic Future	4	Synthetic future

op_if_buy_c (Operation if Buy)	
Datatype	CHAR
Description	Specifies whether to buy or sell the Series when buying the combination.
Value Set	value: B — description: Buy value: S — description: Sell

op_if_sell_c (Operation if Sell)	
Datatype	CHAR
Description	Specifies whether to buy or sell the Series when selling the combination.
Value Set	value: B — description: Buy value: S — description: Sell

orderbook_id_i (Order book id)	
Datatype	INT32_T
Description	Identification of an order book.

order_category_c (Order Category)	
Datatype	UINT8_T
Description	Defines the order category.
Value Set	name: Undefined — value: 0 name: Firm Order/Quote — value: 1 name: Indicative Order/Quote — value: 2

order_index_u (Order Index)	
Datatype	UINT32_T
Description	The order index is a counter that is used as search criteria for querying the next segment of information.

order_number_ask_u (Order Number, Ask)	
Datatype	QUAD_WORD
Description	A unique identity for each order transaction for the ask part.

order_number_bid_u (Order Number, Bid)	
Datatype	QUAD_WORD
Description	A unique identity for each order transaction for the bid part.
order_number_u (Order Number)	
Datatype	QUAD_WORD
Description	A unique identity for each order transaction.
order_state_u (Order State)	
Datatype	UINT32_T
Description	Defines the state of the order.

Value Set

name	value
Preliminary	1
Accepted	2
Rejected	3
Preliminary_enter	4
Preliminary_alter	5
Preliminary_delete	6
Order_altered	7
Order_deleted	8
Deleted	9
Order_active	10
Order_inactive	11

order_type_c (Order Type)	
Datatype	UINT8_T
Description	Order type declares characteristics about an order in terms of a bit map where each bit is assigned a specific property. Trading rules specific to the exchange defines which bit combinations are allowed. For example, a trading rule may state that a best order must also be a limit order, summing up to the value 17 of the Order Type field.

Value Set

name	value	description
ORDER_TYPE_LIMIT	1	Limit order
ORDER_TYPE_MARKET	2	Market order
ORDER_TYPE_MTL	3	Market to Limit This is a market order that is converted to a limit order when a price has been assigned.
ORDER_TYPE_PASSIVE	4	Passive order
ORDER_TYPE_ONLY_BEST	8	Only best order

name	value	description
ORDER_TYPE_BEST_ORDER	16	Best order
ORDER_TYPE_ODD_LOT	32	Odd lot order
ORDER_TYPE_IMBALANCE	64	Imbalance order
ORDER_TYPE_OVERRIDE_MMP	128	Override quote

org_number_s (Organization number)	
Datatype	char[16]
Description	Organization number for owner of account.
original_date_s (Original Date)	
Datatype	char[8]
Description	As of date for delivery. Format is YYYYMMDD
original_delivery_number_i (Original, Delivery Number)	
Datatype	INT32_T
Description	When not zero, it is used to point out another delivery together with fields Series and Original, Key Number.
original_key_number_i (Original, Key Number)	
Datatype	INT32_T
Description	When not zero, it is used to point out another delivery together with fields Series and Original, Delivery Number.
originator_type_c (Originator Type)	
Datatype	UINT8_T
Description	Defines the type of originator for the delivery.

Value Set

value	description
1	Normal
2	Reversing This delivery is created from a reversing trade

origin_c (Origin, Account Type)	
Datatype	CHAR
Description	Defines how trading activites on accounts of the account type are to be classified.

Value Set

name	value
House	H
Client	C

orig_clearing_date_s (Clearing Date, Original)	
Datatype	char[8]

Description	The date the deal was originally cleared. Date in ASCII, format is YYYYMMDD
orig_ext_trade_number_u (Trade Number, Original External)	
Datatype	UINT32_T
Description	Original trade number assigned by external system.
orig_market_value_q (Original market value)	
Datatype	INT64_T
Description	Calculated market value for the position.
orig_shown_quantity_i (Shown Quantity, Original)	
Datatype	INT64_T
Description	Original shown number of units (options, futures, forwards and so on) in an order related transaction.
orig_total_volume_i (Total Volume, Original)	
Datatype	INT64_T
Description	Original total number of units (options, futures, forwards and so on) in an order related transaction.
orig_trade_number_i (Trade Number, Original)	
Datatype	INT32_T
Description	For an overtaking trade, this field references the original trade.
orig_trade_type_c (Trade Type, Original)	
Datatype	UINT8_T
Description	Defines the original trade type, for further description see Trade Type.
other_currency_s (Currency, Other)	
Datatype	char[3]
Description	The other leg of the exchange rate.
output_level_c (Output Level)	
Datatype	UINT8_T
Description	Flags for desired output in margin simulation.
Value Set	

name	value	description
Only sum margin requirements	1	Only sum margin requirements Only sum margin requirements
Level 1 and margin results per series	2	Level 1 and margin results per series Level 1 and margin results per series
Level 2 prices and valuation interval per series and volatilities for options	3	Level 2 prices and valuation interval per series and volatilities for options

name	value	description
		Level 2 prices and valuation interval per series and volatilities for options

outside_info_spread_c (Outside Information Spread)	
Datatype	UINT8_T
Description	Is the trade report outside the spread or not?
Value Set	

name	value
Inside	0
Outside	1

own_inventory_c (Own Inventory)	
Datatype	UINT8_T
Description	Is the account an own inventory account?
Value Set	

value	description
1	Yes
2	No

passthrough_s (Passthrough Information)	
Datatype	char[32]
Description	A reserved field for information sent from external sources to be passed through the clearing system without any processing or validation.

pay_margin_q (Payment Margin)	
Datatype	INT64_T
Description	Defines the payment margin.

physical_delivery_c (Physical Delivery)	
Datatype	UINT8_T
Description	Defines if this an Instrument Group where corresponding Instrument Series are physically delivered.
Value Set	

value	description
1	Yes
2	No

points_of_movement_i (Points, Movement)	
Datatype	INT32_T
Description	The change between two index values expressed as number of points. The value includes implicit decimals with the number as of the index itself.

points_reg_i (Points, Number of)

Datatype	INT32_T
Description	Number of points in valuation interval.
positions_allowed_c (Positions, Allowed)	
Datatype	UINT8_T
Description	Is it allowed to hold positions on the account?
Value Set	**name** / **value** Yes / 1 No / 2
post_trade_proc_c (Post Trade processed)	
Datatype	UINT8_T
Description	Specifies if instrument series connected to the instrument type is processed in the Clearing System.
Value Set	**name** / **value** Yes / 1 No / 2
pos_handling_c (Position handling)	
Datatype	UINT8_T
Value Set	**name** / **value** No position keeping / 1 Single session position keeping / 2 Invariant dual session position keeping / 3 Sequential dual session position keeping / 4
pos_sim_c (Positions, Simulated)	
Datatype	UINT8_T
Description	Defines the positions to be used in margin simulation.
Value Set	**name** / **value** / **description** Only use trades specified in the query / 0 / Only use trades specified in the query Only use trades specified in the query Use real time position / 1 / Use real time position Use real-time position for the account specified in the Account field, together with trades specified in query. Get sum margin requirement / 2 / Get sum margin requirement

name	value	description
		Get sum margin requirement for all indirect pledging accounts if the participant specified in the account field.
Use real time positions for account	3	Use real time positions for account Use real-time position for the account specified in the Account field, together with trades specified in the query. A frozen copy of the real time position is also saved on the back end for use in subsequent simulations. Note: One single user can only save one position at a time.
Use positions previously frozen	4	Use positions previously frozen Use position previously frozen for the user sending the query together with trades specified in the query. Note: The account field in the query is not used in this case.
Use start of day position	5	Use start of day position Use start of day position for the account specified in the Account field, totgether with trades specified in the query

pow_offset_days_i (Value Offset, Known)	
Datatype	INT32_T
Description	Defines offset days for known values in margin calculations.

pqf_modifier_c (Modifier, Price Quotation Factor)	
Datatype	UINT8_T
Description	The modifier is used to recalculate the item after an underlying adjustment. The field is stored with 3 implicit decimals.

Value Set

value	description
1	Modifier is added to the item
2	Modifier is subtracted from the item
3	Modifier is multiplied with the item
4	The item is divided by the modifier factor

pqf_mod_factor_i (Modifier Factor, Price Quotation Factor)	
Datatype	INT32_T

<table>
<tr><td>Description</td><td>The modifier is used to recalculate the item after an underlying adjustment. The field is stored with 7 implicit decimals</td></tr>
<tr><td colspan="2">premium_i (Premium)</td></tr>
<tr><td>Datatype</td><td>INT32_T</td></tr>
<tr><td>Description</td><td>The price of one Series (excluding transaction cost) a user is prepared to pay - or wants to receive. This is always an integer.

In the distribution of data from the exchange these fields may hold a value where bit 31 (highest bit) is set while all other bits are cleared. This indicates that there is no premium available. This differs from the value of zero (all bits cleared) indicating a premium prize of zero.</td></tr>
<tr><td>Value Set</td><td><table><tr><th>value</th><th>description</th></tr><tr><td>>0</td><td>Price</td></tr><tr><td>= 0</td><td>Market price</td></tr><tr><td><0</td><td>Combo price (may be neg).</td></tr></table></td></tr>
<tr><td colspan="2">premium_levels_c (Premium Levels)</td></tr>
<tr><td>Datatype</td><td>UINT8_T</td></tr>
<tr><td>Description</td><td>Defines the number of levels of premiums distributed within the associated premium list. Exchange regulations could set the level to a lower value then both the actual list size and the actual depth in the market.</td></tr>
<tr><td colspan="2">prev_clearing_date_s (Clearing Date, Previous)</td></tr>
<tr><td>Datatype</td><td>char[8]</td></tr>
<tr><td>Description</td><td>Date in ASCII for clearing trade, format is YYYYMMDD.</td></tr>
<tr><td colspan="2">price (PRICE)</td></tr>
<tr><td>Datatype</td><td>INT32_T</td></tr>
<tr><td>Description</td><td>Intermediate field.</td></tr>
<tr><td colspan="2">price_carrier_code_n (Price carrier)</td></tr>
<tr><td>Datatype</td><td>UINT16_T</td></tr>
<tr><td>Description</td><td>Underlying code of price carrier.

