NASDAQ OMX®

Genium INET℠

OMnet Message Reference

NASDAQ OMX Nordic

Version: 4.0.0201

Document ID:	OMnet_MessRef_6
Documentation Release:	GENIUM_Product_a1007
Release Date:	2013-12-05
Publication Date:	2013-12-05
Confidentiality:	Non-confidential
API Kit:	48020

Whilst all reasonable care has been taken to ensure that the details are true and not misleading at the time of publication, no liability whatsoever is assumed by OMX Technology AB, or any subsidiary of OMX Technology AB, with respect to the accuracy or any use of the information provided herein.

Any license, delivery and support of software systems etc. require entering into separate agreements with OMX Technology AB.

This document contains confidential information and may not be modified or reproduced, in whole or in part, or transmitted in any form to any third party, without the written approval from OMX Technology AB.

Copyright © 2013 OMX Technology AB.

All rights reserved.

Table of Contents

© The NASDAQ OMX Group, Inc. • 2013

© The NASDAQ OMX Group, Inc. • 2013

© The NASDAQ OMX Group, Inc. • 2013

© The NASDAQ OMX Group, Inc. • 2013

© The NASDAQ OMX Group, Inc. • 2013

© The NASDAQ OMX Group, Inc. • 2013

© The NASDAQ OMX Group, Inc. • 2013

© The NASDAQ OMX Group, Inc. • 2013

© The NASDAQ OMX Group, Inc. • 2013

© The NASDAQ OMX Group, Inc. • 2013

© The NASDAQ OMX Group, Inc. • 2013

© The NASDAQ OMX Group, Inc. • 2013

© The NASDAQ OMX Group, Inc. • 2013

© The NASDAQ OMX Group, Inc. • 2013

© The NASDAQ OMX Group, Inc. • 2013

© The NASDAQ OMX Group, Inc. • 2013

List of Tables

List of Figures

© The NASDAQ OMX Group, Inc. • 2013

© The NASDAQ OMX Group, Inc. • 2013

1 Summary of Changes

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

Changes between (47395) and (48020) for OM (a279/a282).

	Changed message	Changes	Comments
1	BU125	Changes in struct **ns_inst_series_bo:** New field: **filler_1_s**	
2	DC3	Textual changes in message description: section: Answer, comments: paragraph #1: changed	
3	DC11	Textual changes in message description: section: Answer, comments: paragraph #1: changed	
4	DQ3	Changes in answer **DA3:** Changes in struct **answer_instrument:** Changes in array **item:** New field: **settlement_type_c** Removed field: **filler_1_s**	
5	DQ22	Changes in answer **DA22:** Changes in struct **answer_instrument:** Changes in array **item:** New field: **settlement_type_c** Removed field: **filler_1_s**	
6	DQ125	Changes in answer **DA125:** Changes in struct **ns_inst_series_bo:** New field: **filler_1_s**	
7	DQ131	Changes in answer **DA131:** Changes in struct **ns_inst_type_secur:** New field: **settlement_type_c** Removed field: **filler_1_s**	
8	MO2	Textual changes in message description: section: Usage and Conditions: titled-block #1: figure #1: paragraph #1: deleted content image #1: new	

© The NASDAQ OMX Group, Inc. • 2013

© The NASDAQ OMX Group, Inc. • 2013

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.

© The NASDAQ OMX Group, Inc. • 2013

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>

© The NASDAQ OMX Group, Inc. • 2013

<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.

© The NASDAQ OMX Group, Inc. • 2013

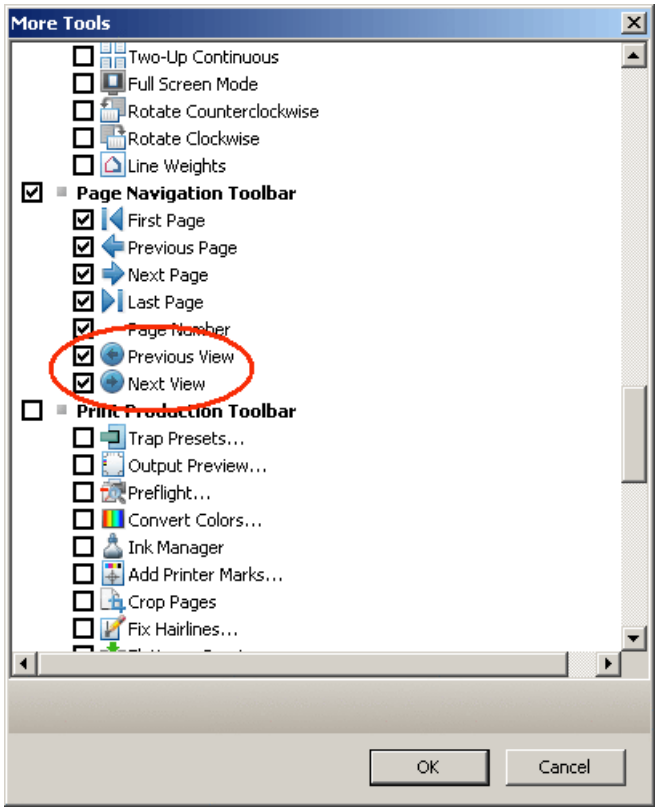

Figure 1: More Tools Dialog

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.

© The NASDAQ OMX Group, Inc. • 2013

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.

For TM combos, it is also sent for deletions.

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
```

© The NASDAQ OMX Group, Inc. • 2013

```
        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.

© The NASDAQ OMX Group, Inc. • 2013

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
```

© The NASDAQ OMX Group, Inc. • 2013

```
        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
```


© The NASDAQ OMX Group, Inc. • 2013

```
    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:

© The NASDAQ OMX Group, Inc. • 2013

```
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

Change Type

for BU9, only value "3, Modification" will be used.

Trading State Number

© The NASDAQ OMX Group, Inc. • 2013

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
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  // Valuation Group 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

© The NASDAQ OMX Group, Inc. • 2013

BROADCAST properties	
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

© The NASDAQ OMX Group, Inc. • 2013

BROADCAST properties	
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

© The NASDAQ OMX Group, Inc. • 2013

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
```


© The NASDAQ OMX Group, Inc. • 2013

```
            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).

© The NASDAQ OMX Group, Inc. • 2013

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
```


© The NASDAQ OMX Group, Inc. • 2013

```
      CHAR[2] mbs_id_s  // Minimum Bid Schedule
      char[12] valuation_group_id_s  // Valuation Group 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 BU44 [Legal Account Instrument Update BROADCAST]

3.1.12.1 Fingerprint

BROADCAST properties	
transaction type	BU44
calling sequence	omniapi_read_event_ext_ex or omniapi_read_event_block
struct name	legal_account_instrument_update_bu44
info type	general

3.1.12.2 Related Messages

DQ44, the answer will take into account any modifications made.

3.1.12.3 Purpose

The Legal Account Instrument Update broadcast is sent whenever a change has occurred.

3.1.12.4 Structure

The BU44 BROADCAST has the following structure:

```
struct legal_account_instrument_update_bu44 {
   struct broadcast_type
   UINT16_T chg_type_n  // Change Type
```


© The NASDAQ OMX Group, Inc. • 2013

```
    char[2] filler_2_s  // Filler
    struct da44 {
       struct series  // Named struct no: 50000
       char[12] acc_type_s  // Account Type
    }
}
```

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 BU47 [Haircut Update BROADCAST]

3.1.13.1 Fingerprint

BROADCAST properties	
transaction type	BU47
calling sequence	omniapi_read_event_ext_ex or omniapi_read_event_block
struct name	haircut_update_bu47
info type	general

3.1.13.2 Related Messages

DQ47, the answer will take into account any modifications made.

3.1.13.3 Purpose

The Haircut Update broadcast is sent whenever a change has occurred regarding a haircut value.

3.1.13.4 Structure

The BU47 BROADCAST has the following structure:

```
struct haircut_update_bu47 {
   struct broadcast_type
   UINT16_T chg_type_n  // Change Type
   char[2] filler_2_s  // Filler
   struct da47 {
      struct series  // Named struct no: 50000
      char[40] description_s  // Description
      UINT32_T haircut_rate_u  // Haircut Rate
      UINT32_T time_to_maturity_u  // Time to maturity
   }
}
```

© The NASDAQ OMX Group, Inc. • 2013

3.1.13.5 Usage and conditions

Change Type

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

3.1.14 BU50 [Non-Settlement Days Update BROADCAST]

3.1.14.1 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

3.1.14.2 Related Messages

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

3.1.14.3 Purpose

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

3.1.14.4 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
   }
}
```

3.1.14.5 Usage and conditions

Change Type

© The NASDAQ OMX Group, Inc. • 2013

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

3.1.15 BU53 [Corporate Action Update BROADCAST]

3.1.15.1 Fingerprint

BROADCAST properties	
transaction type	BU53
calling sequence	omniapi_read_event_ext_ex or omniapi_read_event_block
struct name	corp_action_update_bu53
info type	general

3.1.15.2 Purpose

This broadcast is sent when a new corporate action is added during the day. One broadcast can contain several items.

3.1.15.3 Structure

The BU53 BROADCAST has the following structure:

```
struct corp_action_update_bu53 {
   struct broadcast_type
   UINT16_T chg_type_n  // Change Type
   UINT16_T items_n  // Items
   Array ITEM [max no: 50] {
      struct da53 {
         struct series  // Named struct no: 50000
         char[2] corp_action_code_s  // Code, Corporate Action
         UINT8_T corp_action_type_c  // Corporate Action Type
         UINT8_T corp_action_status_c  // Status, Corporate Action
         UINT8_T corp_action_level_c  // Level, Corporate Action
         char[3] filler_3_s  // Filler
      }
   }
}
```

3.1.15.4 Usage and conditions

The content within each item is the same as sent in DA53.

© The NASDAQ OMX Group, Inc. • 2013

3.1.16 BU54 [Valid Sector Codes Update BROADCAST]

3.1.16.1 Fingerprint

BROADCAST properties	
transaction type	BU54
calling sequence	omniapi_read_event_ext_ex or omniapi_read_event_block
struct name	valid_sector_code_update_bu54
info type	general

3.1.16.2 Purpose

This broadcast is sent when a new sector code is added during the day.

3.1.16.3 Structure

The BU54 BROADCAST has the following structure:

```
struct valid_sector_code_update_bu54 {
   struct broadcast_type
   UINT16_T chg_type_n  // Change Type
   char[2] filler_2_s  // Filler
   struct da54 {
      char[4] sector_code_s  // Sector Code
      char[40] description_s  // Description
   }
}
```

3.1.16.4 Usage and Conditions

The broadcast contains one item as sent in DA54.

3.1.17 BU87 [Market Maker Protection Update BROADCAST]

3.1.17.1 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

© The NASDAQ OMX Group, Inc. • 2013

3.1.17.2 Related Messages

DC87, DQ87

3.1.17.3 Purpose

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

3.1.17.4 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
   }
}
```

3.1.18 BU88 [Turnover List Update VIB]

3.1.18.1 Fingerprint

VIB properties	
transaction type	BU88
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

DQ88

3.1.18.3 Purpose

This broadcast is used to send out information about a new or changed turnover list.

© The NASDAQ OMX Group, Inc. • 2013

3.1.18.4 Structure

The BU88 VIB has the following structure:

```
struct broadcast_segment_hdr
Sequence {
   struct item_hdr
   Sequence {
      struct sub_item_hdr
      Choice {
         struct ns_turnover_list_base  // Named struct no: 37701
         struct ns_turnover_list_item  // Named struct no: 37702
      }
   }
}
```

3.1.18.5 Usage and Conditions

For general information on the content of the broadcasts, refer to section DQ88.

3.1.19 BU90 [Pre Trade Limit Update VIB]

3.1.19.1 Fingerprint

VIB properties	
transaction type	BU90
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

3.1.19.2 Related Messages

DQ90, DC90

3.1.19.3 Purpose

This broadcast is used to send out information about new and changed Pre Trade Limit Groups.

3.1.19.4 Structure

The BU90 VIB has the following structure:

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


© The NASDAQ OMX Group, Inc. • 2013

```
        Sequence {
           struct sub_item_hdr
           Choice {
              struct ns_pre_trade_limit_id  // Named struct no: 37805
              struct ns_pre_trade_limit  // Named struct no: 37801
              struct ns_pre_trade_limit_user  // Named struct no: 37802
              struct ns_pre_trade_limit_not  // Named struct no: 37804
              struct ns_pre_trade_limit_param  // Named struct no: 37803
           }
        }
     }
```

3.1.19.5 Structure Contents

pre_trade_limit

is sent once for each pre trade risk group.

pre_trade_limit_user

is repeated once for every sponsored user, if any are connected to the pre trade limit risk group.

pre_trade_limit_param

is repeated once for every instrument type or instrument class that are connected to the group.

pre_trade_limit_not

is repeated once for every mail receivers, if any are connected to the pre trade risk limit group.

3.1.20 BU92 [Strip Series Update BROADCAST]

3.1.20.1 Fingerprint

BROADCAST properties	
transaction type	BU92
calling sequence	omniapi_read_event_ext_ex or omniapi_read_event_block
struct name	strip_series_update
info type	general

3.1.20.2 Related Messages

DQ92

3.1.20.3 Purpose

This broadcast is used to inform about the relation between a specific strip series and its corresponding cleared series.

© The NASDAQ OMX Group, Inc. • 2013

3.1.20.4 Structure

The BU92 BROADCAST has the following structure:

```
struct strip_series_update {
   struct broadcast_type
   UINT16_T chg_type_n  // Change Type
   char[2] filler_2_s  // Filler
   struct da92 {
      struct series  // Named struct no: 50000
      UINT16_T items_n  // Items
      UINT8_T strip_range_c  // Strip range
      UINT8_T split_rule_c  // Split rule
      Array STRIP_SERIES [max no: 52] {
         struct series  // Named struct no: 50000
      }
   }
}
```

3.1.20.5 Usage and Conditions

The broadcast is sent when a new strip series is defined intraday.

3.1.21 BU120 [Delta Underlying Update VIB]

3.1.21.1 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

3.1.21.2 Related Messages

DQ120

3.1.21.3 Purpose

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

3.1.21.4 Structure

The BU120 VIB has the following structure:

© The NASDAQ OMX Group, Inc. • 2013

```
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
         struct ns_index_linked  // Named struct no: 37204
         struct ns_underlying_power  // Named struct no: 37206
         struct ns_underlying_ext3  // Named struct no: 37209
         struct ns_reference_rate  // Named struct no: 37210
         struct ns_index_value  // Named struct no: 37211
         struct ns_lottery_bonds  // Named struct no: 37212
         struct ns_convertibles  // Named struct no: 37213
         struct ns_derived_from  // Named struct no: 37214
      }
   }
}
```

3.1.21.5 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 or the full answer business date.

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.

3.1.22 BU121 [Delta Underlying Update for Back Office VIB]

3.1.22.1 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

3.1.22.2 Related Messages

DQ121

3.1.22.3 Purpose

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

© The NASDAQ OMX Group, Inc. • 2013

3.1.22.4 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
         struct ns_index_linked  // Named struct no: 37204
         struct ns_underlying_power  // Named struct no: 37206
         struct ns_underlying_ext3  // Named struct no: 37209
         struct ns_reference_rate  // Named struct no: 37210
         struct ns_index_value  // Named struct no: 37211
         struct ns_lottery_bonds  // Named struct no: 37212
         struct ns_convertibles  // Named struct no: 37213
         struct ns_derived_from  // Named struct no: 37214
      }
   }
}
```

3.1.22.5 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**.