0 means no price carrier.</td></tr>
<tr><td colspan="2">price_currency_s (Currency, Price)</td></tr>
<tr><td>Datatype</td><td>char[3]</td></tr>
<tr><td>Description</td><td>The currency in which an exchange rate is defined.</td></tr>
<tr><td colspan="2">price_format_c (Premium/Price Format)</td></tr>
<tr><td>Datatype</td><td>UINT8_T</td></tr>
<tr><td>Description</td><td>Not applicable.</td></tr>
<tr><td colspan="2">price_move_guard_c (Price Movement Guard)</td></tr>
<tr><td>Datatype</td><td>UINT8_T</td></tr>
<tr><td>Description</td><td>Defines if price movements for futures should be guarded in intra day margin calculations.</td></tr>
</table>

Value Set	<table><tr><th>value</th><th>description</th></tr><tr><td>1</td><td>Yes</td></tr><tr><td>2</td><td>No</td></tr></table>
price_param_id_s (Price Parameter)	
Datatype	char[15]
Description	Name of price parameter.
price_quotation_required_c (Price, Quotation Required)	
Datatype	UINT8_T
Description	Price Quotation supervision enabled during the state.
Value Set	<table><tr><th>value</th><th>description</th></tr><tr><td>1</td><td>Yes</td></tr><tr><td>2</td><td>No</td></tr></table>
price_quot_factor_i (Price, Quotation Factor)	
Datatype	INT32_T
Description	Defines the price quotation factor used to calculate the trade price from the order.
price_sim_c (Prices Simulated)	
Datatype	UINT8_T
Description	Flags which prices that should be used in margin simulation.
Value Set	<table><tr><th>name</th><th>value</th><th>description</th></tr><tr><td>Use real time prices</td><td>0</td><td>Use real time prices Use real time prices.</td></tr><tr><td>Use real time prices some ignored</td><td>1</td><td>Use real time prices special Use real-time prices. With this value, the value in the fields "Added trades Simulated", "Series expiting today simulated" and "Futures Profit/Loss simulated" will be ignored. This is for backward compatibility with earlier versions of the query.</td></tr><tr><td>Use real time prices frozen</td><td>2</td><td>Use real time prices frozen Use real time pries. A frozen copy of the real time prices is also saved in the server for use in subsequent simulaitons. Note: One single user can only save on set of prices at a time.</td></tr></table>

name	value	description
Use prices previously frozen	3	Use prices previously frozen Use prices previously frozen for the user sending the query.
Use start of day prices	4	Use start of day prices Use start of day prices.
Use official end of day prices	5	Use official end of day prices Use official day end prices.

price_unit_c (Price Unit, Underlying)	
Datatype	UINT8_T
Description	The price unit for the underlying can be one of the following:

Value Set

value	description
1	Price
2	Yield
3	Points
4	Yield Diff
5	IMM Index
6	Basis Points
7	Inverted Yield
8	Percentage of Nominal
9	Dirty Price

price_unit_premium_c (Price Unit, Premium)	
Datatype	UINT8_T
Description	The premium unit that describes the price unit in the order.

Value Set

value	description
1	Price
2	Yield
3	Points
4	Yield Diff
5	IMM Index
6	Basis Points
7	Inverted Yield
8	Percentage of Nominal
9	Dirty Price

<table>
<tr><td colspan="2">price_unit_strike_c (Price Unit, Strike)</td></tr>
<tr><td>Datatype</td><td>UINT8_T</td></tr>
<tr><td>Description</td><td>The strike price unit for the class can be one of the following:</td></tr>
<tr><td>Value Set</td><td><table>
<tr><th>value</th><th>description</th></tr>
<tr><td>1</td><td>Price</td></tr>
<tr><td>2</td><td>Yield</td></tr>
<tr><td>3</td><td>Points</td></tr>
<tr><td>4</td><td>Yield Diff</td></tr>
<tr><td>5</td><td>IMM Index</td></tr>
<tr><td>6</td><td>Basis Points</td></tr>
<tr><td>7</td><td>Inverted Yield</td></tr>
</table></td></tr>
<tr><td colspan="2">pri_not_s (Notation, Primary)</td></tr>
<tr><td>Datatype</td><td>char[5]</td></tr>
<tr><td>Description</td><td>The currency primary notation, e.g. $.</td></tr>
<tr><td colspan="2">pri_unit_s (Unit, Primary)</td></tr>
<tr><td>Datatype</td><td>char[15]</td></tr>
<tr><td>Description</td><td>Primary Unit.
The currency unit, e.g. DOLLAR, CENT.</td></tr>
<tr><td colspan="2">prod_area_c (Product Area, RIVA)</td></tr>
<tr><td>Datatype</td><td>UINT8_T</td></tr>
<tr><td>Description</td><td>Define the RIVA product area.</td></tr>
<tr><td colspan="2">prod_area_text_s (Product Area Text, RIVA)</td></tr>
<tr><td>Datatype</td><td>char[10]</td></tr>
<tr><td>Description</td><td>Description of a product area in ASCII.</td></tr>
<tr><td colspan="2">prod_grp_offset_i (Product Group Offset)</td></tr>
<tr><td>Datatype</td><td>INT32_T</td></tr>
<tr><td>Description</td><td>Product group offset used in margining by the cardinal method. Expressed in percent, 4 implicit decimals.</td></tr>
<tr><td colspan="2">program_trader_c (Program Trader)</td></tr>
<tr><td>Datatype</td><td>UINT8_T</td></tr>
<tr><td>Description</td><td>Defines if the User is a program trader ot not:</td></tr>
<tr><td>Value Set</td><td><table>
<tr><th>value</th><th>description</th></tr>
<tr><td>1</td><td>Yes</td></tr>
<tr><td>2</td><td>No</td></tr>
</table></td></tr>
<tr><td colspan="2">propagation_u (Propagation)</td></tr>
</table>

Datatype	UINT32_T
Description	States from what event the propagation is generated, e.g. Trade.
Value Set	

name	value	description
Propagate_none	0	
Propagate_trade	1	
Propagate_net_position	2	
Propagate_gross_position	3	
Propagate_delivery_flow	4	
Propagate_accrued	5	

prop_type_c (Type of Propagation)	
Datatype	UINT8_T
Description	Defines the type of account propagation.
Value Set	

value	description
1	Trade
2	Position
3	Margin
4	Settlement
5	Origin

public_deal_information_c (Public Deal Information)	
Datatype	UINT8_T
Description	Specifies how the post trade public deal information is distributed.
Value Set	

name	value
No information	0
Without identity	1
With identity	2

pub_inf_id_n (Public Order Info)	
Datatype	UINT16_T
Description	Specifies how order information is distributed
Value Set	

name	value	description
Without identity	1	The order information is distributed with broadcast BO2 and the answer of query MQ7 is without identity.
With identity	2	The order information is distributed with broadcast BO1

name	value	description
		and the answer of query MQ7 is with identity.
Query information without identity	3	The answer of MQ7 is without identity. No BO2 generated.
Query information with identity	4	The answer of MQ7 is with identity. No BO1 generated.
No information	5	No MQ7 generated, No BO1 or BO2 generated.

pur_id_s (Parameter Block)	
Datatype	char[15]
Description	Named block of margin parameters.
qry_segment_number_n (Segment Number, Query)	
Datatype	UINT16_T
Description	Defines the segment number in the query.
qty_closed_out_q (Quantity, Closed out)	
Datatype	INT64_T
Description	Quantity closed out on position
quantity_cover_u (Quantity Cover)	
Datatype	UINT32_T
Description	Defines the number of underlying shares used as cover for a short position.
quantity_i (Quantity)	
Datatype	INT64_T
Description	Defines the quantity.
quantity_protection_q (Quantity protection)	
Datatype	INT64_T
Description	Specifies the limit of the total traded contracts per underlying within the exposure time interval when market maker protection is triggered. When this value is exceeded the system automatically removes the quotes for the instruments connected to the underlying. A value of 0 means that no quantity protection exists.
quantity_request_i (Quantity, request)	
Datatype	INT32_T
quantity_tot_cover_u (Quantity, total cover)	
Datatype	UINT32_T
query_on_date_c (Query on Date)	
Datatype	UINT8_T
Description	Defines whether date is part of the search criteria.

Value Set	**value** / **description** 0 / No 1 / Yes
quote_action_c (Quote Action)	
Datatype	UINT8_T
Value Set	**name** / **value** None / 1 Update / 2 Delete / 3
rank_class_i (Risk Ranking Class)	
Datatype	INT32_T
Description	The risk ranking class of an account or member.
rate_determ_days_n (Rate Determination Days)	
Datatype	UINT16_T
Description	Specifies number of rate determination days.
rate_high_i (Rate, High)	
Datatype	INT32_T
Description	Defines the high exchange rate used when currency risk is applied.
rate_low_i (Rate, Low)	
Datatype	INT32_T
Description	Defines the low exchange rate used when currency risk is applied.
rate_nominal_i (Rate, Nominal)	
Datatype	INT32_T
Description	Defines the nominal exchange rate.
ratio_n (Ratio)	
Datatype	UINT16_T
Description	Relative numbers of contracts between the combo legs.
read_access_c (Read Access)	
Datatype	UINT8_T
Description	Defines what type of data the owner of the account can read.
Value Set	**value** / **description** 0 / None 1 / Position 2 / Trade

real_time_price_fut_c (Real Time Price, Futures/Forwards)	
Datatype	UINT8_T
Description	Specifies usage of real time prices in real time margin calculations for futures/forwards.
Value Set	**name** / **value** Directly / 0 Implied / 1 TMC style / 2 Spread difference spot month / 4
real_time_price_opt_c (Real Time Price, Options)	
Datatype	UINT8_T
Description	Specifies usage of real time prices in real time margin calculations for options.
Value Set	**name** / **value** Directly / 0 Implied / 1 TMC style / 2 Spread Difference / 3
real_time_price_use_c (Real Time Price Usage)	
Datatype	UINT8_T
Description	Specifies usage of real time prices in real time margin calculations. Note: This field is kept for compatility purposes only. Please use fields Real time price, Options or Real Time Price, Futures/Forwards instead.
Value Set	**value** / **description** 0 / Use always prices directly 1 / Use implied volatility for options 2 / Use implied volatility for options and implied rate for futures/forwards
rectify_trade_number_i (Rectify Trade Number)	
Datatype	INT32_T
Description	A number that together with series identifies a specific rectified trade.
redemption_value_i (Redemption Value)	
Datatype	INT32_T
Description	Redemption value equals the amount paid at the maturity. The redemption value will be equal to the nominal value except for securities with amortization or options. The redemption value is expressed in percentage of Nominal Value. The value is a decimal value stored with 6 decimals, e.g. 100% is stored as 1000000.
ref_price_i (Price, Reference)	

Datatype	INT32_T
Description	Reference price of the underlying/instrument series.
remaining_contract_size_i (Contract Size, Remaining)	
Datatype	INT32_T
Description	Defines the remaining contract size.
rem_quantity_i (Quantity, Remaining)	
Datatype	INT64_T
Description	Number of contracts, etc. Depending of instrument type. It reflects: Quantity still to be transferred from a transitory trade, for example, if a buy trade is created with quantity 25 on a transitory account, then rem_quantity_i will contain 25, as this quantity is still remaining to be moved to a position account. Quantity still to be exercised for trade with an instrument type that has trade exercise ability, for example if a trade is created with quantity 25 on a option series then rem_quantity_i will contain 25, as this quantity is still remaining to be exercised.
report_owner_s (Report owner)	
Datatype	char[12]
Description	Name of member or customer that is the owner of the report.
report_version_s (Report Version)	
Datatype	char[3]
Description	Zero padded sequence number of the report.
repo_type_c (Repo Type)	
Datatype	UINT8_T
Description	Defines the type of the REPO.
Value Set	**value** / **description** 0 / Not applicable 1 / GC 2 / GCF 3 / Special 4 / Security Lending 5 / IR Swap
reserved_11_s (Reserved)	
Datatype	char[11]
Description	Filler for alignment
reserved_1_c (Reserved)	
Datatype	CHAR
Description	Filler for alignment.
reserved_2_s (Reserved)	

Datatype	char[2]
Description	Filler for alignment.
reserved_8_s (Reserved)	
Datatype	char[8]
Description	Filler for alignment.
reserved_i (Reserved)	
Datatype	INT32_T
Description	Filler for alignment.
reserved_prop_c (Reserved Properties)	
Datatype	UINT8_T
Description	Generic bit mask flag dependant on the specific configuration or installation.
Value Set	**name** / **value** None / 0 Anonymized / 1
residual_i (Residual)	
Datatype	INT32_T
Description	Residual due to rounding in average price trade.
revised_open_balance_u (Revised Open Interest)	
Datatype	INT64_T
Description	Revised calculation of the number of outstanding contracts at end of the business day.
rho_i (Rate Of Change, Option Value)	
Datatype	INT32_T
Description	The rate of change in an options value, due to a change in the interest rate. Given with 4 decimals.
risk_array_p10_i (Risk array point 10)	
Datatype	INT32_T
Description	Risk array value in scenario point 10.
risk_array_p11_i (Risk array point 11)	
Datatype	INT32_T
Description	Risk array value in scenario point 11.
risk_array_p12_i (Risk array point 12)	
Datatype	INT32_T
Description	Risk array value in scenario point 12.
risk_array_p13_i (Risk array point 13)	
Datatype	INT32_T
Description	Risk array value in scenario point 13.

risk_array_p14_i (Risk array point 14)	
Datatype	INT32_T
Description	Risk array value in scenario point 14.
risk_array_p15_i (Risk array point 15)	
Datatype	INT32_T
Description	Risk array value in scenario point 15.
risk_array_p16_i (Risk array point 16)	
Datatype	INT32_T
Description	Risk array value in scenario point 16.
risk_array_p1_i (Risk array point 1)	
Datatype	INT32_T
Description	Risk array value in scenario point 1.
risk_array_p2_i (Risk array point 2)	
Datatype	INT32_T
Description	Risk array value in scenario point 2.
risk_array_p3_i (Risk array point 3)	
Datatype	INT32_T
Description	Risk array value in scenario point 3.
risk_array_p4_i (Risk array point 4)	
Datatype	INT32_T
Description	Risk array value in scenario point 4.
risk_array_p5_i (Risk array point 5)	
Datatype	INT32_T
Description	Risk array value in scenario point 5.
risk_array_p6_i (Risk array point 6)	
Datatype	INT32_T
Description	Risk array value in scenario point 6.
risk_array_p7_i (Risk array point 7)	
Datatype	INT32_T
Description	Risk array value in scenario point 7.
risk_array_p8_i (Risk array point 8)	
Datatype	INT32_T
Description	Risk array value in scenario point 8.
risk_array_p9_i (Risk array point 9)	
Datatype	INT32_T
Description	Risk array value in scenario point 9.
risk_currency_s (Currency, Risk)	

Datatype	char[3]
Description	Currency after currency conversion.
risk_cur_conv_c (Risk, Currency Conversion)	
Datatype	UINT8_T
Description	Condition for currency conversion for margin requirements.
Value Set	**value** / **description** 0 / Default 1 / Only Positive Only convert margin gains to risk currency 2 / Always Always convert margin to risk currency 3 / None Do not convert margin to risk currency
risk_free_rate_i (Interest, Risk Free)	
Datatype	INT32_T
Description	Risk free interest rate, expressed in percent. The value is stored with 4 implicit decimals, e.g. 11% is stored as 110000.
risk_margin_net_c (Risk, Margin Net)	
Datatype	UINT8_T
Description	Net margin requirements between markets.
Value Set	**value** / **description** 1 / Do not Net 2 / Net
risk_margin_q (Margining Requirements, Risk)	
Datatype	INT64_T
Description	Margin requirement after currency conversion.
rnt_id_n (Ranking Type)	
Datatype	UINT16_T
Description	This identifies how the instrument is ranked.
Value Set	**value** / **description** 1 / Rule 1 1. Price 2. Time 2 / Rule 2 1. Inverted Price 2. Time

value	description
3	Rule 3 1. Price 2. Traders before MM 3. Time
4	Rule 4 1. Inverted Price 2. Traders before MM 3. Time
5	Rule 5 1. Price 2. MM before Traders 3. Time
6	Rule 6 1. Inverted Price 2. MM before Traders 3. Time
7	Rule 7 1. Price 2. Baits before Normal Orders 3. Time
8	Rule 8 1. Inverted Price 2. Baits before Normal Orders 3. Time
11	Rule 11 1. Price 2. Own Orders 3. Time
12	Rule 12 1. Inverted Price 2. Own Orders 3. Time

seconds_to_state_change_n (State Change, Seconds)	
Datatype	UINT16_T
Description	This identifies how many seconds that are left until a change of state. If the value is larger than zero it is a warning. If the value is zero it means that it is the actual state change. Value = 0 State Change Value larger than 0 Warning

sector_code_s (Sector Code)	
Datatype	char[4]
Description	The sector code that the underlying is connected to.
sec_not_s (Notation, Secondary)	
Datatype	char[5]
Description	The currency secondary notation, e.g. C.
sec_rel_primary_n (Relation to Primary, Secondary)	
Datatype	UINT16_T
Description	Relation between the first and the secondary unit. E.g.If the primary unit is DOLLAR and the secondary unit is CENT, the relation will be 100.
sec_unit_s (Unit, Secondary)	
Datatype	char[15]
Description	Secondary Unit. The currency unit, e.g. DOLLAR, CENT.
segment_number_n (Segment Number)	
Datatype	UINT16_T
Description	Each part of a big data transfer has a segment number. In a query the segment to fetch is specified and the received answer contains the same segment number. The last answer message is indicated by segment number 0.
sell_price_i (Ask Price)	
Datatype	INT32_T
Description	the sell price for a quote
sell_quantity_u (Sell Quantity)	
Datatype	INT64_T
Description	Number of units (options, futures, forwards and so on) in an double price order related transaction.
send_or_receive_c (Send or Receive)	
Datatype	UINT8_T
Description	Indicates if a commission rule should be used while sending or receiving a give-up.
Value Set	**value** / **description** 0 / None 1 / Send 2 / Receive
sent_date_s (Date, Sent)	
Datatype	char[8]
Description	Defines the sent date. Format: YYYYMMDD.
sent_time_s (Time, Sent)	