3.1.23 BU122 [Delta Instrument Class Update VIB]

3.1.23.1 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

3.1.23.2 Related Messages

DQ122

3.1.23.3 Purpose

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

3.1.23.4 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_calc_rule  // Named struct no: 37104
         struct ns_inst_class_secur  // Named struct no: 37105
         struct ns_inst_class_leg_calc_rule  // Named struct no: 37115
         struct ns_price_tick_corr  // Named struct no: 37113
         struct ns_inst_class_trr_def_publ  // Named struct no: 37118
         struct ns_inst_class_ext6  // Named struct no: 37120
      }
   }
}
```

3.1.23.5 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**.

© The NASDAQ OMX Group, Inc. • 2013

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

3.1.24.1 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

3.1.24.2 Related Messages

DQ123

3.1.24.3 Purpose

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

3.1.24.4 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_calc_rule  // Named struct no: 37104
         struct ns_inst_class_secur  // Named struct no: 37105
         struct ns_inst_class_cms  // Named struct no: 37114
         struct ns_inst_class_leg_calc_rule  // Named struct no: 37115
         struct ns_price_tick_corr  // Named struct no: 37113
         struct ns_inst_class_trr_def_publ  // Named struct no: 37118
         struct ns_inst_class_ext6  // Named struct no: 37120
      }
   }
}
```

© The NASDAQ OMX Group, Inc. • 2013

3.1.24.5 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**.

3.1.25 BU124 [Delta Instrument Series Update VIB]

3.1.25.1 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

3.1.25.2 Related Messages

DQ124

3.1.25.3 Purpose

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

3.1.25.4 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
```


© The NASDAQ OMX Group, Inc. • 2013

```
            struct ns_inst_series_power  // Named struct no: 37303
            struct ns_inst_series_repo  // Named struct no: 37304
            struct ns_inst_series_leg_flow  // Named struct no: 37309
         }
      }
   }
```

3.1.25.5 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**.

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

3.1.26.1 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

3.1.26.2 Related Messages

DQ125

3.1.26.3 Purpose

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

3.1.26.4 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_power  // Named struct no: 37303
               struct ns_inst_series_repo  // Named struct no: 37304
               struct ns_inst_series_bo  // Named struct no: 37306
               struct ns_inst_series_leg_flow  // Named struct no: 37309
               struct ns_inst_series_ext5  // Named struct no: 37313
            }
         }
      }
```

3.1.26.5 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**.

3.1.27 BU126 [Combo Series Update VIB]

3.1.27.1 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

3.1.27.2 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)

3.1.27.3 Purpose

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

3.1.27.4 Structure

The BU126 VIB has the following structure:

© The NASDAQ OMX Group, Inc. • 2013

```
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
      }
   }
}
```

3.1.27.5 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.

3.1.28 BU134 [Account Type update VIB]

3.1.28.1 Fingerprint

VIB properties	
transaction type	BU134
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

3.1.28.2 Related Messages

DQ134

3.1.28.3 Purpose

This broadcast is sent whenever a change has occurred regarding an account type.

3.1.28.4 Structure

The BU134 VIB has the following structure:

```
struct broadcast_segment_hdr
Sequence {
   struct item_hdr
   Sequence {
      struct sub_item_hdr
      Choice {
         struct ns_account_type_basic  // Named struct no: 37901
```

```
        }
    }
}
```

3.1.28.5 Usage and Conditions

Change Type

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

3.1.29 BU135 [Market Maker Obligations update VIB]

3.1.29.1 Fingerprint

VIB properties	
transaction type	BU135
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

3.1.29.2 Related Messages

DQ135

3.1.29.3 Purpose

This broadcast is used to send out information about a market maker obligations.

3.1.29.4 Structure

The BU135 VIB has the following structure:

```
struct broadcast_segment_hdr
Sequence {
   struct item_hdr
   Sequence {
      struct sub_item_hdr
      Choice {
         struct ns_price_quote_resp  // Named struct no: 37951
         struct ns_vld_max_spread  // Named struct no: 37952
         struct ns_price_quote_criteria  // Named struct no: 37953
      }
   }
}
```


© The NASDAQ OMX Group, Inc. • 2013

3.1.29.5 Usage and Conditions

The struct **ns_vld_max_spread** contains all unique Max Spreads that are referenced from struct **ns_price_quote_criteria**.

3.1.29.6 Structure Contents

The DA135 VIA has the following structure:

```
struct broadcast_segment_hdr
Sequence {
   struct item_hdr
   Sequence {
      struct sub_item_hdr
      Choice {
         struct ns_price_quote_resp     // Named struct no: 37951
         struct ns_vld_max_spread       // Named struct no: 37952
         struct ns_price_quote_criteria // Named struct no: 37953
      }
   }
}
```

3.1.30 DC3 [Add TM Combo QUERY]

3.1.30.1 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

3.1.30.2 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.

© The NASDAQ OMX Group, Inc. • 2013

3.1.30.3 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
   }
}
```

3.1.30.4 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

© The NASDAQ OMX Group, Inc. • 2013

3.1.30.5 Answer Structure

The DI3 ANSWER has the following structure:

```
struct answer_add_tm_combo {
   struct transaction type
   struct series  // Named struct no: 50000
}
```

3.1.30.6 Answer, comments

The answer received contains the binary code of the created TM Combo as in BU2 and BU126.

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.

3.1.31 DC11 [Add TM Combo QUERY]

3.1.31.1 Fingerprint

QUERY properties	
transaction type	DC11
calling sequence	omniapi_query_ex
struct name	add_tm_combo_ext
facility	EP5
partitioned	false
segmented	false
answers	DI11

ANSWER properties	
transaction type	DI11
struct name	answer_add_tm_combo
segmented	false

3.1.31.2 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).

© The NASDAQ OMX Group, Inc. • 2013

3.1.31.3 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. This query works in the same way as DC3. But instead of 4 legs, the maximum number of legs in DC11 is 5.

Note:

Regardless whether the combination was created using DC3 or DC11:

- All TM Combinations will be available in BU124/DQ124 and BU126/DQ126.
- All TM Combinations with up to 4 legs will be available in BU2/DQ2 and BU5/DQ5.

3.1.31.4 Structure

The DC11 QUERY has the following structure:

```
struct add_tm_combo_ext {
   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: 5] {
      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
   }
}
```

3.1.31.5 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:	

© The NASDAQ OMX Group, Inc. • 2013

Ratio	1
Operation if Buy	B
Operation if Sell	S
Instrument 2:	
Ratio	2
Operation if Buy	S
Operation if Sell	B

3.1.31.6 Answer Structure

The DI11 ANSWER has the following structure:

```
struct answer_add_tm_combo {
   struct transaction type
   struct series  // Named struct no: 50000
}
```

3.1.31.7 Answer, comments

The answer received contains the binary code of the created TM Combo as in BU2 and BU126.

The DI11 answer can however also contain the binary code of an already existing Combo series corresponding to what is sent in the DC11, as well as an already existing Combo series that is a mirrored version of what is sent in the DC11. In order to handle order entry of Tailor Made Combos correctly, a front-end application must be able to handle a case where the DI11 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.

3.1.32 DC80 [Suspend/Resume Participant TRANSACTION]

3.1.32.1 Fingerprint

TRANSACTION properties	
transaction type	DC80
calling sequence	omniapi_tx_ex
struct name	susp_res_participant
facility	EP0
partitioned	false

3.1.32.2 Purpose

This transaction is used to suspend or resume an own participant for trading.

© The NASDAQ OMX Group, Inc. • 2013

3.1.32.3 Structure

The DC80 TRANSACTION has the following structure:

```
struct susp_res_participant {
   struct transaction type
   struct series  // Named struct no: 50000
   char[2] country_id_s  // Name, Country
   char[5] ex_customer_s  // Customer, Identity
   CHAR trading_suspend_resume_c  // Trading, Suspend/Resume
}
```

3.1.32.4 Return Codes

Resumed	Suspended
1 (int)	2 (int)

3.1.33 DC82 [Add generic TM repo QUERY]

3.1.33.1 Fingerprint

QUERY properties	
transaction type	DC82
calling sequence	omniapi_query_ex
struct name	add_gen_tm_repo
facility	EP0
partitioned	false
segmented	false
answers	DI82

ANSWER properties	
transaction type	DI82
struct name	answer_add_gen_tm_repo
segmented	false

3.1.33.2 Purpose

The purpose of this transaction is to add a Tailor-Made Repo instrument. The transaction is sent as a query, because the added instrument is returned as an answer. If the instrument already exists, the existing instrument is returned instead.

© The NASDAQ OMX Group, Inc. • 2013

3.1.33.3 Structure

The DC82 QUERY has the following structure:

```
struct add_gen_tm_repo {
   struct transaction type
   struct series  // Named struct no: 50000
   struct series_template {
      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
   }
   char[8] start_date_s  // Date, Start
   char[8] end_date_s  // Date, End
}
```

3.1.33.4 Usage and conditions

Series

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

Series Template

is the series used as a template for the new series. The template series must be a Repo instrument.

Start date

is the start date of the new requested series.

End date

is the end date of the new requested series.

3.1.33.5 Answer Structure

The DI82 ANSWER has the following structure:

```
struct answer_add_gen_tm_repo {
   struct transaction type
   struct series  // Named struct no: 50000
}
```

© The NASDAQ OMX Group, Inc. • 2013

3.1.34 DC86 [Create TM Instrument QUERY]

3.1.34.1 Fingerprint

QUERY properties	
transaction type	DC86
calling sequence	omniapi_query_ex
struct name	create_tm_instrument
facility	EP0
partitioned	false
segmented	false
answers	DI86

ANSWER properties	
transaction type	DI86
struct name	answer_create_tm_instrument
segmented	false

3.1.34.2 Purpose

This transaction is used to create a tailor made derivative instrument. The transaction is sent as a query because the added or existing instrument is returned in the answer.

3.1.34.3 Structure

The DC86 QUERY has the following structure:

```
struct create_tm_instrument {
   struct transaction_type
   struct series  // Named struct no: 50000
   INT32_T strike_price_i  // Strike Price
   UINT16_T commodity_n  // Commodity Code
   char[8] date_expiration_s  // Date, Expiration
   char[2] filler_2_s  // Filler
}
```

3.1.34.4 Usage and conditions

Series

Contains the template instrument that is used as base for the new TM Instrument created.

Commodity Code

© The NASDAQ OMX Group, Inc. • 2013

Is the Underlying code the TM Instrument should be connected to.

Expiration Date

Is the expiration date for the TM Instrument.

Strike Price

Is the strike price for the TM Instrument. It must be defined as an integer with implicit decimals as given in Decimal, Strike Price defined for the Instrument Class connected to the template instrument.

3.1.34.5 Answer Structure

The DI86 ANSWER has the following structure:

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

3.1.34.6 Answer, comments

The answer received contains the binary code of the created TM Instrument.

3.1.35 DC87 [Set Market Maker Protection TRANSACTION]

3.1.35.1 Fingerprint

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

3.1.35.2 Related Messages

BU87, DQ87

3.1.35.3 Purpose

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

3.1.35.4 Structure

The DC87 TRANSACTION has the following structure:

© The NASDAQ OMX Group, Inc. • 2013

```
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
   }
}
```

3.1.35.5 Usage and conditions

Series

Should be filled with 0 (zero)

3.1.36 DC90 [Set Pre Trade Limit VIT]

3.1.36.1 Fingerprint

VIT properties	
transaction type	DC90
calling sequence	omniapi_tx_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	false

3.1.36.2 Related Messages

BU90, DQ90

3.1.36.3 Purpose

This transaction is used by the Sponsoring Participant to add or update own Pre Trade Risk Groups.

3.1.36.4 Structure

The DC90 VIT has the following structure:

© The NASDAQ OMX Group, Inc. • 2013

```
struct trans_otri_hdr {
   struct transaction_type
   UINT16_T items_n  // Items
   UINT16_T size_n  // Size
}
Sequence {
   struct item_hdr
   Sequence {
      struct sub_item_hdr
      Choice {
         struct ns_pre_trade_limit  // Named struct no: 37801
         struct ns_pre_trade_limit_user  // Named struct no: 37802
         struct ns_pre_trade_limit_not  // Named struct no: 37804
         struct ns_pre_trade_limit_param  // Named struct no: 37803
      }
   }
}
```

3.1.36.5 Usage and conditions

Each update should contain the full data for the group. This full data can be retrieved with DQ90.

If the Pre Trade Risk Group has both current and pending items, and the intraday update is sent, the following rules apply:

- If values in current and pending items are the same, a new updated value will be populated in both current and pending item.
- If values in current and pending items are different, a new updated value will be populated in current item and pending item will keep its value.

Pre Trade Limit Suffix

is used as a suffix in the identity of the created group in order to generate a unique id.

Name, Country, Customer, Identity

represents the Sponsoring Participant, must be the same as the sender of the transaction.

Sponsoring User

must be connected to the Sponsoring Participant.

Sponsored Client, Country, Sponsored Client, Customer

represents the Sponsored Client. Any users specified in pre_trade_limit_user must be connected to this participant.

Intraday

Only a subset of Pre Trade Risk Group parameters are applicable for intraday update:

- update (change or disabling) of parameters defined in Pre Trade Limit Group
- update of notification parameters

© The NASDAQ OMX Group, Inc. • 2013

- update of order rate limit.

3.1.36.6 Structure Contents

Each update should contain the full data for the group. This full data can be retrieved with DQ90.

pre_trade_limit

should be sent only once, contains e.g. the identity of the Sponsoring Participant and the Sponsored Client.

pre_trade_limit_user

should be repeated for every sponsored user, if any.

pre_trade_limit_param

should be repeated for every instrument type or instrument class that should be included in the group.

pre_trade_limit_not

should be repeated for every mail receivers, if any.

3.1.37 DQ2 [Series QUERY]

3.1.37.1 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

3.1.37.2 Related Messages

BU2

© The NASDAQ OMX Group, Inc. • 2013

3.1.37.3 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.

3.1.37.4 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
}
```

3.1.37.5 Usage and Conditions

In the case of a full answer, series denoted as Traded=yes and with a pased Last trade Date will also be returned if today is regarded as an SQ day for the instrument series.