Datatype	char[6]
Description	Defines the sent time. Format: HHMMSS
sequence (SEQUENCE)	
Datatype	INT32_T
Description	intermediate field.
sequence_first_i (Number, First Sequential)	
Datatype	INT32_T
Description	First number in a sequence.
sequence_last_i (Number, Last Sequential)	
Datatype	INT32_T
Description	Last number in a sequence.
sequence_nbr_u (Sequence Number)	
Datatype	UINT32_T
Description	Defines a sequence number.
sequence_number_i (Sequence Number)	
Datatype	INT32_T
Description	Define a sequence number.
sequence_number_u (Sequence Number)	
Datatype	UINT32_T
Description	Define a sequence number.
seq_nbr_1_u (Sequence Number)	
Datatype	UINT32_T
Description	Defines a sequence number.
seq_nbr_2_u (Sequence Number)	
Datatype	UINT32_T
Description	Defines a sequence number.
seq_num_srm_n (Sequence number for SRM)	
Datatype	UINT16_T
Description	An unique sequence number used by SRM
series_exp_today_sim_c (Series expiring today simulated)	
Datatype	UINT8_T
Description	Defines how series expiring today should be handled in margin simulation.
Value Set	**name** / **value** / **description** Not included / 0 / Not included Not included. Evening mode / 1 / Evening mode Evening mode..

name	value	description
		This means included only if also included in EndOfDay calculations of today.
Intra day mode	2	Intra day mode Intra mode, price moves of tomorrow. This means included, current prices will remain until EndOf-Day.
Intra mode price moves of today	3	Intra mode price moves of today Intra mode, price moves of today. This means included, current prices move in the same way as in normal margin calcula-tions.

series_id_s (Series, Identity)	
Datatype	char[32]
Description	Instrument Series name is ASCII.
series_sequence_number_u (Series, Sequence Number)	
Datatype	UINT32_T
Description	Not applicable.
series_status_c (Series, Status)	
Datatype	UINT8_T
Description	The actual status of the series:

Value Set

value	description
1	Active (both expired and not expired)
2	Suspended (temporarily stopped)
3	Issued
4	Delisted

server_name_s (Server Name)	
Datatype	char[20]
Description	Name of the server.
server_type_c (Server Type)	
Datatype	CHAR
Description	The server type at the central Exchange. Different target servers exist for different tasks. The values below are only examples.

Value Set	<table><tr><th>value</th><th>description</th></tr><tr><td>O</td><td>Order</td></tr><tr><td>Q</td><td>Query</td></tr><tr><td>D</td><td>Deal</td></tr><tr><td>A</td><td>Answer (only from the Central System)</td></tr><tr><td>I</td><td>Information</td></tr></table>
settlement_date_q (Date, Settlement)	
Datatype	INT64_T
settlement_date_s (Date, Settlement)	
Datatype	char[8]
Description	Settlement date for delivery or payment. Format YYYYMMDD.
settlement_days_n (Settlement, Days or Month)	
Datatype	UINT16_T
Description	Number of settlement days (or month) calculation rule.
settlement_instr_date_s (Date, Settlement instruction)	
Datatype	char[8]
Description	Date for generating instructions for settlement in external settlement systems. Format: YYYYMMDD.
settlement_price_type_c (Settlement Price Type)	
Datatype	UINT8_T
Description	Different types of Settlement prices
Value Set	<table><tr><th>name</th><th>value</th><th>description</th></tr><tr><td>sp_type_query_on_all</td><td>1</td><td>Apply to all types. For query use only</td></tr><tr><td>sp_type_normal</td><td>2</td><td>Normal</td></tr></table>
settle_price_i (Price, Settlement)	
Datatype	INT32_T
Description	The daily settlement price for the Series.
settl_cur_id_s (Currency, Settlement)	
Datatype	char[32]
Description	Defines the settlement currency for the instrument. The representation of the currency follows the S.W.I.F.T. handbook and ISO 3166 standard, e.g. SEK, GBP, USD and ATS.
settl_day_unit_c (Settlement Day Unit)	
Datatype	UINT8_T
Description	Describes the unit of the number of Settlement Days Rule for the instrument class

Value Set	

name	value
Not applicable	0
Days	1
Month	2

settl_price_i (Settlement Price)	
Datatype	INT32_T
Description	Defines the settlement price.
short_code (SHORT_CODE)	
Datatype	CHAR
Description	Intermediate field.
sim_item_type_c (Item type, Simulation Answer)	
Datatype	UINT8_T
Description	Flags type of item in margin simulation answer.
Value Set	

name	value	description
Sum margin requirement per currency	1	Sum margin requirement per currency Sum margin requirement per currency
Individual margin requirement single open position	2	Individual margin requirement single open position Individual margin requirement for a single open position
Individual margin requirement single delivery position	3	Individual margin requirement single delivery position Individual margin requirement for a single delivery position
Individual margin requirement single payment position	4	Individual margin requirement single payment position Individual margin requirement for a single payment position
Sum margin requirement of open and delivery positions for underlying	5	Sum margin requirement of open and delivery positions for underlying Sum margin requirement of open and delivery positions for an underlying
Sum margin requirement of payment positions for underlying	6	Sum margin requirement of payment positions for underlying Sum margin requirement of payment positions for an underlying
Prices and valuation intervals used in the calculations	7	Prices and valuation intervals used in the calculations

name	value	description
		Prices and valuation intervals used in the calculations
Volatilities and naked margin requirements for options	8	Volatilities and naked margin requirements for options Volatilities and naked margin requirements for options used in the calculations

sim_qty_q (Quantity, Simulation)	
Datatype	INT64_T
Description	Defines the quantity in simulation.
size_n (Size)	
Datatype	UINT16_T
Description	Size of following struct including header where size resides.
skewness_down_i (Skewness, Down)	
Datatype	INT32_T
Description	Skewness value down with 4 implicit decimals.
skewness_up_i (Skewness, Up)	
Datatype	INT32_T
Description	Skewness value up with 4 implicit decimals.
sort_type_c (Sort Criteria)	
Datatype	UINT8_T
Description	Not applicable.

Value Set

value	description
0	Default

so_commodity_n (Commodity code, Spin Off)	
Datatype	UINT16_T
Description	Specified if the adjusted series are moved to a new underlying compared to the original series. If keeping the original underlying, the value is zero.
so_contract_size_modifier_c (Modifier, Contract Size)	
Datatype	UINT8_T
Description	The modifier is used to recalculate the item after an underlying adjustment. The field is stored with 3 implicit decimals.

Value Set

value	description
1	Modifier is added to the item
2	Modifier is subtracted from the item
3	Modifier is multiplied with the item

value	description
4	The item is divided by the modifier factor

so_contr_size_mod_factor_i (Modifier Factor, Spin Off Contract Size)	
Datatype	INT32_T
Description	The modifier is used to recalculate the item after an underlying adjustment. The field is stored with 5 implicit decimals.
so_country_c (Market, Spin Off)	
Datatype	UINT8_T
Description	Is defined if the Spin off series is moved to a new market compared to the original series. If the original market is kept, the field is 0.
so_deal_price_modifier_c (Modifier, Spin Off Deal Price)	
Datatype	UINT8_T
Description	The modifier is used to recalculate the item after an underlying adjustment. The field is stored with 3 implicit decimals.

Value Set

value	description
1	Modifier is added to the item
2	Modifier is subtracted from the item
3	Modifier is multiplied with the item
4	The item is divided by the modifier factor

so_deal_price_mod_factor_i (Modifier Factor, Spin Off Deal Price)	
Datatype	INT32_T
Description	The modifier is used to recalculate the item after an underlying adjustment. The field is stored with 7 implicit decimals
so_market_c (Market, Spin Off)	
Datatype	UINT8_T
Description	Is defined if the Spin off series is moved to a new market compared to the original series. If the the original market is kept, the field is 0.
so_pqf_modifier_c (Modifier, Spin Off Price Quotation Factor)	
Datatype	UINT8_T
Description	The modifier is used to recalculate the item after an underlying adjustment. The field is stored with 3 implicit decimals.

Value Set

value	description
1	Modifier is added to the item
2	Modifier is subtracted from the item
3	Modifier is multiplied with the item
4	The item is divided by the modifier factor

so_pqf_mod_factor_i (Modifier Factor, Spin Off Price Quotation Factor)	
Datatype	INT32_T
Description	The modifier is used to recalculate the item after an underlying adjustment. The field is stored with 7 implicit decimals
so_strike_price_modifier_c (Modifier, Spin Off Strike Price)	
Datatype	UINT8_T
Description	The modifier is used to recalculate the item after an underlying adjustment. The field is stored with 3 implicit decimals.
Value Set	**value** / **description** 1 / Modifier is added to the item 2 / Modifier is subtracted from the item 3 / Modifier is multiplied with the item 4 / The item is divided by the modifier factor
so_strike_price_mod_factor_i (Modifier Factor, Spin Off Strike Price)	
Datatype	INT32_T
Description	The modifier is used to recalculate the item after an underlying adjustment. The field is stored with 7 implicit decimals
spinoff_c (Spinoff)	
Datatype	UINT8_T
Description	Is the actual adjustment containing also Spin off series?
Value Set	**value** / **description** 1 / Yes 2 / No
start_date_s (Date, Start)	
Datatype	char[8]
Description	Start date. Format: YYYYMMDD.
state_c (State)	
Datatype	UINT8_T
Description	Defines the state of a request.
Value Set	**name** / **value** / **description** None / 0 / None holding / 1 / Holding Object is holding and awaits countersign. holding_indirectly / 2 / Holding Indirectly Object is awaiting a holding object.