Series

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

3.1.37.6 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
```

© The NASDAQ OMX Group, Inc. • 2013

```
        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
    }
}
```

3.1.37.7 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

© The NASDAQ OMX Group, Inc. • 2013

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.

3.1.38 DQ3 [Instrument Type QUERY]

3.1.38.1 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
segmented	true

3.1.38.2 Purpose

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

3.1.38.3 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
}
```

3.1.38.4 Usage and conditions

Series

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

3.1.38.5 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
      UINT8_T pricing_method_c  // Pricing method
      UINT8_T settlement_type_c  // Settlement, Type
   }
}
```

3.1.38.6 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.

3.1.39 DQ4 [Underlying QUERY]

3.1.39.1 Fingerprint

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

© The NASDAQ OMX Group, Inc. • 2013

ANSWER properties	
transaction type	DA4
struct name	answer_underlying
segmented	true

3.1.39.2 Related Messages

BU4

3.1.39.3 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).

3.1.39.4 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
}
```

3.1.39.5 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 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
```

© The NASDAQ OMX Group, Inc. • 2013

```
        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
    }
}
```

3.1.39.7 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

© The NASDAQ OMX Group, Inc. • 2013

are used to interpret the Price Information for the Underlying.

3.1.40 DQ5 [Combination QUERY]

3.1.40.1 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

3.1.40.2 Related Messages

BU5

3.1.40.3 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).

3.1.40.4 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
}
```

3.1.40.5 Usage and conditions

Series

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

3.1.40.6 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
      }
   }
}
```

3.1.40.7 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.

3.1.41 DQ6 [Broker Signatures QUERY]

3.1.41.1 Fingerprint

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

© The NASDAQ OMX Group, Inc. • 2013

QUERY properties	
answers	DA6

ANSWER properties	
transaction type	DA6
struct name	answer_broker
segmented	true

3.1.41.2 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.

3.1.41.3 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
}
```

3.1.41.4 Usage and Conditions

Series

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

3.1.41.5 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
```

© The NASDAQ OMX Group, Inc. • 2013

```
        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
        }
     }
  }
```

3.1.41.6 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.

3.1.42 DQ7 [Market QUERY]

3.1.42.1 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

© The NASDAQ OMX Group, Inc. • 2013

3.1.42.2 Purpose

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

3.1.42.3 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
}
```

3.1.42.4 Usage and Conditions

Series

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

3.1.42.5 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
   }
}
```

3.1.42.6 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.

© The NASDAQ OMX Group, Inc. • 2013

3.1.43 DQ8 [Instrument Group QUERY]

3.1.43.1 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

3.1.43.2 Purpose

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

3.1.43.3 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
}
```

3.1.43.4 Usage and Conditions

Series

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

3.1.43.5 Answer Structure

The DA8 ANSWER has the following structure:

© The NASDAQ OMX Group, Inc. • 2013

```
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
      UINT8_T overnight_index_swap_c  // OIS Overnight index swap
      CHAR filler_1_s  // Filler
   }
}
```

3.1.43.6 Answer, comments

The answer received contains a list of instrument groups.

3.1.44 DQ9 [Series Backoffice QUERY]

3.1.44.1 Fingerprint

QUERY properties	
transaction type	DQ9
calling sequence	omniapi_query_ex
struct name	query_series
facility	EP0
partitioned	false

QUERY properties	
segmented	true
answers	DA9

ANSWER properties	
transaction type	DA9
struct name	answer_series_bo
segmented	true

3.1.44.2 Related Messages

BU9

3.1.44.3 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).

3.1.44.4 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
}
```

3.1.44.5 Usage and conditions

Series

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

3.1.44.6 Answer Structure

The DA9 ANSWER has the following structure:

```
struct answer_series_bo {
```


© The NASDAQ OMX Group, Inc. • 2013

```
    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
    }
 }
```

3.1.44.7 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.

3.1.45 DQ10 [Instrument Class QUERY]

3.1.45.1 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

3.1.45.2 Related Messages

BU10

3.1.45.3 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).

© The NASDAQ OMX Group, Inc. • 2013

3.1.45.4 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
}
```

3.1.45.5 Usage and conditions

Series

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

3.1.45.6 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
      }
```

© The NASDAQ OMX Group, Inc. • 2013

```
        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  // Valuation Group Identity ; Of type:
VAG_ID_S
        char[3] filler_3_s  // Filler
    }
}
```

3.1.45.7 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**.

3.1.46 DQ12 [Account Type QUERY]

3.1.46.1 Fingerprint

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

© The NASDAQ OMX Group, Inc. • 2013

QUERY properties	
answers	DA12

ANSWER properties	
transaction type	DA12
struct name	answer_account_type
segmented	true

3.1.46.2 Related Messages

BU12

3.1.46.3 Purpose

This query retrieves all existing account types in the system.

3.1.46.4 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
}
```

3.1.46.5 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
   }
}
```

© The NASDAQ OMX Group, Inc. • 2013

3.1.47 DQ13 [Account Fee Type QUERY]

3.1.47.1 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

3.1.47.2 Related Messages

BU13

3.1.47.3 Purpose

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

3.1.47.4 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
}
```

3.1.47.5 Answer Structure

The DA13 ANSWER has the following structure:

```
struct answer_account_fee_type {
   struct transaction_type
   UINT16_T segment_number_n  // Segment Number
```


© The NASDAQ OMX Group, Inc. • 2013

```
   UINT16_T items_n  // Items
   Array ITEM [max no: 100] {
      char[12] fee_type_s  // Account Fee Type
      char[40] description_s  // Description
   }
}
```

3.1.48 DQ14 [Underlying Adjustment QUERY]

3.1.48.1 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

3.1.48.2 Purpose

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

3.1.48.3 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
}
```

3.1.48.4 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.

3.1.48.5 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
```


© The NASDAQ OMX Group, Inc. • 2013

```
        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
        }
     }
  }
```

3.1.48.6 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.

3.1.49 DQ15 [Converted Series QUERY]

3.1.49.1 Fingerprint

QUERY properties	
transaction type	DQ15
calling sequence	omniapi_query_ex
struct name	query_converted_series
facility	EP0
partitioned	false
segmented	true
answers	DA15

ANSWER properties	
transaction type	DA15
struct name	answer_converted_series
segmented	true

3.1.49.2 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.

3.1.49.3 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
}
```

3.1.49.4 Usage and Conditions

Adjustment Identifier

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

3.1.49.5 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
   }
}
```

3.1.49.6 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

© The NASDAQ OMX Group, Inc. • 2013

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.

3.1.50 DQ16 [Series Delivery QUERY]

3.1.50.1 Fingerprint

QUERY properties	
transaction type	DQ16
calling sequence	omniapi_query_ex
struct name	query_series_delivery
facility	EP0
partitioned	false
segmented	true
answers	DA16

ANSWER properties	
transaction type	DA16
struct name	answer_series_delivery
segmented	true

3.1.50.2 Purpose

The purpose of this query is to receive information how an expired series will be delivered.

3.1.50.3 Structure

The DQ16 QUERY has the following structure:

```
struct query_series_delivery {
   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

is filled in with **Commodity Code** and **Expiration Date**.

The query is used only for series that result in deliveries of several other series at expiration.

3.1.50.5 Answer Structure

The DA16 ANSWER has the following structure:

```
struct answer_series_delivery {
   struct transaction_type
   UINT16_T segment_number_n  // Segment Number
   UINT16_T items_n  // Items
   Array ITEM [max no: 10] {
      struct series  // Named struct no: 50000
      UINT16_T items_n  // Items
      char[2] filler_2_s  // Filler
      Array SERIES_DELIVERY [max no: 400] {
         struct series  // Named struct no: 50000
      }
   }
}
```

3.1.51 DQ18 [Non-Trading Days QUERY]

3.1.51.1 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

© The NASDAQ OMX Group, Inc. • 2013

3.1.51.2 Related Messages

BU18

3.1.51.3 Purpose

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

3.1.51.4 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
}
```

3.1.51.5 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**.

3.1.51.6 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
   }
}
```

© The NASDAQ OMX Group, Inc. • 2013

3.1.52 DQ19 [Underlying Backoffice QUERY]

3.1.52.1 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

3.1.52.2 Related Messages

BU19

3.1.52.3 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).

3.1.52.4 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
}
```


© The NASDAQ OMX Group, Inc. • 2013

3.1.52.5 Usage and conditions

Series

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

3.1.52.6 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
```

© The NASDAQ OMX Group, Inc. • 2013

```
        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
    }
}
```

3.1.52.7 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.

3.1.53 DQ20 [Instrument Class Backoffice QUERY]

3.1.53.1 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

3.1.53.2 Related Messages

BU20

© The NASDAQ OMX Group, Inc. • 2013

3.1.53.3 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).

3.1.53.4 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
}
```

3.1.53.5 Usage and conditions

Series

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

3.1.53.6 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
```

© The NASDAQ OMX Group, Inc. • 2013

```
        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  // Valuation Group Identity ; Of type:
VAG_ID_S
        char[3] filler_3_s  // Filler
     }
}
```

3.1.53.7 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**.

3.1.54 DQ22 [Instrument Type Backoffice QUERY]

3.1.54.1 Fingerprint

QUERY properties	
transaction type	DQ22

© The NASDAQ OMX Group, Inc. • 2013

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

ANSWER properties	
transaction type	DA22
struct name	answer_instrument
segmented	true

3.1.54.2 Purpose

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

3.1.54.3 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
}
```

3.1.54.4 Usage and conditions

Series

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

3.1.54.5 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
        UINT8_T pricing_method_c  // Pricing method
        UINT8_T settlement_type_c  // Settlement, Type
    }
}
```

3.1.54.6 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.

3.1.55 DQ23 [Market Backoffice QUERY]

3.1.55.1 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

3.1.55.2 Purpose

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

3.1.55.3 Structure

The DQ23 QUERY has the following structure:

© The NASDAQ OMX Group, Inc. • 2013

```
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
}
```

3.1.55.4 Usage and Conditions

Series

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

3.1.55.5 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
   }
}
```

3.1.55.6 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.

3.1.56 DQ24 [Exchange QUERY]

3.1.56.1 Fingerprint

QUERY properties	
transaction type	DQ24
calling sequence	omniapi_query_ex

© The NASDAQ OMX Group, Inc. • 2013

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

3.1.56.2 Purpose

This query provides information on all exchanges in the system.

3.1.56.3 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
}
```

3.1.56.4 Usage and conditions

Series

must be zeroed.

3.1.56.5 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
```


© The NASDAQ OMX Group, Inc. • 2013

```
            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
        }
    }
}
```

3.1.56.6 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.

3.1.57 DQ28 [Central Group QUERY]

3.1.57.1 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

3.1.57.2 Related Messages

BU28

3.1.57.3 Purpose

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

© The NASDAQ OMX Group, Inc. • 2013

3.1.57.4 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
}
```

3.1.57.5 Usage and Conditions

Series

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

3.1.57.6 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
   }
}
```

3.1.57.7 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 NASDAQ OMX Group, Inc. • 2013

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.

3.1.58 DQ29 [Trading State QUERY]

3.1.58.1 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

3.1.58.2 Purpose

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

3.1.58.3 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
}
```

3.1.58.4 Usage and Conditions

Series

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

3.1.58.5 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
   }
}
```

3.1.58.6 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.

3.1.59 DQ30 [User Type Info QUERY]

3.1.59.1 Fingerprint

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

© The NASDAQ OMX Group, Inc. • 2013

QUERY properties	
partitioned	false
segmented	true
answers	DA30

ANSWER properties	
transaction type	DA30
struct name	answer_user_type_info
segmented	true

3.1.59.2 Purpose

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

3.1.59.3 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
}
```

3.1.59.4 Usage and Conditions

Series

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

3.1.59.5 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
    }
}
```

3.1.59.6 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.

3.1.60 DQ33 [Currency QUERY]

3.1.60.1 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

3.1.60.2 Purpose

The purpose of this transaction is to get valid currencies.

3.1.60.3 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
}
```


© The NASDAQ OMX Group, Inc. • 2013

3.1.60.4 Usage and Conditions

Series

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

3.1.60.5 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
   }
}
```

3.1.60.6 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.

3.1.61 DQ34 [Account Type Rule QUERY]

3.1.61.1 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

© The NASDAQ OMX Group, Inc. • 2013

ANSWER properties	
transaction type	DA34
struct name	answer_account_type_rule
segmented	true

3.1.61.2 Purpose

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

3.1.61.3 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
}
```

3.1.61.4 Usage and conditions

Series

may be zeroed.

3.1.61.5 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
   }
}
```

3.1.61.6 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.

© The NASDAQ OMX Group, Inc. • 2013

3.1.62 DQ35 [Participant QUERY]

3.1.62.1 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

3.1.62.2 Purpose

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

3.1.62.3 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
}
```

3.1.62.4 Usage and conditions

Series

may be zeroed.

3.1.62.5 Answer Structure

The DA35 ANSWER has the following structure:

© The NASDAQ OMX Group, Inc. • 2013

```
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
   }
}
```

3.1.62.6 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.

3.1.63 DQ42 [Rate Index QUERY]

3.1.63.1 Fingerprint

QUERY properties	
transaction type	DQ42
calling sequence	omniapi_query_ex
struct name	query_rate_index
facility	EP0
partitioned	false
segmented	true
answers	DA42

ANSWER properties	
transaction type	DA42
struct name	answer_rate_index
segmented	true

3.1.63.2 Purpose

This query returns all available entries for current date from rate index tranche tables defined for instrument classes.

© The NASDAQ OMX Group, Inc. • 2013

3.1.63.3 Structure

The DQ42 QUERY has the following structure:

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

3.1.63.4 Usage and Conditions

Series

is of no significance here.

3.1.63.5 Answer Structure

The DA42 ANSWER has the following structure:

```
struct answer_rate_index {
   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
      struct rate_index {
         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 fixing_series  // Of type: SERIES ; Named struct no: 50000
      UINT16_T days_from_n  // DAYS_FROM_N
      UINT16_T days_to_n  // DAYS_TO_N
   }
}
```

3.1.63.6 Answer, comments

Series

holds the instrument class to which the rate is associated.

Rate Index

holds the name of the rate index, for example BBSW Ask.

Fixing Series

holds the name of the actual series for which the rate figure (price) is set (fixed). This value is set with respect to a specific time interval.

Days, from

holds the minimum number of days in the interval the fixing value is with respect to.

Days, to

holds the maximum number of days in the interval the fixing value is with respect to.

3.1.64 DQ44 [Legal Account Instrument QUERY]

3.1.64.1 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

3.1.64.2 Purpose

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

3.1.64.3 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
```


© The NASDAQ OMX Group, Inc. • 2013

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

3.1.64.4 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
   }
}
```

3.1.65 DQ45 [Trade Report Type QUERY]

3.1.65.1 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

3.1.65.2 Purpose

This query is used to retrieve all trade report types.

3.1.65.3 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
}
```

3.1.65.4 Usage and conditions

Series

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

3.1.65.5 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
      UINT8_T allow_delayed_c  // Allow delayed trade reporting
      CHAR filler_1_s  // Filler
   }
}
```

3.1.65.6 Answer, comments

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

3.1.66 DQ46 [Deal Source QUERY]

3.1.66.1 Fingerprint

QUERY properties	
transaction type	DQ46

© The NASDAQ OMX Group, Inc. • 2013

QUERY properties	
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

3.1.66.2 Purpose

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

3.1.66.3 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
}
```

3.1.66.4 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
   }
}
```

3.1.66.5 Answer, comments

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

3.1.67 DQ47 [Haircut QUERY]

3.1.67.1 Fingerprint

QUERY properties	
transaction type	DQ47
calling sequence	omniapi_query_ex
struct name	query_haircut
facility	EP0
partitioned	false
segmented	true
answers	DA47

ANSWER properties	
transaction type	DA47
struct name	answer_haircut
segmented	true

3.1.67.2 Related Messages

BU47

3.1.67.3 Purpose

This query is used to retrieve the haircut values used when valuing collaterals.

3.1.67.4 Structure

The DQ47 QUERY has the following structure:

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


© The NASDAQ OMX Group, Inc. • 2013

3.1.67.5 Usage and conditions

Series

must be zeroed (all markets) or completed as **Country Number** and **Market Code** or **Instrument Type** or **Instrument Class**.

3.1.67.6 Answer Structure

The DA47 ANSWER has the following structure:

```
struct answer_haircut {
   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[40] description_s  // Description
      UINT32_T haircut_rate_u  // Haircut Rate
      UINT32_T time_to_maturity_u  // Time to maturity
   }
}
```

3.1.67.7 Answer, comments

The answer received contains a list of haircut rates with time to maturity and the instrument class it is connected to.

Hair cut can be specified to be different depending on remaining time to maturity for a collateral instrument.

Each item in the answer is applicable for all instruments of the instrument class in the item.

The haircut rate that applies for an instrument is specified in the item where remaining time to maturity for the instrument is larger than the time to maturity in the item, but smaller than or equal to any other time to maturity specified for the same Instrument Class.

Series

specifies an **Instrument Class** (**Country Number**, **Market Code**, **Instrument Group** and **Underlying Code**).

Time to Maturity

specifies a number of months to maturity, applicable for instruments with a remaining time to maturity up to the specified value.

Hair Cut Rate

© The NASDAQ OMX Group, Inc. • 2013

specifies the value after hair cut, e.g. a figure of 80% means that the actual hair cut is 20%, and the value after hair cut is 80%.

3.1.68 DQ48 [Allowed TM Markets QUERY]

3.1.68.1 Fingerprint

QUERY properties	
transaction type	DQ48
calling sequence	omniapi_query_ex
struct name	query_allowed_tm_market
facility	EP0
partitioned	false
segmented	true
answers	DA48

ANSWER properties	
transaction type	DA48
struct name	answer_allowed_tm_market
segmented	true

3.1.68.2 Purpose

The purpose of this transaction is to receive all available TM markets.

3.1.68.3 Structure

The DQ48 QUERY has the following structure:

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

3.1.68.4 Answer Structure

The DA48 ANSWER has the following structure:

```
struct answer_allowed_tm_market {
   struct transaction_type
   UINT16_T segment_number_n  // Segment Number
   UINT16_T items_n  // Items
```


© The NASDAQ OMX Group, Inc. • 2013

```
   Array ITEM [max no: 100] {
      UINT8_T country_c  // Country Number
      UINT8_T market_c  // Market Code
      UINT16_T commodity_n  // Commodity Code
      char[12] csd_id_s  // CSD, Identity
      char[2] filler_2_s  // Filler
   }
}
```

3.1.68.5 Answer, comments

The answer received contains a list of all available legal TM markets.

3.1.69 DQ49 [Clearance System QUERY]

3.1.69.1 Fingerprint

QUERY properties	
transaction type	DQ49
calling sequence	omniapi_query_ex
struct name	query_clearance_system
facility	EP0
partitioned	false
segmented	true
answers	DA49

ANSWER properties	
transaction type	DA49
struct name	answer_clearance_system
segmented	true

3.1.69.2 Purpose

The purpose of this transaction is to retrieve the system where the series is cleared.

3.1.69.3 Structure

The DQ49 QUERY has the following structure:

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

3.1.69.4 Answer Structure

The DA49 ANSWER has the following structure:

```
struct answer_clearance_system {
   struct transaction_type
   UINT16_T segment_number_n  // Segment Number
   UINT16_T items_n  // Items
   Array ITEM [max no: 100] {
      char[12] csd_id_s  // CSD, Identity
      char[32] name_s  // Name
      char[5] ntd_id_s  // Non-trading Days, Identity
      CHAR filler_1_s  // Filler
   }
}
```

3.1.69.5 Answer, comments

The answer contains a list of all available clearance systems.

3.1.70 DQ50 [Non-Settlement Days QUERY]

3.1.70.1 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

3.1.70.2 Related Messages

BU50

© The NASDAQ OMX Group, Inc. • 2013

3.1.70.3 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.

3.1.70.4 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
}
```

3.1.70.5 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
   }
}
```

3.1.70.6 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.

3.1.71 DQ53 [Corporate Action QUERY]

3.1.71.1 Fingerprint

QUERY properties	
transaction type	DQ53

QUERY properties	
calling sequence	omniapi_query_ex
struct name	query_corp_action
facility	EP0
partitioned	false
segmented	true
answers	DA53

ANSWER properties	
transaction type	DA53
struct name	answer_corp_action_da53
segmented	true

3.1.71.2 Purpose

The purpose of this transaction is to get all active Corporate Actions that exists for a Linked, an Underlying, an Instrument Class or an Instrument Series.

3.1.71.3 Structure

The DQ53 QUERY has the following structure:

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

3.1.71.4 Usage and conditions

Series

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

3.1.71.5 Answer Structure

The DA53 ANSWER has the following structure:

```
struct answer_corp_action_da53 {
   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
```


© The NASDAQ OMX Group, Inc. • 2013

```
        char[2] corp_action_code_s  // Code, Corporate Action
        UINT8_T corp_action_type_c  // Corporate Action Type
        UINT8_T corp_action_status_c  // Status, Corporate Action
        UINT8_T corp_action_level_c  // Level, Corporate Action
        char[3] filler_3_s  // Filler
    }
}
```

3.1.71.6 Answer, comments

The answer received contains a list of Linked Underlying, Underlying, Instrument Class or Instrument Series and its associated code. Level, Corporate Action in the answer indicates on which level the code is assigned.

Each Linked Underlying, Underlying, Instrument Class or Instrument Series could have several entries in the answer, depending of how many Codes it has assigned.

An Instrument Series always inherits all codes assigned on a higher level (Linked Underlying, Underlying or Instrument Class). This means that for one Instrument Series the same code can be assigned several times. The reason for that is that the same code is possible to assign for, for example, both the underlying and the connected instrument series.

Series

is filled in with different information depending on the level the Corporate Action is assigned to:

Linked Underlying level: Only a value in Commodity Code is filled in, the rest of the fields are zero. All underlying(s) connected to this linked underlying is affected. The Linked Underlying is distributed in the answer of DQ10, Query Instrument Class.

Underlying level: Only a value in Commodity Code is filled in, the rest of the fields are zero.

Instrument Class level: A value in Country, Market, Instrument Group and Commodity Code.

Instrument Series level: A complete series definition.

Status, Corporate Action

Each entry has a status assigned with either enabled or disabled, where disabled means that the actual code no longer is active.

Level, Corporate Action

indicates on which level the code is assigned.

> **Note:** An equity series contains the same information in series as the connected instrument class. However, one equity Instrument Class can only have one connected Equity instrument series.

3.1.72 DQ54 [Valid Sector Codes QUERY]

3.1.72.1 Fingerprint

QUERY properties	
transaction type	DQ54

QUERY properties	
calling sequence	omniapi_query_ex
struct name	query_valid_sector_code
facility	EP0
partitioned	false
segmented	true
answers	DA54

ANSWER properties	
transaction type	DA54
struct name	answer_valid_sector_code
segmented	true

3.1.72.2 Purpose

Underlyings may be conected to sectors and this query retrieves all sector codes and corresponding descriptions in order to pick a suitable sector.

3.1.72.3 Structure

The DQ54 QUERY has the following structure:

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

3.1.72.4 Usage and conditions

Series

may be zeroed (all markets, that is) or completed with Country Number and Market Code, or a complete Instrument Type.

3.1.72.5 Answer Structure

The DA54 ANSWER has the following structure:

```
struct answer_valid_sector_code {
   struct transaction_type
   UINT16_T segment_number_n  // Segment Number
   UINT16_T items_n  // Items
   Array ITEM [max no: 200] {
      char[4] sector_code_s  // Sector Code
```


© The NASDAQ OMX Group, Inc. • 2013

```
        char[40] description_s  // Description
    }
}
```

3.1.72.6 Answer, comments

The answer returns a list of valid sector codes.

3.1.73 DQ57 [Member Obligation QUERY]

3.1.73.1 Fingerprint

QUERY properties	
transaction type	DQ57
calling sequence	omniapi_query_ex
struct name	query_member_obligation
facility	EP0
partitioned	false
segmented	true
answers	DA57

ANSWER properties	
transaction type	DA57
struct name	answer_member_obligation_da57
segmented	true

3.1.73.2 Purpose

The purpose of this query is to receive all Participants you have the rights to trade on behalf of.

3.1.73.3 Structure

The DQ57 QUERY has the following structure:

```
struct query_member_obligation {
   struct transaction_type
   struct series  // Named struct no: 50000
   UINT16_T segment_number_n  // Segment Number
   UINT8_T on_behalf_of_type_c  // On Behalf of Type
   CHAR filler_1_s  // Filler
}
```

© The NASDAQ OMX Group, Inc. • 2013

3.1.73.4 Answer Structure

The DA57 ANSWER has the following structure:

```
struct answer_member_obligation_da57 {
   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
      UINT16_T cst_id_n  // Customer Number
      char[2] country_id_s  // Name, Country
      char[5] ex_customer_s  // Customer, Identity
      UINT8_T right_type_c  // Right type
      char[2] filler_2_s  // Filler
   }
}
```

3.1.74 DQ76 [State Type QUERY]

3.1.74.1 Fingerprint

QUERY properties	
transaction type	DQ76
calling sequence	omniapi_query_ex
struct name	query_state_type
facility	EP0
partitioned	false
segmented	true
answers	DA76

ANSWER properties	
transaction type	DA76
struct name	answer_state_type
segmented	true

3.1.74.2 Purpose

The purpose of this transaction is to retrieve the description of all existing State Types.

The State Type Number is used when entering a Session State Order.

3.1.74.3 Structure

The DQ76 QUERY has the following structure:

© The NASDAQ OMX Group, Inc. • 2013

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

3.1.74.4 Usage and Conditions

Series

Must be zeroed.

3.1.74.5 Answer Structure

The DA76 ANSWER has the following structure:

```
struct answer_state_type {
   struct transaction_type
   UINT16_T segment_number_n  // Segment Number
   UINT16_T items_n  // Items
   Array ITEM [max no: 100] {
      UINT16_T state_type_number_n  // State Type Number
      char[20] state_type_name_s  // State Type Name
      char[32] name_s  // Name
      UINT8_T country_c  // Country Number
      UINT8_T market_c  // Market Code
   }
}
```

3.1.74.6 Answer, comments

The answer received contains a list of existing state type numbers and their names and descriptions. Each response is prefaced with the Transaction Type (DA76) and an item field specifying the number of records contained in the response.

3.1.75 DQ87 [Market Maker Protection QUERY]

3.1.75.1 Fingerprint

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

© The NASDAQ OMX Group, Inc. • 2013

QUERY properties	
answers	DA87

ANSWER properties	
transaction type	DA87
struct name	answer_mm_protection
segmented	true

3.1.75.2 Related Messages

BU87, DC87

3.1.75.3 Purpose

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

3.1.75.4 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
}
```

3.1.75.5 Usage and conditions

Series

Should be filled with 0 (zero)

3.1.75.6 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
```


© The NASDAQ OMX Group, Inc. • 2013

```
        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
    }
}
```

3.1.76 DQ88 [Turnover List QUERY]

3.1.76.1 Fingerprint

QUERY properties	
transaction type	DQ88
calling sequence	omniapi_query_ex
struct name	query_turnover_list
facility	EP0
partitioned	false
segmented	true
answers	DA88

VIA properties	
transaction type	DA88
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

3.1.76.2 Related Messages

BU88

3.1.76.3 Purpose

This query is used retrieve turnover lists. A Turnover List is an exchange official list of instrument series in a specific Market.

3.1.76.4 Structure

The DQ88 QUERY has the following structure:

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

```
}
```

3.1.76.5 Usage and conditions

Series

Is used for partitioning and should be null-filled.

3.1.76.6 Answer Structure

The DA88 VIA has the following structure:

```
struct answer_segment_hdr
Sequence {
   struct item_hdr
   Sequence {
      struct sub_item_hdr
      Choice {
         struct ns_turnover_list_base  // Named struct no: 37701
         struct ns_turnover_list_item  // Named struct no: 37702
      }
   }
}
```

3.1.76.7 Answer, comments

The answer received contains one basic structure valid for the whole list and one structure for all the instrument series within the list.

List Heading in the item list is repeated for each instrument series that belongs to this list heading.

3.1.77 DQ90 [Pre Trade Limit QUERY]

3.1.77.1 Fingerprint

QUERY properties	
transaction type	DQ90
calling sequence	omniapi_query_ex
struct name	query_pre_trade_limit
facility	EP0
partitioned	false
segmented	true
answers	DA90

VIA properties	
transaction type	DA90

© The NASDAQ OMX Group, Inc. • 2013

VIA 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.
segmented	true

3.1.77.2 Related Messages

BU90, DC90

3.1.77.3 Purpose

This query is used by the Sponsoring Participant to query for own Pre Trade Risk Groups.

3.1.77.4 Structure

The DQ90 QUERY has the following structure:

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

3.1.77.5 Usage and conditions

Series

is used for routing and should be zero-filled.

3.1.77.6 Answer Structure

The DA90 VIA has the following structure:

```
   struct answer_segment_hdr
   Sequence {
      struct item_hdr
      Sequence {
         struct sub_item_hdr
         Choice {
            struct ns_pre_trade_limit_id  // Named struct no: 37805
            struct ns_pre_trade_limit  // Named struct no: 37801
            struct ns_pre_trade_limit_user  // Named struct no: 37802
            struct ns_pre_trade_limit_not  // Named struct no: 37804
            struct ns_pre_trade_limit_param  // Named struct no: 37803
         }
      }
   }
```

© The NASDAQ OMX Group, Inc. • 2013

3.1.77.7 Answer Structure, comments

If the Pre Trade Risk Group has both current and pending items, both are returned in the answer.

All pending items have the date from when the item is active in the **Valid From Date**. The current item has blank in this field.

pre_trade_limit

is sent once for each pre trade risk group.

pre_trade_limit_user

is repeated once for every sponsored user, if any are connected to the pre trade limit risk group.

pre_trade_limit_param

is repeated once for every instrument type or instrument class that are connected to the group.

pre_trade_limit_not

is repeated once for every mail receivers, if any are connected to the pre trade risk limit group.

3.1.78 DQ92 [Strip Series QUERY]

3.1.78.1 Fingerprint

QUERY properties	
transaction type	DQ92
calling sequence	omniapi_query_ex
struct name	query_strip_series
facility	EP0
partitioned	false
segmented	true
answers	DA92

ANSWER properties	
transaction type	DA92
struct name	answer_strip_series
segmented	true

3.1.78.2 Purpose

This query is used to retrieve the relation between a specific strip series and its corresponding cleared series.

© The NASDAQ OMX Group, Inc. • 2013

3.1.78.3 Structure

The DQ92 QUERY has the following structure:

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

3.1.78.4 Usage and conditions

Strip Range

can have the values: Annual, Semi annual or Quarterly.

For an Annual strip, the delivery period is covering a whole year, etc.

3.1.78.5 Answer Structure

The DA92 ANSWER has the following structure:

```
struct answer_strip_series {
   struct transaction type
   UINT16_T segment number n  // Segment Number
   UINT16_T items n  // Items
   Array ITEM [max no: 75] {
      struct series  // Named struct no: 50000
      UINT16_T items n  // Items
      UINT8_T strip range c  // Strip range
      UINT8_T split rule c  // Split rule
      Array STRIP_SERIES [max no: 52] {
         struct series  // Named struct no: 50000
      }
   }
}
```

3.1.78.6 Answer, comments

The answer contain a list of strip series, the delivery period range it cover, number of included cleared series and a list of these series.

3.1.79 DQ120 [Delta Underlying QUERY]

3.1.79.1 Fingerprint

QUERY properties	
transaction type	DQ120

QUERY properties	
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

3.1.79.2 Related Messages

BU120

3.1.79.3 Purpose

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

3.1.79.4 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.

© The NASDAQ OMX Group, Inc. • 2013

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).

3.1.79.5 Structure

The DQ120 QUERY has the following structure:

`struct query_delta`

3.1.79.6 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

© The NASDAQ OMX Group, Inc. • 2013

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.

3.1.79.7 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
         struct ns_index_linked  // Named struct no: 37204
         struct ns_underlying_power  // Named struct no: 37206
         struct ns_underlying_ext3  // Named struct no: 37209
         struct ns_reference_rate  // Named struct no: 37210
         struct ns_index_value  // Named struct no: 37211
         struct ns_lottery_bonds  // Named struct no: 37212
         struct ns_convertibles  // Named struct no: 37213
         struct ns_derived_from  // Named struct no: 37214
      }
   }
}
```

3.1.79.8 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

© The NASDAQ OMX Group, Inc. • 2013

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.