name	value	description
pending	3	Pending Object is awaiting a later operation.
active	4	Active Object has been confirmed, if it was originally holding.
completed	5	Completed A pending object has been completed.
rejected	6	Rejected Object has been rejected.
business_completed	7	Business Completed Realtime events done. This value is logically between Active and Completed.
delivered	8	Delivered Object has been completed due to delivery.
rectified	9	Rectified
deleted	10	Deleted
pending_rectify	11	Pending Rectify
expired	12	Expired
pending_authorize	13	Pending Authorize
delete_holding	14	Delete Holding Object is holding for delete and awaits countersign.

state_level_e (Level)	
Datatype	UINT16_T
Description	Indicates the level which a state applies to:

Value Set

value	description
0	All_Levels
1	Market
2	Instrument_Type
3	Instrument_Class
4	Instrument_Series
5	Underlying

state_name_s (Trading State Name)	
Datatype	char[20]

Description	The ASCII representation of the trading state.
state_number_n (Trading State Number)	
Datatype	UINT16_T
Description	The binary representation of the Trading State or Instrument Session State. Available values can be fetched by means of the Query Trading State. Value 0 is distributed when an Instrument Session State ends.
state_priority_c (State Priority)	
Datatype	UINT8_T
Description	The priority of the State, either the Trading Session State or Instrument Session State. The State Priority is a number between 1-255. 0 (zero) is for internal usage only. A higher priority has a higher number.
state_type_number_n (State Type Number)	
Datatype	UINT16_T
Description	Numeric identification of the State Type.
step_size_i (Tick Size)	
Datatype	INT32_T
Description	The tick size is the minimum valid step in the Premium or Price.
step_size_multiple_n (Tick Size, Multiple)	
Datatype	UINT16_T
Description	Tick size multiple is used to calculate the tick size for the instrument. The tick size itself is distributed in the instrument class. If the same tick size is used for all expirations, the value in this field will be 1 for all instruments.
stock_code_s (Stock Code)	
Datatype	char[6]
Description	Not applicable.
stopped_by_issue_c (Stopped By Issue)	
Datatype	UINT8_T
Description	The series is stopped from trading depending on an issue.
Value Set	**name** / **value** Yes / 1 No / 2
stop_condition_c (Stop Condition)	
Datatype	UINT8_T
Description	Condition to be met for a stop order to be activated:
Value Set	**value** / **description** 0 / No stop condition 1 / Bid price larger or equals stop price

value	description
2	Bid price less or equals stop price
3	Ask price larger or equals stop price
4	Ask price less or equals stop price
5	Last traded larger or equals stop price
6	Last traded less or equals stop price

strike_interval_i (Strike Interval)	
Datatype	INT32_T
Description	Strike interval with 4 implicit decimals.
strike_price_format_c (Strike Price, Format)	
Datatype	UINT8_T
Description	Not applicable.
strike_price_i (Strike Price)	
Datatype	INT32_T
Description	The Strike Price is a part of the binary Series for options. If the Strike Price is equal to zero, it implies that the Strike Price is not applicable. This is always an integer. The implicit number of decimals is given in the decimals, strike price field.
strike_price_modifier_c (Modifier, Strike Price)	
Datatype	UINT8_T
Description	The modifier is used to recalculate the item after an underlying adjustment. The field is stored with 3 implicit decimals.

Value Set

value	description
1	Modifier is added to the item
2	Modifier is subtracted from the item
3	Modifier is multiplied with the item
4	The item is divided by the modifier factor

strike_price_mod_factor_i (Modifier Factor, Strike Price)	
Datatype	INT32_T
Description	The modifier is used to recalculate the item after an underlying adjustment. The field is stored with 7 implicit decimals.
subscription_price_i (Subscription, Price)	
Datatype	INT32_T
Description	Not applicable.
sub_user_s (Sub User)	
Datatype	char[32]
Description	User name of real end user.

	Should be non-blank only for GENIUM INET Clearing Back Office Server.
summary_i (Summary)	
Datatype	INT32_T
Description	Defines whether or not to aggregate positions by the account level selected.
Value Set	**value** / **description** 1 / Yes 2 / No
suspended_c (Suspended)	
Datatype	UINT8_T
Description	Defines if the series is suspended or not.
Value Set	**value** / **description** 1 / Yes 2 / No
swap_lead_time_i (Swaps, Lead Time)	
Datatype	INT32_T
Description	Lead time for swaps. Represented with 1 implicit decimal.
swap_style_c (Style, Swap)	
Datatype	UINT8_T
Description	Defines if this an Instrument Group where corresponding Instrument Series are swap styled.
Value Set	**value** / **description** 0 / Not applicable 1 / Fixed-Fixed 2 / Fixed-Float 3 / Float-Float 4 / TOM next 5 / Generic
swift_member_c (SWIFT Member)	
Datatype	UINT8_T
Description	The field defines whether a member is also a SWIFT member or not.
Value Set	**value** / **description** 1 / Yes 2 / No
sw_settl_days_i (Settlement Days, Swap)	

Datatype	INT32_T
Description	Settlement days for swaps in margin calculations.
sw_trd_days_in_year_i (Swaps, Trading Days)	
Datatype	INT32_T
Description	Trading days per year in margin calculations for swap calculations.
synthetic_type_c (Type, Synthetic)	
Datatype	UINT8_T
Description	Not Applicable.
Value Set	**value** / **description** 0 / Not applicable
tailor_made_c (Tailor Made)	
Datatype	UINT8_T
Description	Is the instrument group used for tailor made created series:
Value Set	**value** / **description** 1 / Yes 2 / No
tdp_id_s (Parameter, Time Dependent Identity)	
Datatype	char[16]
Description	Time dep. param
td_long_q (Today long position)	
Datatype	INT64_T
td_short_q (Today short position)	
Datatype	INT64_T
text_buffer_s (Text, Buffer)	
Datatype	char[50000]
Description	The text buffer contains text records with an uint32 followed by the text line. The records are word aligned in the text buffer.
text_id (TEXT_ID)	
Datatype	char[12]
Description	Intermediate field.
text_line (TEXT_LINE)	
Datatype	char[80]
Description	intermediate field.
text_line_s (Text, Line)	
Datatype	char[80]

Description	One line of text information.
theta_i (Theta)	
Datatype	INT32_T
Description	The rate of change in an options value, due to time decay. Given as terms of decay over one full year. Given with 4 decimals.
third_not_s (Notation, Tertiary)	
Datatype	char[5]
Description	The currency tertiary notation.
third_rel_primary_n (Relation to Primary, Tertiary)	
Datatype	UINT16_T
Description	Relation between the first and the tertiary unit
third_unit_s (Unit, Tertiary)	
Datatype	char[15]
Description	Tertiary Unit. The currency unit, e.g. DOLLAR, CENT.
timelen (TIMELEN)	
Datatype	char[5]
Description	intermediate field.
timestamp_comp_s (Time, Computation)	
Datatype	char[5]
Description	A time stamp, "HH.MM".
timestamp_dist_s (Time, Distribution)	
Datatype	char[5]
Description	Defines a time stamp. Format: "HH.MM".
timestamp_in_q (Timestamp In)	
Datatype	INT64_T
Description	The time when an order related transaction is recieved by the central system.
timestamp_log_q (Timestamp, Last Change)	
Datatype	INT64_T
Description	Internal system time when the order change took place in the Order Book. The number represents the number of nanoseconds since 17 Nov. 1858 expressed in GMT.
time_delivery_start_s (Time, Delivery Start)	
Datatype	char[6]
Description	Delivery start time. Format: HHMMSS.
time_delivery_stop_s (Time, Delivery Stop)	
Datatype	char[6]
Description	Delivery stop time. Format: HHMMSS.
time_first_trading_s (Time, First Trading)	

Datatype	char[6]
Description	The first valid trading time of the series. The time is together with DATE, FIRST TRADING distributed as UTC. Time in ASCII, format is HHMMSS.
time_last_trading_s (Time, Last Trading)	
Datatype	char[6]
Description	The last valid trading time of the series. The time is together with DATE, LAST TRADING distributed as UTC. Time in ASCII, format is HHMMSS.
time_of_agreement_date_s (Time of agreement, date part)	
Datatype	char[8]
Description	The time when the trade was agreed, date part. Format YYYYMMDD. The date is together with Time of agreement, time part specified as UTC.
time_of_agreement_q (Time Of Agreement)	
Datatype	INT64_T
Description	When a trade report was agreed.
time_of_agreement_time_s (Time of agreement, time part)	
Datatype	char[6]
Description	The time when the trade was agreed, time part. Format HHMMSS. The time is together with Time of agreement, date part specified as UTC.
time_of_agree_gran_c (Time of agreement granularity)	
Datatype	UINT8_T
Description	Specifies if the time of agreement contains date or both date and time.
Value Set	**name** / **value** Not applicable / 0 Date / 1 Date and Time / 2
time_of_agree_req_c (Time of agreement required)	
Datatype	UINT8_T
Description	Specifies how time of agreement is specified and validated in the trade report.
Value Set	**name** / **value** Not required / 0 On first reported / 1 On both sides - not matched / 2 On both sides - must match / 3 On both sides - must match on date / 4
time_validity_n (Validity Time)	

<table>
<tr><td>Datatype</td><td>UINT16_T</td></tr>
<tr><td>Description</td><td>Defines the validity period for an order transaction, i.e. the amount of time an order will remain in the order book if not fully matched.

Of the two bytes in the field, the most significant byte (MSB) is used to define the unit of the time validity. If applicable, the least significant byte (LSB) specifies the value of the time validity, expressed in the unit defined in the most significant byte.