3.1.80 DQ121 [Delta Underlying for Back Office QUERY]

3.1.80.1 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

3.1.80.2 Related Messages

BU121

3.1.80.3 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.

3.1.80.4 Structure

The DQ121 QUERY has the following structure:

```
struct query_delta
```

3.1.80.5 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**.

3.1.80.6 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
         struct ns_index_linked  // Named struct no: 37204
         struct ns_underlying_power  // Named struct no: 37206
         struct ns_underlying_ext3  // Named struct no: 37209
         struct ns_reference_rate  // Named struct no: 37210
         struct ns_index_value  // Named struct no: 37211
         struct ns_lottery_bonds  // Named struct no: 37212
         struct ns_convertibles  // Named struct no: 37213
         struct ns_derived_from  // Named struct no: 37214
      }
   }
}
```

3.1.80.7 Answer, comments

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

3.1.81 DQ122 [Delta Instrument Class QUERY]

3.1.81.1 Fingerprint

QUERY properties	
transaction type	DQ122
calling sequence	omniapi_query_ex
struct name	query_delta
facility	EP0

© The NASDAQ OMX Group, Inc. • 2013

QUERY properties	
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

3.1.81.2 Related Messages

BU122

3.1.81.3 Purpose

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

3.1.81.4 Structure

The DQ122 QUERY has the following structure:

```
struct query_delta
```

3.1.81.5 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**.

3.1.81.6 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_calc_rule  // Named struct no: 37104
                struct ns_inst_class_secur  // Named struct no: 37105
                struct ns_inst_class_leg_calc_rule  // Named struct no: 37115
                struct ns_price_tick_corr  // Named struct no: 37113
                struct ns_inst_class_trr_def_publ  // Named struct no: 37118
                struct ns_inst_class_ext6  // Named struct no: 37120
            }
        }
    }
```

3.1.81.7 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.

3.1.82 DQ123 [Delta Instrument Class for Back Office QUERY]

3.1.82.1 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

3.1.82.2 Related Messages

BU123

© The NASDAQ OMX Group, Inc. • 2013

3.1.82.3 Purpose

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

3.1.82.4 Structure

The DQ123 QUERY has the following structure:

```
struct query_delta
```

3.1.82.5 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**.

3.1.82.6 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_calc_rule  // Named struct no: 37104
         struct ns_inst_class_secur  // Named struct no: 37105
         struct ns_inst_class_cms  // Named struct no: 37114
         struct ns_inst_class_leg_calc_rule  // Named struct no: 37115
         struct ns_price_tick_corr  // Named struct no: 37113
         struct ns_inst_class_trr_def_publ  // Named struct no: 37118
         struct ns_inst_class_ext6  // Named struct no: 37120
      }
   }
}
```

3.1.82.7 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.

© The NASDAQ OMX Group, Inc. • 2013

3.1.83 DQ124 [Delta Instrument Series QUERY]

3.1.83.1 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

3.1.83.2 Related Messages

BU124

3.1.83.3 Purpose

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

3.1.83.4 Structure

The DQ124 QUERY has the following structure:

```
struct query_delta
```

3.1.83.5 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**.

© The NASDAQ OMX Group, Inc. • 2013

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.

3.1.83.6 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_power  // Named struct no: 37303
         struct ns_inst_series_repo  // Named struct no: 37304
         struct ns_inst_series_leg_flow  // Named struct no: 37309
      }
   }
}
```

3.1.83.7 Answer, comments

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

3.1.84 DQ125 [Delta Instrument Series for Back Office QUERY]

3.1.84.1 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

VIA 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.
segmented	true

3.1.84.2 Related Messages

BU125

3.1.84.3 Purpose

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

3.1.84.4 Structure

The DQ125 QUERY has the following structure:

```
struct query_delta
```

3.1.84.5 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.

3.1.84.6 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_power  // Named struct no: 37303
         struct ns_inst_series_repo  // Named struct no: 37304
         struct ns_inst_series_bo  // Named struct no: 37306
```


© The NASDAQ OMX Group, Inc. • 2013

```
            struct ns_inst_series_leg_flow  // Named struct no: 37309
            struct ns_inst_series_ext5  // Named struct no: 37313
        }
    }
}
```

3.1.84.7 Answer, comments

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

3.1.85 DQ126 [Combo Series QUERY]

3.1.85.1 Fingerprint

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

3.1.85.2 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)

3.1.85.3 Purpose

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

3.1.85.4 Structure

The DQ126 QUERY has the following structure:

© The NASDAQ OMX Group, Inc. • 2013

```
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
}
```

3.1.85.5 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**.

3.1.85.6 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
         }
      }
   }
```

3.1.86 DQ131 [Instrument Type for Back Office QUERY]

3.1.86.1 Fingerprint

QUERY properties	
transaction type	DQ131
calling sequence	omniapi_query_ex
struct name	query_instrument
facility	EP0
partitioned	false
segmented	true
answers	DA131

© The NASDAQ OMX Group, Inc. • 2013

VIA properties	
transaction type	DA131
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

3.1.86.2 Purpose

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

3.1.86.3 Structure

The DQ131 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
}
```

3.1.86.4 Usage and conditions

Series

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

3.1.86.5 Answer Structure

The DA131 VIA has the following structure:

```
   struct answer_segment_hdr
   struct item_hdr
   struct sub_item_hdr
   Sequence {
      struct item_hdr
      Sequence {
         struct sub_item_hdr
         Choice {
            struct ns_inst_type_basic  // Named struct no: 37601
            struct ns_inst_type_secur  // Named struct no: 37602
         }
      }
   }
```

3.1.86.6 Answer, comments

The answer received contains a list of types. Each response is prefaced with the transaction type (DA131) and an item field specifying the number of records contained in the response. The answer uses the VIM concept.

3.1.87 DQ132 [Valuation Group QUERY]

3.1.87.1 Fingerprint

QUERY properties	
transaction type	DQ132
calling sequence	omniapi_query_ex
struct name	query_valuation_group
facility	EP0
partitioned	false
segmented	true
answers	DA132

ANSWER properties	
transaction type	DA132
struct name	answer_valuation_group
segmented	true

3.1.87.2 Purpose

The purpose of this transaction is to retrieve information on collateral limits per valuation group.

3.1.87.3 Structure

The DQ132 QUERY has the following structure:

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

3.1.87.4 Usage and conditions

Series

© The NASDAQ OMX Group, Inc. • 2013

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

3.1.87.5 Answer Structure

The DA132 ANSWER has the following structure:

```
struct answer_valuation_group {
   struct transaction_type
   UINT16_T segment_number_n  // Segment Number
   UINT16_T items_n  // Items
   Array ITEM [max no: 100] {
      INT32_T vag_limit_i  // Valuation Group Limit (%)
      char[12] valuation_group_id_s  // Valuation Group Identity ; Of type:
VAG_ID_S
      char[40] description_s  // Description
   }
}
```

3.1.88 DQ134 [Account Type QUERY]

3.1.88.1 Fingerprint

QUERY properties	
transaction type	DQ134
calling sequence	omniapi_query_ex
struct name	query_account_type
facility	EP0
partitioned	false
segmented	true
answers	DA134

VIA properties	
transaction type	DA134
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

3.1.88.2 Purpose

This query is used to retrieve all account types in the system.

© The NASDAQ OMX Group, Inc. • 2013

3.1.88.3 Structure

The DQ134 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
}
```

3.1.88.4 Usage and Conditions

Series

should be zero filled.

3.1.88.5 Answer Structure

The DA134 VIA has the following structure:

```
   struct answer_segment_hdr
   Sequence {
      struct item_hdr
      Sequence {
         struct sub_item_hdr
         Choice {
            struct ns_account_type_basic  // Named struct no: 37901
         }
      }
   }
```

3.1.88.6 Answer, comments

The answer received contains a list of types. Each response is prefaced with the transaction type (DA134) and an item field specifying the number of records contained in the response. The answer uses the VIM concept.

3.1.89 DQ135 [Market Maker Obligations QUERY]

3.1.89.1 Fingerprint

QUERY properties	
transaction type	DQ135
calling sequence	omniapi_query_ex
struct name	query_market_maker_obl
facility	EP0

© The NASDAQ OMX Group, Inc. • 2013

QUERY properties	
partitioned	false
segmented	true
answers	DA135

VIA properties	
transaction type	DA135
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

3.1.89.2 Purpose

This query is used to retrieve information about a market maker obligations.

3.1.89.3 Structure

The DQ135 QUERY has the following structure:

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

3.1.89.4 Usage and Conditions

Series

should be filled with 0 (zero).

3.1.89.5 Structure Contents

The DQ135 query has the following structure:

```
typedef struct query_market_maker_obl
{
   transaction_type_t transaction_type;
   series_t series;
   uint16_t segment_number_n;
   char filler_2_s [2];
} query_market_maker_obl_t;
```

© The NASDAQ OMX Group, Inc. • 2013

3.1.89.6 Answer Structure

The DA135 VIA has the following structure:

```
struct answer_segment_hdr
Sequence {
   struct item_hdr
   Sequence {
      struct sub_item_hdr
      Choice {
         struct ns_price_quote_resp  // Named struct no: 37951
         struct ns_vld_max_spread  // Named struct no: 37952
         struct ns_price_quote_criteria  // Named struct no: 37953
      }
   }
}
```

3.1.89.7 Answer, comments

The struct **`ns_vld_max_spread`** contains all unique Max Spreads that are referenced from struct **`ns_price_quote_criteria`**.

3.1.89.8 Answer, Structure Contents

The DA135 VIA has the following structure:

```
struct answer_segment_hdr
Sequence {
   struct item_hdr
   Sequence {
      struct sub_item_hdr
      Choice {
         struct ns_price_quote_resp     // Named struct no: 37951
         struct ns_vld_max_spread       // Named struct no: 37952
         struct ns_price_quote_criteria // Named struct no: 37953
      }
   }
}
```

3.2 Order Management

3.2.1 BO5 [Firm Order Book VIB]

3.2.1.1 Fingerprint

VIB properties	
transaction type	BO5
calling sequence	omniapi_read_event_ext_ex or omniapi_read_event_block

© The NASDAQ OMX Group, Inc. • 2013

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 dedicated
segmented	true

3.2.1.2 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.

3.2.1.3 Structure

The BO5 VIB has the following structure:

```
struct broadcast_hdr
Sequence {
   struct sub_item_hdr
   Choice {
      struct block_price_trans  // Named struct no: 34007
      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 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 order_price_change  // Named struct no: 34905
      struct order_return_info  // Named struct no: 34904
      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 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 order_info  // Named struct no: 34917
      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 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
   }
```

```
}
```

3.2.1.4 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.

3.2.1.5 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 NASDAQ OMX Group, Inc. • 2013

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.

3.2.2 BO55 [Trade Report Notification VIB]

3.2.2.1 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

3.2.2.2 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.

3.2.2.3 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
   }
}
```

3.2.2.4 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.

3.2.3 BO61 [Issuer Order Book Changes BROADCAST]

3.2.3.1 Fingerprint

BROADCAST properties	
transaction type	BO61
calling sequence	omniapi_read_event_ext_ex or omniapi_read_event_block
struct name	ob_changes_id
info type	instrument class

3.2.3.2 Related Messages

MQ67

3.2.3.3 Purpose

BO61 is a market by order broadcast specifically used by the issuer during an issuing auction.

3.2.3.4 Structure

The BO61 BROADCAST has the following structure:

```
struct ob_changes_id {
   struct broadcast_type
   struct changes
   QUAD_WORD order_number_u  // Order Number
   struct order_no_id
   struct party
}
```


© The NASDAQ OMX Group, Inc. • 2013

3.2.3.5 Usage and conditions

If the trader identity is not public information, party is blanked.

To obtain an Order Book mirror copy, all broadcasts should be stored until the query is completed. When the sequence number is higher than the sequence number for this series in the answer, the broadcast must be taken care of.

An Order Book change is either ADD, DELETE or ALTER. This is specified in the Order Book Command.

Information for an Order Book command equal to ADD should be interpreted as follows:

- Sequence Number is a consecutive number per series.
- Quantity difference is equal to the Quantity field for an ADD operation.

Information for an Order Book command equal to DELETE is to be interpreted as follows:

- The deleted order is identified by the position (position in the Order Book) held in the Order Book and by the order number. Remaining fields contain redundant information.

Information for an Order Book command equal to ALTER is to be interpreted as follows:

- The order that has changed (that is, the content has changed but the position in the order book remains) is defined by both order position and order number.
- Quantity difference is the difference between old and new quantity, if the quantity field is changed. (Quantity difference = new quantity - old quantity.)
- The fields that follow contain the values of the order after the alteration has taken place regardless of which field has been changed.

3.2.4 BO98 [Indicative Quote Changes VIB]

3.2.4.1 Fingerprint

VIB properties	
transaction type	BO98
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

3.2.4.2 Purpose

This broadcast disseminates the participants own indicative quotes.

3.2.4.3 Structure

The BO98 VIB has the following structure:

© The NASDAQ OMX Group, Inc. • 2013

```
struct broadcast_hdr
Sequence {
   struct sub_item_hdr
   Choice {
      struct indicative_quote_base  // Named struct no: 34026
      struct indicative_quote_fixed_income  // Named struct no: 34027
   }
}
```

3.2.5 BO99 [Block Transaction Response BROADCAST]

3.2.5.1 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

3.2.5.2 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 and MO96.

3.2.5.3 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
   }
}
```

3.2.5.4 Usage and Conditions

The BO99 is similar to the answer response used in MO30/MO414.

© The NASDAQ OMX Group, Inc. • 2013

Note:
The BO99 is only sent for failed MO36/MO96 items and 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
Transaction aborted	GEN_CDC_INT_CLOSED – Instrument Type is not open for this Transaction Type
Transaction aborted	...

3.2.6 MO2 [Multi Leg Order Entry TRANSACTION]

3.2.6.1 Fingerprint

TRANSACTION properties	
transaction type	MO2
calling sequence	omniapi_tx_ex
struct name	multi_leg_order_insert
facility	EP0
partitioned	true

3.2.6.2 Purpose

The multi order (non-standard combination) transaction support that the price of the combination is given as an average price of all legs in the combination, including adjustment for differences in contract size between the legs (Contract Weighted Average Price).

This presupposes that several orders are sent in the same transaction and that all orders are to be closed (fill or kill) – if this is not the case, the whole transaction will be discarded.

3.2.6.3 Structure

The MO2 TRANSACTION has the following structure:

```
struct multi_leg_order_insert  // Named struct no: 34817
```

3.2.6.4 Usage and Conditions

multi_leg_order_insert:

Price type

The price type of the combination (multi_leg_price_type_c) shall be set to 5 (contract weighted average price). The price calculation will then be defined by:

$$P_{combination_{AVERAGE}} = \frac{P_{leg_1} r_{leg_1} c_{leg_1} + P_{leg_2} r_{leg_2} c_{leg_2} + \cdots + P_{leg_n} r_{leg_n} c_{leg_n}}{r_{leg_1} c_{leg_1} + r_{leg_2} c_{leg_2} + \cdots + r_{leg_n} c_{leg_n}}$$

where P = price, r = ratio, c = contract size.

Order type

Order type (order_type_c) shall be set to 1 (limit order).

multi_leg_order_insert_item:

- The maximum number of legs is 5.
- All legs of the combination have to be on the same side, i.e. buying the combination means buying all legs, selling the combination means selling all the legs.
- Premium in the legs shall be set to 0 (zero) or INT_MIN.
- Quantity method in the legs (calculate_quantity_method_c) is not used and shall be set to 0 (zero).
- The following parameters have to be the same for all legs in the transaction:
  - The number of decimals in price
  - Premium Unit
  - Base Currency
  - Traded Currency Unit
  - Price unit = ”price”

3.2.6.5 Return Codes

After a successful MO2 transaction, an order number and information regarding the state of the order will be returned to the sender.

Cstatus	Txstat	ordidt
Successful	1 – no part of the order placed in the Order book and no part closed	order number
Successful	2 – the whole order closed	order number
Transaction aborted	GEN_CDC_INT_CLOSED – Instrument Type is not open for this Transaction Type	-
Transaction aborted	...	-

© The NASDAQ OMX Group, Inc. • 2013

An MO2 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.

3.2.7 MO4 [Order Deletion TRANSACTION]

3.2.7.1 Fingerprint

TRANSACTION properties	
transaction type	MO4
calling sequence	omniapi_tx_ex
struct name	delete_trans
facility	EP0
partitioned	true

3.2.7.2 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.

3.2.7.3 Structure

The MO4 TRANSACTION has the following structure:

```
struct delete_trans  // Named struct no: 34011
```

3.2.7.4 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.

© The NASDAQ OMX Group, Inc. • 2013

3.2.7.5 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.

3.2.8 MO31 [Order Entry TRANSACTION]

3.2.8.1 Fingerprint

TRANSACTION properties	
transaction type	MO31
calling sequence	omniapi_tx_ex
struct name	hv_order_trans
facility	EP0
partitioned	true

3.2.8.2 Purpose

This transaction is used for placing orders in the Order book.

3.2.8.3 Structure

The MO31 TRANSACTION has the following structure:

```
struct hv_order_trans  // Named struct no: 34005
```

3.2.8.4 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.

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.

Total Size Volume cannot be greater than 32766.

Quantity can not be greater than Total Size Volume.

Validity Time

can be set to a certain value or to zero. Please see the Detailed Field Information chapter for details.

© The NASDAQ OMX Group, Inc. • 2013

The Exchange defines a minimum Validity Time for an order.

Trade Report Type

is in this context used to define in which session states the order will be active. Only applicable for Session State Orders.

Example 1:

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 2:

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 3:

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).

© The NASDAQ OMX Group, Inc. • 2013

Example 4:

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.

3.2.8.5 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	1 – no part of the order placed in the Order book and no part closed	order number
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.

© The NASDAQ OMX Group, Inc. • 2013

3.2.9 MO33 [Alteration TRANSACTION]

3.2.9.1 Fingerprint

TRANSACTION properties	
transaction type	MO33
calling sequence	omniapi_tx_ex
struct name	hv_alter_trans
facility	EP0
partitioned	true

3.2.9.2 Purpose

MO33 is used to alter an order in the order book.

3.2.9.3 Structure

The MO33 TRANSACTION has the following structure:

```
struct hv_alter_trans  // Named struct no: 34010
```

3.2.9.4 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:

© The NASDAQ OMX Group, Inc. • 2013

- **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.

Note that MO33 cannot be used for altering an MO75 order that has been placed in the order book. To perform an alteration of MO75, perform a delete order transaction first and then send a new MO75 transaction order to the backend.

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

© The NASDAQ OMX Group, Inc. • 2013

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 =1000 total volume_i = 1800	delta_quantity_c = 2 mp_quantity_i = 600 total volume_i = 0	mp_quantity_i = 1600 total volume_i = 1800
mp_quantity_i =1000 total volume_i = 1000	delta_quantity_c = 2 mp_quantity_i = -600 total volume_i = 0	mp_quantity_i = 400 total volume_i = 1000
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.**

3.2.9.5 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	ordidt
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.

3.2.10 MO36 [Two-Sided Price Quotation Block TRANSACTION]

3.2.10.1 Fingerprint

TRANSACTION properties	
transaction type	MO36
calling sequence	omniapi_tx_ex
struct name	block_price_trans
facility	EP0
partitioned	true

© The NASDAQ OMX Group, Inc. • 2013

3.2.10.2 Purpose

This transaction is used for placing up to configurable maximum number of two-sided quotations in the Order book.

3.2.10.3 Structure

The MO36 TRANSACTION has the following structure:

```
struct block_price_trans  // Named struct no: 34007
```

3.2.10.4 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.

© The NASDAQ OMX Group, Inc. • 2013

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.

3.2.10.5 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.

© The NASDAQ OMX Group, Inc. • 2013

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.

3.2.11 MO37 [Two-Sided Price Quotation TRANSACTION]

3.2.11.1 Fingerprint

TRANSACTION properties	
transaction type	MO37
calling sequence	omniapi_tx_ex
struct name	hv_price_2_trans
facility	EP0
partitioned	true

3.2.11.2 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.

3.2.11.3 Structure

The MO37 TRANSACTION has the following structure:

```
struct hv_price_2_trans  // Named struct no: 34001
```

3.2.11.4 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

© The NASDAQ OMX Group, Inc. • 2013

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 participant)	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 participant)	5	-	10	-	-
Order six	-	10	8	-	-

© The NASDAQ OMX Group, Inc. • 2013

Order	Ask Size	Bid Size	Premium	Ask Total Size	Bid Total Size
(from the same partici-pant)					

3.2.11.5 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
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

© The NASDAQ OMX Group, Inc. • 2013

cstatus	txstat		ordidt
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.

3.2.12 MO40 [Inactive Deletion TRANSACTION]

3.2.12.1 Fingerprint

TRANSACTION properties	
transaction type	MO40
calling sequence	omniapi_tx_ex
struct name	delete_trans
facility	EP0
partitioned	true

3.2.12.2 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.

© The NASDAQ OMX Group, Inc. • 2013

3.2.12.3 Structure

The MO40 TRANSACTION has the following structure:

```
struct delete_trans  // Named struct no: 34011
```

3.2.12.4 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
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**

© The NASDAQ OMX Group, Inc. • 2013

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.

3.2.12.5 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 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.

3.2.13 MO41 [External Stop Order TRANSACTION]

3.2.13.1 Fingerprint

TRANSACTION properties	
transaction type	MO41
calling sequence	omniapi_tx_ex

© The NASDAQ OMX Group, Inc. • 2013

TRANSACTION properties	
struct name	stop_order_trans
facility	EP0
partitioned	false

3.2.13.2 Purpose

This transaction is used for placing stop (stop-loss) orders in the order book.

3.2.13.3 Structure

The MO41 TRANSACTION has the following structure:

```
struct stop_order_trans  // Named struct no: 34017
```

3.2.13.4 Usage and Conditions

Client

Character "*" and "%" are **not** allowed in the **Client** field.

The rest of the fields should be completed as defined in the MO31 transaction except for the fields described below. These fields defines the order when the stop order has been converted to a normal order.

Stop Condition

defines what trigger mechanism to use for the stop order.

Stop Series

is the series that will be used for checking the stop condition.

Premium, Limit

is the stop price that is compared to the price defined by the stop condition.

Example

The order book for series A contains an ask order: Quantity 10, Premium 20.

The order book for series B is empty.

An incoming stop order enters the system with the following parameters:

Series = series B

Premium = 30

Quantity = 50

Bid or Ask = Ask

Stop condition = 4 (ask price <= stop price)

Stop series = series A

© The NASDAQ OMX Group, Inc. • 2013

Premium, Limit = 18

The stop order will not be activated, since the ask price for series A is larger than the stop price.

An incoming ask order (MO31) enters for series A with premium 15. The stop order will be activated, since the ask price of series A is less than the stop price. The order book for series B will then contain the following order:

Quantity 50, Premium 30

When the stop order is activated, a broadcast (BO5) will be sent to the user that entered the stop order. This will contain the information on the status of the order when it was activated.

Note:

Only order_type = 1, 2 and 3 are valid for stop order.

Note:

When a stop order condition is triggered the order is converted to a normal order. This means that after a stop order condition is triggered, the normal order modification and order deletion transactions should be used instead of the stop order equivalents.

3.2.13.5 Return codes

Cstatus	Txstat	Ordidt
Successful	4 - The whole order placed in the order book.	Order number
Transaction aborted	GEN_CDC_INT_CLOSED - Instrument Type is not open for this Transaction Type.	-
Transaction Aborted	...	

After a successful MO41 transaction, an order number and information regarding the state of the order will be returned to the sender.

An MO41 transaction may also be aborted by the Marketplace, in which case only the reason for the transaction being aborted is returned to the sender.

3.2.14 MO43 [External Alter Stop Order TRANSACTION]

3.2.14.1 Fingerprint

TRANSACTION properties	
transaction type	MO43
calling sequence	omniapi_tx_ex
struct name	hv_alter_trans
facility	EP0
partitioned	false

© The NASDAQ OMX Group, Inc. • 2013

3.2.14.2 Purpose

This transaction is used to alter a stop order.

3.2.14.3 Structure

The MO43 TRANSACTION has the following structure:

```
struct hv_alter_trans  // Named struct no: 34010
```

3.2.14.4 Usage and conditions

Only one existing stop order can be altered at a time. A unique order number refers to that order. Both the order number and the transaction header must be stated.

The fields can be altered in the same way as in the MO33 transaction with the following exceptions:

Stop Condition

cannot be altered.

Block Size

can be altered.

Delta Quantity

must be filled with 1.

The fields that should be altered must be completed. The remaining fields are set to zero.

3.2.14.5 Return codes

After a successful MO43 transaction, the number of contracts before the stop order is changed, or zero if no order exists, is returned to the sender. Not finding a stop order to alter is considered to be a successful operation.

3.2.15 MO44 [External Delete Stop Order TRANSACTION]

3.2.15.1 Fingerprint

TRANSACTION properties	
transaction type	MO44
calling sequence	omniapi_tx_ex
struct name	delete_trans
facility	EP0
partitioned	true

© The NASDAQ OMX Group, Inc. • 2013

3.2.15.2 Purpose

This transaction is used to remove one or more stop orders from the order book. In contrast to the alter stop order transaction, this transaction can affect several stop orders at once – a group of stop orders to be deleted can be specified.

3.2.15.3 Structure

The MO44 TRANSACTION has the following structure:

```
struct delete_trans  // Named struct no: 34011
```

3.2.15.4 Usage and conditions

This transaction behaves exactly like the MO4 delete transaction. For details, refer to the MO4 section.

3.2.15.5 Return codes

Cstatus	Txstat	Ordidt
Successful	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	n – number of contracts before deletion (for specific order deletion only, the whole Instrument [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	...	-

After a successful MO44 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.

An MO44 transaction may also be aborted by the Marketplace, in which case only the reason for the transaction being aborted is returned to the sender.

3.2.16 MO74 [Trade Report Deletion, Unmatched TRANSACTION]

3.2.16.1 Fingerprint

TRANSACTION properties	
transaction type	MO74
calling sequence	omniapi_tx_ex
struct name	delete_trans

© The NASDAQ OMX Group, Inc. • 2013

TRANSACTION properties	
facility	EP0
partitioned	false

3.2.16.2 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).

3.2.16.3 Structure

The MO74 TRANSACTION has the following structure:

```
struct delete_trans  // Named struct no: 34011
```

3.2.16.4 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.

3.2.16.5 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.

© The NASDAQ OMX Group, Inc. • 2013

3.2.17 MO75 [Trade Report TRANSACTION]

3.2.17.1 Fingerprint

TRANSACTION properties	
transaction type	MO75
calling sequence	omniapi_tx_ex
struct name	trade_report_1_trans
facility	EP0
partitioned	true

3.2.17.2 Related Messages

MO76 is the two-sided version.

DQ45

3.2.17.3 Purpose

This transaction is used to send orders that have already led to closings outside the Exchange.

3.2.17.4 Structure

The MO75 TRANSACTION has the following structure:

```
struct trade_report_1_trans  // Named struct no: 34021
```

3.2.17.5 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

Deferred Publication

© The NASDAQ OMX Group, Inc. • 2013

The central system will accept the transaction even if the volume is to low to allow the publication to be deferred.

Party

> **Note:**
> All character fields must be space padded up to the total length of the field.

3.2.17.6 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.

3.2.18 MO76 [Trade Report, Two-Sided TRANSACTION]

3.2.18.1 Fingerprint

TRANSACTION properties	
transaction type	MO76
calling sequence	omniapi_tx_ex
struct name	trade_report_2_trans
facility	EP0
partitioned	true

3.2.18.2 Related Messages

MO75 is the single-sided version.

DQ45

3.2.18.3 Purpose

This transaction is used to send orders on behalf of two participants that have already closed a deal outside the Exchange.

3.2.18.4 Structure

The MO76 TRANSACTION has the following structure:

```
struct trade_report_2_trans  // Named struct no: 34022
```

3.2.18.5 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

Deferred Publication

The central system will accept the transaction even if the volume is to low to allow the publication to be deferred.

3.2.19 MO77 [Combination Trade Report TRANSACTION]

3.2.19.1 Fingerprint

TRANSACTION properties	
transaction type	MO77
calling sequence	omniapi_tx_ex
struct name	combo_trade_report_trans
facility	EP0
partitioned	true

© The NASDAQ OMX Group, Inc. • 2013

3.2.19.2 Related Messages

DQ45

3.2.19.3 Purpose

This transaction is used by clients to enter combination trade reports containing up to 6 legs.