Example 1:

To enter an order, which is to be valid for the rest of the day, use MSB=1 and LSB=0. In binary representation this is MSB=00000001 and LSB=00000000, yielding that the Validity Time field should be set to 00000001 00000000 in binary representation, or 256 in decimal representation.

Example 2:

To enter an order, which is to be valid for three days, use MSB=5 and LSB=3. In binary representation this is MSB=00000101 and LSB=00000011, yielding that the Validity Time field should be set to 00000101 00000011 in binary representation, or 1283 in decimal representation.

Example 3:

To enter an order, which is to be valid for the current maximum time allowed, use MSB=6 and LSB=0. In binary representation this is MSB=00000110 and LSB=00000000, yielding that the Validity Time field should be set to 00000110 00000000 in binary representation, or 1536 in decimal representation.</td></tr>
<tr><td>Value Set</td><td>
<table>
<tr><th>value</th><th>description</th></tr>
<tr><td>MSB set to 0</td><td>Bouncing

The order will not be stored in the order book after the completion of order transaction, if the order is not fully matched. LSB should be set to zero.</td></tr>
<tr><td>MSB set to 1</td><td>Rest Of Day

The order will be stored in the order book for the remainder of the business day. LSB should be set to zero.</td></tr>
<tr><td>MSB set to 2</td><td>Good Till Canceled

The order will be stored in the order book until the instrument expires or the order is canceled. LSB should be set to zero.</td></tr>
<tr><td>MSB set to 5</td><td>Days

The order will be stored in the order book for the number of days specified in LSB.</td></tr>
<tr><td>MSB set to 6</td><td>Current Max

The order will be stored in the order book for the maximum amount of time allowed for the instrument. LSB should be set to zero.</td></tr>
<tr><td>MSB set to 32</td><td>Good Till Session

The order will be stored in the order book until end of the session state type specified in LSB.</td></tr>
</table>
</td></tr>
<tr><td colspan="2">tm_series_c (Tailor Made Series)</td></tr>
<tr><td>Datatype</td><td>UINT8_T</td></tr>
<tr><td>Description</td><td>Not applicable.</td></tr>
</table>

tm_template_c (Template Series)	
Datatype	UINT8_T
Description	Defines if this a template series.
Value Set	**name** / **value** Yes / 1 No / 2
total_held_q (Held, Total)	
Datatype	INT64_T
Description	The total number of held in position, i.e. including any trades for the following clearing date.
total_no_of_ask_orders_u (Ask Orders, Total Number)	
Datatype	UINT32_T
Description	Total number of ask orders.
total_no_of_bid_orders_u (Bid Orders, Total Number)	
Datatype	UINT32_T
Description	Total number of bid orders.
total_quantity_ask_u (Quantity, Total Ask)	
Datatype	INT64_T
Description	Defines the total ask quantity.
total_quantity_bid_u (Quantity, Total Bid)	
Datatype	INT64_T
Description	Defines the total bid quantity.
total_volume_i (Total Volume)	
Datatype	INT64_T
Description	Total number of units (options, futures, forwards and so on) in an order related transaction.
total_written_q (Written Total)	
Datatype	INT64_T
Description	The total number of written in position, i.e. including any trades for the following clearing date.
tot_instances_c (Total Instance)	
Datatype	UINT8_T
Description	Total instance count for multiple processes.
to_date_s (Date, To)	
Datatype	char[8]
Description	To date. Format: YYYYMMDD.
to_sequence_number_u (To Sequence Number)	
Datatype	UINT32_T
Description	To Sequence Number

<table>
<tr><td colspan="2">to_time_s (Time, To)</td></tr>
<tr><td>Datatype</td><td>char[6]</td></tr>
<tr><td>Description</td><td>Defines the to time. Format: HHMMSS.</td></tr>
<tr><td colspan="2">traded_c (Traded)</td></tr>
<tr><td>Datatype</td><td>UINT8_T</td></tr>
<tr><td>Description</td><td>Defines if the instrument is a tradable instrument or not.</td></tr>
<tr><td>Value Set</td><td><table><tr><th>name</th><th>value</th></tr><tr><td>Yes</td><td>1</td></tr><tr><td>No</td><td>2</td></tr></table></td></tr>
<tr><td colspan="2">traded_in_click_c (Traded in GENIUM)</td></tr>
<tr><td>Datatype</td><td>UINT8_T</td></tr>
<tr><td>Description</td><td>Specifies whether the series is traded in the system or not.</td></tr>
<tr><td>Value Set</td><td><table><tr><th>value</th><th>description</th></tr><tr><td>1</td><td>Yes</td></tr><tr><td>2</td><td>No</td></tr></table></td></tr>
<tr><td colspan="2">tradenumber (TRADENUMBER)</td></tr>
<tr><td>Datatype</td><td>INT32_T</td></tr>
<tr><td>Description</td><td>intermediate field.</td></tr>
<tr><td colspan="2">trader_authorization_c (Trader, Authorization)</td></tr>
<tr><td>Datatype</td><td>UINT8_T</td></tr>
<tr><td>Description</td><td>Defines if the user is allowed to act on firm orders.</td></tr>
<tr><td>Value Set</td><td><table><tr><th>name</th><th>value</th></tr><tr><td>Allow delete/alter firm orders</td><td>1</td></tr><tr><td>Disallow delete/alter firm orders</td><td>2</td></tr></table></td></tr>
<tr><td colspan="2">trades_allowed_c (Trades, Allowed)</td></tr>
<tr><td>Datatype</td><td>UINT8_T</td></tr>
<tr><td>Description</td><td>Is it allowed to store trades on the account</td></tr>
<tr><td>Value Set</td><td><table><tr><th>name</th><th>value</th></tr><tr><td>Yes</td><td>1</td></tr><tr><td>No</td><td>2</td></tr></table></td></tr>
<tr><td colspan="2">trade_condition_n (Trade Condition)</td></tr>
<tr><td>Datatype</td><td>UINT16_T</td></tr>
<tr><td>Description</td><td>The condition in which a trade was executed.</td></tr>
</table>

Value Set	<table><tr><th>name</th><th>value</th><th>description</th></tr><tr><td>trade_cnd_no_cnd</td><td>0</td><td>No condition</td></tr><tr><td>trade_cnd_late_trade</td><td>1</td><td>Late Trade</td></tr><tr><td>trade_cnd_internal_trade</td><td>2</td><td>Internal Trade/Crossing</td></tr><tr><td>trade_cnd_bulletin_board</td><td>4</td><td>Bulletin Board Trade</td></tr><tr><td>trade_cnd_buy_write</td><td>8</td><td>Buy Write</td></tr><tr><td>trade_off_market</td><td>16</td><td>Off Market</td></tr></table>
trade_number_i (Trade Number)	
Datatype	INT32_T
Description	An increasing sequence number assigned to each trade. Trade number is unique within Instrument type
trade_price_i (Price, Trade)	
Datatype	INT32_T
Description	Defines the trade price.
trade_price_sim_i (Trade Price, Simulated)	
Datatype	INT32_T
Description	Trade price used in simulation.
trade_quantity_i (Quantity, Trade)	
Datatype	INT64_T
Description	Define the number of contracts in the trade.
trade_reported_volume_u (Volume, Trade Reported)	
Datatype	INT64_T
Description	The volume today for reported trades.
trade_reporting_only_c (Only trade reports allowed)	
Datatype	UINT8_T
Description	Specifies whether the series only allows trade reporting.
Value Set	<table><tr><th>value</th><th>description</th></tr><tr><td>1</td><td>Yes</td></tr><tr><td>2</td><td>No</td></tr></table>
trade_rep_code_n (Trade Report Code)	
Datatype	UINT16_T
Description	Defines the trade report type.
trade_state_c (Trade, State)	
Datatype	UINT8_T
Description	In what state is the trade?

Value Set	<table><tr><th>value</th><th>description</th></tr><tr><td>1</td><td>Active. The trade is active.</td></tr><tr><td>2</td><td>Rectified. The trade has been rectified.</td></tr><tr><td>3</td><td>Deleted. The trade has been deleted.</td></tr><tr><td>4</td><td>Transferred. The trade has been transferred.</td></tr></table>
trade_type_c (Type, Trade)	
Datatype	UINT8_T
Description	What type of trade is it?
Value Set	<table><tr><th>value</th><th>description</th></tr><tr><td>1</td><td>Standard The trade is a normally registered trade.</td></tr><tr><td>2</td><td>Transitory Transitory. The trade is placed on a transitory account.</td></tr><tr><td>3</td><td>Overtaking Overtaking. The trade is a result of a rectify operation.</td></tr><tr><td>4</td><td>Reversing Reversing. The trade is a result of a rectify operation.</td></tr><tr><td>5</td><td>Transfer Transfer. The trade is a result of a transfer from a daily account</td></tr><tr><td>6</td><td>Exercise Exercise. The trade is an exercising part in an exercise operation</td></tr><tr><td>7</td><td>Assign Assign. The trade is an assign part in an exercise operation.</td></tr><tr><td>8</td><td>Closing Closing. The trade is a result of a closing series operation.</td></tr><tr><td>9</td><td>Issue</td></tr><tr><td>10</td><td>New_contract New_contract. The trade is a result where delivery is new contract</td></tr><tr><td>11</td><td>Delivery</td></tr><tr><td>12</td><td>Dummy_trade</td></tr><tr><td>13</td><td>Alias</td></tr><tr><td>14</td><td>Offsetting</td></tr></table>

value	description
15	Superseding
16	State_change
17	Give_up
18	Take_up

trade_venue_c (Trade venue)	
Datatype	UINT8_T
Description	Defines the Trade venue, i.e from where the trade emanates.

trading_access_c (Trading, Access)	
Datatype	UINT8_T
Description	Defines the participant trading access:

Value Set

value	description
0	Not applicable
1	Full Participant
2	Associate Participant

trading_end_c (End of Trading)	
Datatype	UINT8_T
Description	Indicates if this state is the end of the trading day:

Value Set

value	description
1	Yes
2	No

transaction_number_n (Transaction Type Number)	
Datatype	UINT16_T
Description	A number used to distinguish between different transactions to the same central subsystem.

transaction_status_i (Transaction, Status)	
Datatype	INT32_T
Description	Indicates success or failure.