3.2.19.4 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
   }
}
```

3.2.19.5 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

Deferred Publication

The central system will accept the transaction even if the volume is to low to allow the publication to be deferred.

3.2.20 MO90 [Linked Order VIT]

3.2.20.1 Fingerprint

VIT properties	
transaction type	MO90
calling sequence	omniapi_tx_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

3.2.20.2 Related Messages

MO100, MO474, MO484

3.2.20.3 Purpose

This transaction is used to insert new linked orders.

Genium INET Trading offers linked orders to support trading strategies of choosing to trade either “one or the other” of two instruments. Linked orders are especially useful when trading bonds, where several very similar bonds might be available and a participant wants to buy one, but doesn’t care which.

Every match made on a linked order results in decrementing a proportional quantity in all its linked legs. Genium INET Trading guarantees that the overall maximum quantity will never be exceeded.

Entering Linked Orders requires specifying the order book of each leg including the side, price, quantity and any allowable order conditions. The sides, quantities and prices can differ. If the AON quantity condition is used, then it must be used on all the legs.

On entry, and using the sequence in which the legs are specified on the order, each leg will be checked to see if it is immediately executable.

3.2.20.4 Structure

The MO90 VIT has the following structure:

```
struct linked_order_insert {
   struct transaction type
   struct series  // Named struct no: 50000
   UINT16_T items_n  // Items
   UINT16_T size_n  // Size
}
Sequence {
```


© The NASDAQ OMX Group, Inc. • 2013

```
    struct sub_item_hdr
    Choice {
       struct linked_order_leg  // Named struct no: 34803
       struct exchange_info  // Named struct no: 50004
       struct free_text  // Named struct no: 34801
       struct clearing_info  // Named struct no: 34802
       struct time_in_force  // Named struct no: 34807
    }
}
```

3.2.20.5 Usage and Conditions

Linked Orders are canceled by using the normal Order Deletion transaction, specifying the Linked Order Number, or any of the Leg Order Numbers.

BO5 broadcasts is sent for linked orders.

Items

is the number of structs in this linked order. Substructs could be either extra conditions to the order as a whole, or specific legs of the order.

At least one **Linked Order Leg** must be submitted per order. Legs cannot be added or removed to an existing linked set – the entire set can be canceled and re-entered to accomplish this change.

3.2.20.6 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.

3.2.21 MO96 [Mass Quote Transaction TRANSACTION]

3.2.21.1 Fingerprint

TRANSACTION properties	
transaction type	MO96
calling sequence	omniapi_tx_ex
struct name	mass_quote_trans
facility	EP0
partitioned	true

3.2.21.2 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.

© The NASDAQ OMX Group, Inc. • 2013

3.2.21.3 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
      INT32_T buy_price_i  // Buy Price
      INT64_T buy_quantity_u  // Buy Quantity
      INT32_T sell_price_i  // Ask Price
      INT64_T sell_quantity_u  // Sell Quantity
   }
}
```

3.2.21.4 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.

3.2.21.5 Return Codes

After a successful MO96 transaction, the number of successful quotes will be returned to the sender.

If at least one quote in the mass quote is rejected, the Dedicated Block Transaction Response Broadcast (BO99) is returned and lets you know which quotes failed as well as their corresponding error code(s).

> **Note:**
>
> If all quotes in the mass quote are rejected, the BO99 is not sent.

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.

© The NASDAQ OMX Group, Inc. • 2013

3.2.22 MO97 [Indicative Quote VIT]

3.2.22.1 Fingerprint

VIT properties	
transaction type	MO97
calling sequence	omniapi_tx_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

3.2.22.2 Purpose

This transaction is used to insert indicative quotes.

3.2.22.3 Structure

The MO97 VIT has the following structure:

```
struct order_trans_hdr
Sequence {
   struct item_hdr
   Sequence {
      struct sub_item_hdr
      Choice {
         struct indicative_quote  // Named struct no: 34025
      }
   }
}
```

3.2.22.4 Usage and Conditions

Series

must be completed for MO97 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.

Items

© The NASDAQ OMX Group, Inc. • 2013

Maximum number of allowed items for one transaction is 30.

3.2.22.5 Structure Contents

Some structures in the transaction require additional explanations.

Indicative Quote

Field usage in this structure:

Buy/Ask Price can be given as one of two values: A positive value or zero means the Buy/Ask price. To indicate an undisclosed price, bit 31 should be set (the highest bit, MIN_INT) while all other bits are set to zero.

Buy/Ask Quantity can be given as one of two values: A positive number means Quoted quantity. Zero means that the quantity is undisclosed.

> **Note:**
> BO5 for canceled indicative quotes consists of one BO5 per item with the input message as one sub-item, IndicativeQuote, and each canceled part (buy/sell) as a sub-item, OrderChangeSeparate.

3.2.23 MO100 [Alter Linked Order VIT]

3.2.23.1 Fingerprint

VIT properties	
transaction type	MO100
calling sequence	omniapi_tx_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

3.2.23.2 Related Messages

MO90, MO474, MO484

3.2.23.3 Purpose

This transaction is used to update linked orders.

3.2.23.4 Structure

The MO100 VIT has the following structure:

© The NASDAQ OMX Group, Inc. • 2013

```
struct linked_order_update {
   struct transaction_type
   struct series  // Named struct no: 50000
   QUAD_WORD order_number_u  // Order Number
   UINT16_T items_n  // Items
   UINT16_T size_n  // Size
}
Sequence {
   struct sub_item_hdr
   Choice {
      struct linked_order_leg  // Named struct no: 34803
      struct exchange_info  // Named struct no: 50004
      struct free_text  // Named struct no: 34801
      struct clearing_info  // Named struct no: 34802
      struct time_in_force  // Named struct no: 34807
   }
}
```

3.2.23.5 Usage and Conditions

Linked Orders are canceled by using the normal Order Deletion transaction, specifying the Linked Order Number, or any of the Leg Order Numbers.

BO5 broadcasts is sent for linked orders.

Items

is the number of structs in this linked order. Substructs could be either extra conditions to the order as a whole, or specific legs of the order.

The same number of **Linked Order Legs** as the original order must be submitted. Legs cannot be added or removed to an existing linked set – the entire set can be canceled and re-entered to accomplish this change.

3.2.23.6 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.

3.2.24 MO388 [Proxy delete order TRANSACTION]

3.2.24.1 Fingerprint

TRANSACTION properties	
transaction type	MO388
calling sequence	omniapi_tx_ex
struct name	delete_trans_p
facility	EP0
partitioned	true

3.2.24.2 Related Messages

This is a proxy transaction for MO4.

3.2.24.3 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.

3.2.24.4 Structure

The MO388 TRANSACTION has the following structure:

```
struct delete_trans_p  // Named struct no: 34111
```

3.2.24.5 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.

3.2.24.6 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.

3.2.25 MO415 [MO31 With Trader ID TRANSACTION]

3.2.25.1 Fingerprint

TRANSACTION properties	
transaction type	MO415
calling sequence	omniapi_tx_ex
struct name	hv_order_trans_p
facility	EP0
partitioned	true

3.2.25.2 Related Messages

This is a proxy transaction for MO31.

3.2.25.3 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.

© The NASDAQ OMX Group, Inc. • 2013

3.2.25.4 Structure

The MO415 TRANSACTION has the following structure:

```
struct hv_order_trans_p  // Named struct no: 34105
```

3.2.25.5 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.

3.2.25.6 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.

3.2.26 MO417 [MO33 With Trader ID TRANSACTION]

3.2.26.1 Fingerprint

TRANSACTION properties	
transaction type	MO417
calling sequence	omniapi_tx_ex
struct name	hv_alter_trans_p
facility	EP0
partitioned	true

3.2.26.2 Related Messages

This is a proxy transaction for MO33.

3.2.26.3 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.

3.2.26.4 Structure

The MO417 TRANSACTION has the following structure:

```
struct hv_alter_trans_p  // Named struct no: 34110
```

© The NASDAQ OMX Group, Inc. • 2013

3.2.26.5 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.

3.2.26.6 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.

3.2.27 MO420 [MO36 With Trader ID TRANSACTION]

3.2.27.1 Fingerprint

TRANSACTION properties	
transaction type	MO420
calling sequence	omniapi_tx_ex
struct name	block_price_trans_p
facility	EP0
partitioned	true

3.2.27.2 Related Messages

This is a proxy transaction for MO36.

3.2.27.3 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.

3.2.27.4 Structure

The MO420 TRANSACTION has the following structure:

```
struct block_price_trans_p  // Named struct no: 34107
```

3.2.27.5 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.

3.2.27.6 Return Codes

The return codes associated with the transaction are the same as for the base transaction, MO36.

© The NASDAQ OMX Group, Inc. • 2013

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

3.2.28 MO421 [MO37 With Trader ID TRANSACTION]

3.2.28.1 Fingerprint

TRANSACTION properties	
transaction type	MO421
calling sequence	omniapi_tx_ex
struct name	hv_price_2_trans_p
facility	EP0
partitioned	true

3.2.28.2 Related Messages

This is a proxy transaction for MO37.

3.2.28.3 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.

3.2.28.4 Structure

The MO421 TRANSACTION has the following structure:

```
struct hv_price_2_trans_p  // Named struct no: 34101
```

3.2.28.5 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.

3.2.28.6 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.

3.2.29 MO424 [Proxy Delete inactive order TRANSACTION]

3.2.29.1 Fingerprint

TRANSACTION properties	
transaction type	MO424

TRANSACTION properties	
calling sequence	omniapi_tx_ex
struct name	delete_trans_p
facility	EP0
partitioned	true

3.2.29.2 Related Messages

This is a proxy transaction for MO40.

3.2.29.3 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.

3.2.29.4 Structure

The MO424 TRANSACTION has the following structure:

```
struct delete_trans_p  // Named struct no: 34111
```

3.2.29.5 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.

3.2.29.6 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.

3.2.30 MO425 [Proxy Stop Order TRANSACTION]

3.2.30.1 Fingerprint

TRANSACTION properties	
transaction type	MO425
calling sequence	omniapi_tx_ex
struct name	stop_order_trans_p
facility	EP0
partitioned	false

© The NASDAQ OMX Group, Inc. • 2013

3.2.30.2 Purpose

This transaction is used for placing stop (stop-loss) orders in the order book on behalf of someone else. This is the proxy version of MO41.

3.2.30.3 Structure

The MO425 TRANSACTION has the following structure:

```
struct stop_order_trans_p  // Named struct no: 34117
```

3.2.30.4 Usage and conditions

Client

Wildcard characters are not allowed in the Client field.

Stop Condition

defines what trigger mechanism to use for the stop order.

Stop Series

is the series that will be used for checking the stop condition.

Premium, Limit

is the stop price that is compared to the price defined by the stop condition.

Trading Code

should be used to specify for whom you are placing the stop order.

3.2.30.5 Return codes

Cstatus	Txstat	Orddt
Successful	1 - No part of the order placed in the order book and no part closed.	Order number
Successful	4 - The whole order placed in the order book.	Order number
Transaction aborted	GEN_CDC_INT_CLOSED - Instrument Type is not open for this Transaction Type.	-
Transaction Aborted	...	

After a successful MO425 transaction, an order number and information regarding the state of the order will be returned to the sender.

An MO425 transaction may also be aborted by the Marketplace, in which case only the reason for the transaction being aborted is returned to the sender.

© The NASDAQ OMX Group, Inc. • 2013

3.2.31 MO427 [Proxy Alter Stop Order TRANSACTION]

3.2.31.1 Fingerprint

TRANSACTION properties	
transaction type	MO427
calling sequence	omniapi_tx_ex
struct name	hv_alter_trans_p
facility	EP0
partitioned	false

3.2.31.2 Purpose

This transaction is used to alter a stop order on behalf of someone else.

3.2.31.3 Structure

The MO427 TRANSACTION has the following structure:

```
struct hv_alter_trans_p  // Named struct no: 34110
```

3.2.31.4 Usage and conditions

Only one existing stop order can be altered at a time. A unique order number refers to the order that should be altered.

The fields can be altered in the same way as in the MO33 transaction with the following exceptions:

Delta Quatity

must be set to 1.

Stop Condition

cannot be altered.

Block Size

can be altered.

Trading Code

Trading Code should be used to specify for whom you are altering the stop order.

The fields that should be altered must be completed. The remaining fields are set to zero.

© The NASDAQ OMX Group, Inc. • 2013

3.2.31.5 Return codes

After a successful MO427 transaction, the number of contracts before the stop order is changed, or zero if no order exists, is returned to the sender. Not finding a stop order to alter is considered to be a successful operation.

3.2.32 MO428 [Proxy Delete Stop Order TRANSACTION]

3.2.32.1 Fingerprint

TRANSACTION properties	
transaction type	MO428
calling sequence	omniapi_tx_ex
struct name	delete_trans_p
facility	EP0
partitioned	true

3.2.32.2 Purpose

This transaction is used to remove one or more stop orders from the order book on behalf of someone else. This is a proxy version of MO44.

3.2.32.3 Structure

The MO428 TRANSACTION has the following structure:

```
struct delete_trans_p  // Named struct no: 34111
```

3.2.32.4 Usage and conditions

This transaction has the same contents as MO44 except for one extra field, Trading Code.

Trading Code

should be used to specify for whom you are removing the stop order.

3.2.32.5 Return codes

Cstatus	Txstat	Ordidt
Successful	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.	-

Cstatus	Txstat	Ordidt
Successful	-	-
Transaction aborted	GEN_CDC_INT_CLOSED - Instrument Type is not open for this Transaction Type.	-
Transaction Aborted	...	-

After a successful MO428 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.

An MO428 transaction may also be aborted by the Marketplace, in which case only the reason for the transaction being aborted is returned to the sender.

3.2.33 MO459 [Trade Report, Proxy TRANSACTION]

3.2.33.1 Fingerprint

TRANSACTION properties	
transaction type	MO459
calling sequence	omniapi_tx_ex
struct name	trade_report_1_trans_p
facility	EP0
partitioned	true

3.2.33.2 Related Messages

- This is a proxy transaction for MO75.

  DQ45

3.2.33.3 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.

3.2.33.4 Structure

The MO459 TRANSACTION has the following structure:

```
struct trade_report_1_trans_p  // Named struct no: 34119
```


© The NASDAQ OMX Group, Inc. • 2013

3.2.34 MO474 [Linked Order Proxy VIT]

3.2.34.1 Fingerprint

VIT properties	
transaction type	MO474
calling sequence	omniapi_tx_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

3.2.34.2 Purpose

This transaction is used to insert new linked orders.

3.2.34.3 Structure

The MO474 VIT has the following structure:

```
struct linked_order_insert {
   struct transaction_type
   struct series  // Named struct no: 50000
   UINT16_T items_n  // Items
   UINT16_T size_n  // Size
}
Sequence {
   struct sub_item_hdr
   Choice {
      struct linked_order_leg  // Named struct no: 34803
      struct exchange_info  // Named struct no: 50004
      struct free_text  // Named struct no: 34801
      struct clearing_info  // Named struct no: 34802
      struct time_in_force  // Named struct no: 34807
      struct order_owner  // Named struct no: 34804
   }
}
```

3.2.35 MO481 [Indicative Quote Proxy VIT]

3.2.35.1 Fingerprint

VIT properties	
transaction type	MO481
calling sequence	omniapi_tx_ex

© The NASDAQ OMX Group, Inc. • 2013

VIT 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.
facility	EP0
partitioned	true

3.2.35.2 Related Messages

This is a proxy transaction for MO97.