Value Set

value	description
0	Success
1	Failure

transitory_c (Transitory)	
Datatype	UINT8_T
Description	Is the account a transitory account?

Value Set	**value** / **description** 1 / Yes 2 / No
trans_ack_i (Transaction, Acknowledgement)	
Datatype	INT32_T
Description	The answer to the user. Contains the same information as Txstat and indicates the action taken as a result of the transaction or a reason for an aborted transaction. See the Error Messages Reference manual for details about why transactions are aborted. Return codes vary depending on the context in which they occur, but some common examples would be:
Value Set	**value** / **description** 1 / No part of the order placed in the Order book and no part closed. 2 / The whole order closed. 3 / The order partially closed and nothing placed in the Order Book. 4 / The whole order placed in the Order Book. 6 / The order partially placed in the Order Book and partially closed. 17 / Circuit breaker started, no part of the order placed in the Order Book and no part closed. 19 / Circuit breaker started, the order partially closed and nothing placed in the Order Book. GEN_CDC_INT_CLOSED / Instrument type not open for this transaction type. MP_MATCH_LOW_VOLUME / Fill or Kill order could not be filled because of low Order Book volume
trans_or_bdx_c (Transaction or Broadcast)	
Datatype	UINT8_T
Description	Defines if Transaction Type is a transaction or a broadcast.
Value Set	**name** / **value** Transaction / 1 Broadcast / 2
tra_cl_next_day_c (Cleared Next Day)	
Datatype	CHAR
Description	Indicates whether the clearing date has been switched over to next clearing date or not for the instrument type.

Value Set	<table><tr><th>value</th><th>description</th></tr><tr><td>Y</td><td>Yes</td></tr><tr><td>N</td><td>No</td></tr></table>
trc_id_s (Trade Report Class)	
Datatype	char[10]
Description	The ID string for a trade report class. The trade report class contains a list of Trade Report Types.
trd_cur_unit_c (Traded Currency Unit)	
Datatype	UINT8_T
Description	Specifies the currency unit the instrument is traded in.
Value Set	<table><tr><th>name</th><th>value</th></tr><tr><td>Primary Unit</td><td>1</td></tr><tr><td>Secondary Unit</td><td>2</td></tr><tr><td>Tertiary Unit</td><td>3</td></tr></table>
trend_indicator_c (Trend Indicator)	
Datatype	CHAR
Description	Trend indicator for the latest price compared to the previous one.
Value Set	<table><tr><th>name</th><th>value</th><th>description</th></tr><tr><td>Up</td><td>+</td><td>Price is higher price than previously.</td></tr><tr><td>Down</td><td>-</td><td>Price is lower price than previously.</td></tr><tr><td>Same</td><td>=</td><td>Price is unchanged.</td></tr><tr><td>None</td><td></td><td>No trend available, it might for example be the first price of the day. The value is blank (space).</td></tr></table>
trr_id_s (Trade Report, Identity)	
Datatype	char[4]
Description	The ID string for a trade report type.
turnaround_today_u (Turnover, Today)	
Datatype	INT64_T
Description	The total traded amount, today.
turnaround_yesterday_u (Turnover, Yesterday)	
Datatype	INT64_T
Description	The total traded amount yesterday.

turnover_u (Turnover)	
Datatype	INT64_T
Description	The number of traded contracts during the day. If there are 100 contracts in a deal (100 bids and 100 asks), the turnover will increase by 100.
turnover_value_q (Turnover, Value)	
Datatype	INT64_T
Description	The total traded amount today.
tv_nsec (Time in nanoseconds)	
Datatype	INT32_T
Description	Elapsed time since the time in tv_sec, expressed in nanoseconds.
tv_sec (Time in seconds)	
Datatype	UINT32_T
Description	Elapsed time in seconds since the Epoch (1970-01-01 00:00:00 UTC).
tz_exchange_s (Time Zone, Exchange)	
Datatype	char[40]
Description	The time zone environment variable for the exchange. (POSIX standard) e.g. MET-1MET_DST-2,M3.5.0/2,M10.5.0/3
tz_variable_s (TZ-Variable)	
Datatype	char[40]
Description	The TZ environment variable for the exchange (POSIX standard). e.g. MET-1MET_DST-2,M3.5.0/2,M10.5.0/3
ulg_price_spread_i (Underlying Price Spread)	
Datatype	INT32_T
Description	Price spread used when only bid or ask underlying price is present. Expressed in percent, 4 implicit decimals.
ulg_vola_i (Underlying volatility value)	
Datatype	INT32_T
Description	Not applicable.
unconv_market_value_q (Unconverted market value)	
Datatype	INT64_T
Description	Calculated market value for the position. Given with 2 decimals.
underlying_issuer_s (Underlying Issuer)	
Datatype	char[6]
Description	Defines the issuer of the underlying.
underlying_price_i (Price, Underlying)	
Datatype	INT32_T
Description	Defines the price of the underying.

underlying_status_c (Underlying Status)	
Datatype	UINT8_T
Description	Define the status of the underlying.
Value Set	**value** / **description** 1 / Active 2 / Delisted

underlying_type_c (Type, Underlying)	
Datatype	UINT8_T
Description	What type of underlying is it?
Value Set	**value** / **description** 1 / Stock 2 / Currency 3 / Interest rate 4 / Energy 5 / Soft and Agrics 6 / Metal 7 / Stock Index 8 / Currency Index 9 / Interest Rate Index 10 / Energy Index 11 / Softs and Agrics Index 12 / Metal Index

undisclosed_ask_volume_c (Undisclosed Ask Volume)	
Datatype	UINT8_T
Description	Undisclosed volume on the ask side:
Value Set	**value** / **description** 1 / Yes 2 / No

undisclosed_bid_volume_c (Undisclosed Bid Volume)	
Datatype	UINT8_T
Description	Undisclosed volume on the bid side:
Value Set	**value** / **description** 1 / Yes 2 / No

<table>
<tr><td colspan="2">undisclosed_min_ord_val_i (Minimum Order Value, Undisclosed Quantity)</td></tr>
<tr><td>Datatype</td><td>INT32_T</td></tr>
<tr><td>Description</td><td>Minimum order value for undisclosed quantity orders.
The value is always expressed in the primary currency unit.
The value is defined as quantity*price*price quotation factor.</td></tr>
<tr><td colspan="2">und_price_modifier_c (Modifier, Underlying Price)</td></tr>
<tr><td>Datatype</td><td>UINT8_T</td></tr>
<tr><td>Description</td><td>The modifier is used to recalculate the item after an underlying adjustment. The field is stored with 3 implicit decimals.</td></tr>
<tr><td>Value Set</td><td><table><tr><th>value</th><th>description</th></tr><tr><td>1</td><td>Modifier is added to the item</td></tr><tr><td>2</td><td>Modifier is subtracted from the item</td></tr><tr><td>3</td><td>Modifier is multiplied with the item</td></tr><tr><td>4</td><td>The item is divided by the modifier factor</td></tr></table></td></tr>
<tr><td colspan="2">und_price_mod_factor_i (Modifier Factor, Underlying Price)</td></tr>
<tr><td>Datatype</td><td>INT32_T</td></tr>
<tr><td>Description</td><td>The modifier is used to recalculate the item after an underlying adjustment. The field is stored with 7 implicit decimals</td></tr>
<tr><td colspan="2">update_status_note_c (Status Note, Update)</td></tr>
<tr><td>Datatype</td><td>UINT8_T</td></tr>
<tr><td>Description</td><td>Create notification code in CDB, is exchange specific.</td></tr>
<tr><td>Value Set</td><td><table><tr><th>value</th><th>description</th></tr><tr><td>1</td><td>Yes</td></tr><tr><td>2</td><td>No</td></tr></table></td></tr>
<tr><td colspan="2">upper_limit_i (Premium/Price, High Limit)</td></tr>
<tr><td>Datatype</td><td>INT32_T</td></tr>
<tr><td>Description</td><td>The upper limit in the price interval.</td></tr>
<tr><td colspan="2">up_int_i (Valuation Interval, Up)</td></tr>
<tr><td>Datatype</td><td>INT32_T</td></tr>
<tr><td>Description</td><td>Defines the valuation interval up in margin calculations. Expressed in percent of underlying price. Represented with 4 implicit decimals.</td></tr>
<tr><td colspan="2">url_link_s (Link, URL)</td></tr>
<tr><td>Datatype</td><td>CHAR[255]</td></tr>
<tr><td>Description</td><td>The Link, URL field hold the full URL for a link elsewhere on the Web, typically a document.</td></tr>
<tr><td colspan="2">user_id_s (User)</td></tr>
<tr><td>Datatype</td><td>char[5]</td></tr>
</table>

Description	Defines the user signature.
usr_id_n (User, Number)	
Datatype	UINT16_T
Description	A unique number that identified the user, used when subscribing for directed broadcast information.
ust_id_s (User Type, Identity)	
Datatype	char[5]
Description	The name of the user type.
utc_date_s (UTC, Date)	
Datatype	char[8]
Description	UTC date, format: YYYYMMDD.
utc_offset_i (UTC, Offset)	
Datatype	INT32_T
Description	Current offset between UTC and the local time specified in the TZ-variable.
utc_time_s (UTC, Time)	
Datatype	char[6]
Description	UTC time, format: HHMMSS.
vag_id_s (VAG, Identity)	
Datatype	char[12]
Description	Collateral valuation group ID
val_closed_risk_down_i (Closed Risk Down)	
Datatype	INT32_T
Description	Always set to zero.
val_closed_risk_up_i (Closed Risk Up)	
Datatype	INT32_T
Description	Always set to zero.
val_ivl_base_c (Valuation Interval, Base)	
Datatype	UINT8_T
Description	Defines the base of valuation interval.
Value Set	**value** / **description** 1 / Last paid price for underlying 2 / Margin settlement price for futures/forwards 3 / Fix for futures/forwards 4 / Compatibility 5 / Margin settlement price for underlying
val_ivl_high_i (Valuation Interval, High)	

Datatype	INT32_T
Description	Defines the high end of valuation interval.
val_ivl_low_i (Valuation Interval, Low)	
Datatype	INT32_T
Description	Defines the low end of valuation interval.
val_ivl_mid_i (Valuation Interval, Mid)	
Datatype	INT32_T
Description	Define the mid point of valuation interval.
val_ivl_type_c (Valuation Interval, Type)	
Datatype	UINT8_T
Description	Defines the type of valuation interval.
Value Set	**value** / **description** 1 / Variable 2 / Fixed 3 / Variable, reversed 4 / Fixed, reversed 5 / Compatibility 6 / Money 7 / Money, reversed
vega_i (Vega)	
Datatype	INT32_T
Description	The rate of change in an options value, due to a change in the volatility of the underlying. Given with 4 decimals.
virtual_c (Virtual)	
Datatype	UINT8_T
Description	Is the underlying a virtual underlying?
Value Set	**value** / **description** 1 / Yes 2 / No
virt_commodity_n (Virtual Underlying)	
Datatype	UINT16_T
Description	When distributing broadcasts classified with information type "Instrument Class", a virtual underlying can be used to group a number of instrument classes together. The virtual underlying is used in these broadcast subscriptions. If zero, no virtual underlying is used but the real underlying code is used in broadcast subscriptions.
volatility_i (volatility)	

Datatype	INT32_T
Description	Volatility
volume_today_i (Volume, Today)	
Datatype	INT64_T
Description	Today's volume.
volume_u (Volume)	
Datatype	INT64_T
Description	Order or trade volume.
volume_yesterday_i (Volume, Yesterday)	
Datatype	INT64_T
Description	Yesterday's volume.
vol_base_i (Volatility Base)	
Datatype	INT32_T
Description	Base volatility expressed in percent with 4 implicit decimals.
vol_interval_type_c (Volatility Interval, Type)	
Datatype	UINT8_T
Description	Define the type of volatility interval.
Value Set	**name** / **value** / **description** vol_interval_type_variable / 1 / Variable Size is a percentage of nominal volatility vol_interval_type_fixed / 2 / Fixed Fixed size
vol_ivl_held_high_i (Volatility Interval Held, High)	
Datatype	INT32_T
Description	The high implied volatility used in margin calculations for held options. Expressed in percent. 4 implicit decimals
vol_ivl_held_low_i (Volatility Interval Held, Low)	
Datatype	INT32_T
Description	The low implied volatility used in margin calculations for held options. Expressed in percent. 4 implicit decimals
vol_ivl_held_mid_i (Volatility Interval Held, Mid)	
Datatype	INT32_T
Description	The mid implied volatility used in margin calculations for held options. Expressed in percent. 4 implicit decimals
vol_ivl_writ_high_i (Volatility Interval Written, High)	
Datatype	INT32_T

Description	The high implied volatility used in margin calculations for written options. Expressed in percent. 4 implicit decimals
vol_ivl_writ_low_i (Volatility Interval Written, Low)	
Datatype	INT32_T
Description	The low implied volatility used in margin calculations for written options. Expressed in percent. 4 implicit decimals
vol_ivl_writ_mid_i (Volatility Interval Written, Mid)	
Datatype	INT32_T
Description	The mid implied volatility used in margin calculations for written options. Expressed in percent. 4 implicit decimals
vol_sim_c (Volatility Simulated)	
Datatype	UINT8_T
Description	Flags the volatilities that should be used in margin simulation. 1 = Use volatilities calculated from current prices. Must be set to 1.
vol_spread_held_i (Volatility Spread, Held)	
Datatype	INT32_T
Description	Volatility spread used when single or fixed volatility is used. Expressed in percent, 4 implicit decimals.
vol_spread_writ_i (Volatility Spread, Written)	
Datatype	INT32_T
Description	Volatility spread used when single or fixed volatility is used. Expressed in percent, 4 implicit decimals.
vol_src_c (Volatility Source)	
Datatype	UINT8_T
Description	Defines how volatility is fetched for this series.

Value Set

name	value	description
Non Option	0	Non-option
Fixed	1	Fixed volatility
Individual	2	Individual volatility
Average	3	Uses average volatility this is the strike most at the money for this market, underlying, type and expiration
Strike Below	4	Uses average volatility this is the strike nearest below at the money for this market, underlying, type and expiration.
Strike Above	5	Uses average volatility this is the strike nearest above at the money for this market, underlying, type and expiration

name	value	description
Uses Average	6	Uses average volatility this is none of the three most at the money options for this market, underlying, type and expiration
Volume Weighted Last Paid	7	Uses volume-weighted last paid the option is in the expiration used in volatility calculation

vol_steps_held_i (Volatility Steps, Held)	
Datatype	INT32_T
Description	Number of volatility steps in margin calculations.

vol_steps_writ_i (Volatility Steps, Written)	
Datatype	INT32_T
Description	Number of volatility steps in margin calculations.

vol_used_c (Volatility Type)	
Datatype	CHAR
Description	Defines the type of volatility to use when evaluating options.

Value Set

value	description
F	Fixed volatility
A	Average Separate Average of the implied volatility for the three most at-the-money options with the same underlying and expiration date. Separate on long and short side based on bid and ask prices.
I	Individual Separate Implied volatility for the series itself. Separate on long and short side based on bid and ask prices.
B	Average Single Average of the implied volatility for the three most at-the-money options with the same underlying and expiration date. Single volatility based on margin settlement price.
J	Individual Single Implied volatility for the series itself. Single volatility based on margin settlement price.
V	Volume weigthed last paid Volatility based on volume-weighted last paid prices.

warning_msg_s (Warning Message)

<table>
<tr><td>Datatype</td><td>char[80]</td></tr>
<tr><td>Description</td><td>This is a warning message that will be shown at a trading state change.</td></tr>
<tr><td colspan="2">warrant_c (Warrant)</td></tr>
<tr><td>Datatype</td><td>UINT8_T</td></tr>
<tr><td>Description</td><td>If the instrument is a warrant:</td></tr>
<tr><td>Value Set</td><td><table><tr><th>value</th><th>description</th></tr><tr><td>1</td><td>Yes</td></tr><tr><td>2</td><td>No</td></tr></table></td></tr>
<tr><td colspan="2">when_issued_c (When Issued)</td></tr>
<tr><td>Datatype</td><td>UINT8_T</td></tr>
<tr><td>Description</td><td>Not applicable.</td></tr>
<tr><td>Value Set</td><td><table><tr><th>value</th><th>description</th></tr><tr><td>2</td><td>No</td></tr></table></td></tr>
<tr><td colspan="2">win_id_s (Window Class)</td></tr>
<tr><td>Datatype</td><td>char[15]</td></tr>
<tr><td>Description</td><td>Window class used in window method in margin calculations.</td></tr>
<tr><td colspan="2">writ_for_adj_i (Future Adjustment Written)</td></tr>
<tr><td>Datatype</td><td>INT32_T</td></tr>
<tr><td>Description</td><td>Adjustment factor for margin calculation of written futures and forwards. Expressed in percent with 4 implicit decimals.</td></tr>
<tr><td colspan="2">writ_marg_q (Marginables, Written)</td></tr>
<tr><td>Datatype</td><td>INT64_T</td></tr>
<tr><td>Description</td><td>The number of written marginables in a position.</td></tr>
<tr><td colspan="2">writ_val_min_i (Value Written, Min)</td></tr>
<tr><td>Datatype</td><td>INT32_T</td></tr>
<tr><td>Description</td><td>Min value for written options in margin calculations with 4 implicit decimals.</td></tr>
<tr><td colspan="2">writ_vol_down_i (Volatility Written, Down)</td></tr>
<tr><td>Datatype</td><td>INT32_T</td></tr>
<tr><td>Description</td><td>Volatility interval down for written options in margin calculations. Expressed in percent, 4 implicit decimals.</td></tr>
<tr><td colspan="2">writ_vol_min_i (Volatility Written, Min)</td></tr>
<tr><td>Datatype</td><td>INT32_T</td></tr>
<tr><td>Description</td><td>Min volatility for written options. Expressed in percent with 4 implicit decimals.</td></tr>
<tr><td colspan="2">writ_vol_up_i (Volatility Written, Up)</td></tr>
<tr><td>Datatype</td><td>INT32_T</td></tr>
</table>

Description	Volatility interval up for written options in margin calculations. Expressed in percent, 4 implicit decimals.
yyyymmdd (YYYYMMDD)	
Datatype	char[8]
Description	Intermediate field for date in YYYYMMDD format.
yyyymmdd_s (Date)	
Datatype	char[8]
Description	Date in ASCII. Format: YYYYMMDD

Index Register

A

B

C

D

E

F

G

H

I

K

L

M

N

O

P

Q

R

S

T

U

V

W

Y