3.2.35.3 Purpose

This is a Trader ID transaction, which is used when a trader, user, or application wants to insert indicative quotes on behalf of someone else.

3.2.35.4 Structure

The MO481 VIT has the following structure:

```
struct order_trans_hdr
Sequence {
   struct item_hdr
   Sequence {
      struct sub_item_hdr
      Choice {
         struct indicative_quote  // Named struct no: 34025
         struct order_owner  // Named struct no: 34804
      }
   }
}
```

3.2.35.5 Structure Contents

Some structures in the transaction require additional explanations.

Indicative Quote

Field usage in this structure:

Buy/Ask Price can be given as one of two values: A positive value or zero means the Buy/Ask price. To indicate an undisclosed price, bit 31 should be set (the highest bit, MIN_INT) while all other bits are set to zero.

© The NASDAQ OMX Group, Inc. • 2013

Buy/Ask Quantity can be given as one of two values: A positive number means Quoted quantity. Zero means that the quantity is undisclosed.

3.2.35.6 Usage and conditions

The only thing that differentiates MO481 from MO97 is an extra sub-struct called order_owner , which must be filled with the trading code of the user the order originates from. The sub-struct order_owner can only exist once.

3.2.36 MO484 [Alter Linked Order Proxy VIT]

3.2.36.1 Fingerprint

VIT properties	
transaction type	MO484
calling sequence	omniapi_tx_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

3.2.36.2 Purpose

This transaction is used to update linked orders.

3.2.36.3 Structure

The MO484 VIT has the following structure:

```
struct linked_order_update {
   struct transaction_type
   struct series  // Named struct no: 50000
   QUAD_WORD order_number_u  // Order Number
   UINT16_T items_n  // Items
   UINT16_T size_n  // Size
}
Sequence {
   struct sub_item_hdr
   Choice {
      struct linked_order_leg  // Named struct no: 34803
      struct exchange_info  // Named struct no: 50004
      struct free_text  // Named struct no: 34801
      struct clearing_info  // Named struct no: 34802
      struct time_in_force  // Named struct no: 34807
      struct order_owner  // Named struct no: 34804
   }
```

```
}
```

3.2.37 MQ5 [Proxy Order QUERY]

3.2.37.1 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

3.2.37.2 Purpose

This transaction is used for querying orders entered on behalf of someone else (with MOX+384 transactions).

3.2.37.3 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
}
```

3.2.37.4 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.

© The NASDAQ OMX Group, Inc. • 2013

Note:

All character fields must be space padded up to the total length of the field.

3.2.37.5 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.

3.2.37.6 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
   }
}
```

3.2.37.7 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.

3.2.38 MQ7 [Total Order Book QUERY]

3.2.38.1 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

3.2.38.2 Purpose

This transaction is used for querying all orders in the Order Book.

3.2.38.3 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
}
```


© The NASDAQ OMX Group, Inc. • 2013

3.2.38.4 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.

3.2.38.5 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
    }
}
```

3.2.38.6 Answer, comments

If the trader identity is not public information, party is blanked.

Algorithm to consolidate MQ7 and BO2

Below is a simple client side algorithm to consolidate / align the MQ7 responses with the BO2 stream of broadcasts in order to obtain a consistent view of the market.

The Trading Workstation does the consolidation based on this algorithm, and it can be used by other API clients too when there is a need to do this consolidation (i.e. for instruments where BO2s are disseminated).

The algorithm handles the situations when:

- The two data sources for MQ7 and BO2 have applied a number of order book updates (to a given series) and they are out of sync.
- There is a gap due to the asynchronous nature of the subscription request.

```
• Subscribe to BO2 broadcasts and keep all BO2s in memory
• Submit a period query (MQ7) to retrieve the current order book snapshot
  (e.g. every 5 seconds)
• While the MA42 response is empty, and as long as there is no BO2 received,
   continue the periodic query
• Now:
   • Assume first received BO2 sequence number is x (e.g. 39)
   • Assume the most recently received MQ7 response (MA42) sequence number
     is y (e.g. 50)
   • When y is greater or equal to x, the periodic query is aborted
• It is then safe to replay any BO2 messages numbered y+1 and higher (51 and
   onwards) to the most recent MA42 and thus create a consistent order book
   depth view.
```

3.2.39 MQ8 [Total Order QUERY]

3.2.39.1 Fingerprint

QUERY properties	
transaction type	MQ8
calling sequence	omniapi_query_ex
struct name	query_tot_order
facility	EP0
partitioned	true
answers	MA43

© The NASDAQ OMX Group, Inc. • 2013

ANSWER properties	
transaction type	MA43
struct name	answer_tot_order
segmented	true

3.2.39.2 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.

3.2.39.3 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
}
```

3.2.39.4 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.

© The NASDAQ OMX Group, Inc. • 2013

3.2.39.5 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 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.

3.2.39.6 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
   }
}
```


© The NASDAQ OMX Group, Inc. • 2013

3.2.39.7 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.

3.2.40 MQ9 [Total Inactive Order QUERY]

3.2.40.1 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

© The NASDAQ OMX Group, Inc. • 2013

3.2.40.2 Purpose

This transaction is used for querying own inactive orders in the Order Book.

3.2.40.3 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
}
```

3.2.40.4 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.

3.2.40.5 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	orderidt	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.

3.2.40.6 Answer Structure

The MA44 ANSWER has the following structure:

© The NASDAQ OMX Group, Inc. • 2013

```
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
   }
}
```

3.2.40.7 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.

3.2.41 MQ32 [Total Session State Type Order QUERY]

3.2.41.1 Fingerprint

QUERY properties	
transaction type	MQ32

QUERY properties	
calling sequence	omniapi_query_ex
struct name	query_tot_order
facility	EP0
partitioned	true
answers	MA32

ANSWER properties	
transaction type	MA32
struct name	answer_tot_order
segmented	true

3.2.41.2 Related Messages

BO5

3.2.41.3 Purpose

This query is used to retrieve your own session state orders in the Order Book.

3.2.41.4 Structure

The MQ32 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
}
```

3.2.41.5 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.

© The NASDAQ OMX Group, Inc. • 2013

Note:

All character fields must be space padded up to the total length of the field.

3.2.41.6 Return Codes

An MQ32 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** Illegal transaction at this time	-	-
Transaction aborted	**MP_QUERY_CUST_UND** Underlying or Customer is not fully defined in query.	-	-

See OMnet System Error Messages Reference for details on why transactions are aborted.

3.2.41.7 Answer Structure

The MA32 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
   }
```

```
}
```

3.2.41.8 Answer, comments

After a successful MQ32 transaction, a list of own session state 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.

3.2.42 MQ34 [Proxy Session State Type Order QUERY]

3.2.42.1 Fingerprint

QUERY properties	
transaction type	MQ34
calling sequence	omniapi_query_ex
struct name	query_tot_order
facility	EP0
partitioned	true
answers	MA34

ANSWER properties	
transaction type	MA34
struct name	answer_tot_order
segmented	true

3.2.42.2 Related Messages

BO5

3.2.42.3 Purpose

This query is used to retrieve your own session state proxy orders in the Order Book.

3.2.42.4 Structure

The MQ34 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
}
```


© The NASDAQ OMX Group, Inc. • 2013

3.2.42.5 Return Codes

An MQ34 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** Illegal transaction at this time	-	-
Transaction aborted	**MP_QUERY_CUST_UND** Underlying or Customer is not fully defined in query.	-	-

See OMnet System Error Messages Reference for details on why transactions are aborted.

3.2.42.6 Answer Structure

The MA34 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
   }
}
```

3.2.42.7 Answer, comments

After a successful MQ34 transaction, a list of own session state 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.

3.2.43 MQ47 [Proxy Query Stop Order QUERY]

3.2.43.1 Fingerprint

QUERY properties	
transaction type	MQ47
calling sequence	omniapi_query_ex
struct name	query_tot_order
facility	EP0
partitioned	false
answers	MA47

ANSWER properties	
transaction type	MA47
struct name	answer_stop_order
segmented	false

3.2.43.2 Purpose

This transaction is used for querying own proxy orders in the order book.

3.2.43.3 Structure

The MQ47 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
}
```

3.2.43.4 Usage and Conditions

Whose

© The NASDAQ OMX Group, Inc. • 2013

should be used to specify for whom you are querying the stop orders. It could be either the firm or a user. To specify a firm, the user field should be space padded.

3.2.43.5 Return Codes

cstatus	txstat	ordidt	rvcbuf
Successful	Normal	Transaction identification	List of stop 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	-	-

An MQ47 transaction may be aborted by the Marketplace. If that happens only the reason for the transaction being aborted is returned to the sender.

3.2.43.6 Answer Structure

The MA47 ANSWER has the following structure:

```
struct answer_stop_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
      struct party
      struct order
      struct stop_series
      struct timestamp_created  // Of type: TIME_SPEC
      struct timestamp_in  // Of type: TIME_SPEC
      INT32_T limit_premium_i  // Premium, Limit
      INT64_T total_volume_i  // Total Volume
      INT64_T display_quantity_i  // Quantity, Display
   }
}
```

3.2.43.7 Answer, comments

After a successful MQ47 transaction, a list of own stop 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 next segment 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 has been reached.

© The NASDAQ OMX Group, Inc. • 2013

3.2.44 MQ48 [External Query Own Stop Orders QUERY]

3.2.44.1 Fingerprint

QUERY properties	
transaction type	MQ48
calling sequence	omniapi_query_ex
struct name	query_tot_order
facility	EP0
partitioned	false
answers	MA48

ANSWER properties	
transaction type	MA48
struct name	answer_stop_order
segmented	false

3.2.44.2 Purpose

The Query Total Stop Order Transaction is used for querying own stop orders in the order book.

3.2.44.3 Structure

The MQ48 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
}
```

3.2.44.4 Usage and conditions

Series

must be zero-filled to get the start segment of the partition. To get next segment and partition, the Series in the previous answer should be used.

Order Index

© The NASDAQ OMX Group, Inc. • 2013

must be zero-filled to get the start segment of the partition. To get next segment and partition, the Order Index in the previous answer should be used.

3.2.44.5 Return Codes

An MQ48 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	Transaction identification
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 the query	-	-

Please refer to the **OMnet Error Message Reference** manual for details on why transactions are aborted.

3.2.44.6 Answer Structure

The MA48 ANSWER has the following structure:

```
struct answer_stop_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
      struct party
      struct order
      struct stop_series
      struct timestamp_created  // Of type: TIME_SPEC
      struct timestamp_in  // Of type: TIME_SPEC
      INT32_T limit_premium_i  // Premium, Limit
      INT64_T total_volume_i  // Total Volume
      INT64_T display_quantity_i  // Quantity, Display
   }
}
```

3.2.44.7 Answer, comments

After a successful MQ48 transaction, a list of own stop orders in the order book is returned to the sender.

Series

used to get the next segment.

Order Index

used to get the next segment.

Item

number of stop orders in this answer.

If the series in the answer is zero-filled, the end of the last partition has been reached.

3.2.45 MQ49 [Ext. Query Inactive Stop Orders QUERY]

3.2.45.1 Fingerprint

QUERY properties	
transaction type	MQ49
calling sequence	omniapi_query_ex
struct name	query_tot_order
facility	EP0
partitioned	false
answers	MA49

ANSWER properties	
transaction type	MA49
struct name	answer_stop_order
segmented	false

3.2.45.2 Purpose

The Query Total Inactive Stop Order Transaction is used to for querying own inactive stop orders in the order book.

Note that inactive stop orders cannot be activated using MO99.

To enter the same stop order the application can enter a new stop order with the same details using the regular enter stop order transaction.

3.2.45.3 Structure

The MQ49 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
}
```


© The NASDAQ OMX Group, Inc. • 2013

3.2.45.4 Usage and conditions

Series

must be zero-filled to get the start segment of the partition. To get next segment and partition, the Series in the previous answer should be used.

3.2.45.5 Return Codes

An MQ49 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 stop orders – see below
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 the query	-	-

Please refer to **OMnet Error Message Reference** for details on why transactions are aborted.

3.2.45.6 Answer Structure

The MA49 ANSWER has the following structure:

```
struct answer_stop_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
      struct party
      struct order
      struct stop_series
      struct timestamp_created  // Of type: TIME_SPEC
      struct timestamp_in  // Of type: TIME_SPEC
      INT32_T limit_premium_i  // Premium, Limit
      INT64_T total_volume_i  // Total Volume
      INT64_T display_quantity_i  // Quantity, Display
   }
}
```

3.2.45.7 Answer, comments

After a successful MQ49 transaction, a list of own inactive stop orders in the order book is returned to the sender.

Series

used to get the next segment.

Item

number of stop orders in this answer.

Order Index

used to get the next segment.

If the series in the answer is zero-filled, the end of the last partition has been reached.

3.2.46 MQ67 [Total Order Book Query for Issuer QUERY]

3.2.46.1 Fingerprint

QUERY properties	
transaction type	MQ67
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

3.2.46.2 Purpose

This transaction is used for querying all orders in the Order Book.

Note: This transaction can return different answer structures.

3.2.46.3 Structure

The MQ67 QUERY has the following structure:

© The NASDAQ OMX Group, Inc. • 2013

```
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
}
```

3.2.46.4 Return Codes

An MQ67 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	…	-	-

3.2.46.5 Answer Structure

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
```

```
    }
}
```

3.2.46.6 Answer, comments

If the trader identity is not public information, party is blanked.

3.2.47 MQ78 [Query Trade Reports, Unmatched QUERY]

3.2.47.1 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

3.2.47.2 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).

3.2.47.3 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
}
```

3.2.47.4 Usage and conditions

Series

© The NASDAQ OMX Group, Inc. • 2013

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.

3.2.47.5 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
      struct exchange_info_s  // Internally overlayed structure:
OM_EXCHANGE_INFO
      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
   }
}
```

3.2.47.6 Answer, comments

After a successful MQ78 transaction, a number of answer items are 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.

3.2.48 MQ80 [Query Trade Reports Counterpart, Unmatched QUERY]

3.2.48.1 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

3.2.48.2 Purpose

This query is used to retrieve all unmatched trade reports where the participant has been appointed as a counterparty.

3.2.48.3 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
}
```


© The NASDAQ OMX Group, Inc. • 2013

3.2.48.4 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.

3.2.48.5 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
      struct exchange_info_s  // Internally overlayed structure:
OM_EXCHANGE_INFO
      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
   }
}
```

3.2.48.6 Answer, comments

After a successful MQ80 transaction, a number of answer items are returned to the sender. The Order Number, Series and Bid or Ask must be zero-filled to get the start segment of the partition. To get the next

© The NASDAQ OMX Group, Inc. • 2013

segments and partition, the Order Number, Series and Bid or Ask in the previous answer should be used. If the Series in the answer is zero-filled the end of the last partition is reached.

3.2.49 MQ90 [Own Linked Order QUERY]

3.2.49.1 Fingerprint

QUERY properties	
transaction type	MQ90
calling sequence	omniapi_query_ex
struct name	query_order_private
facility	EP0
partitioned	true
answers	MA100

VIA properties	
transaction type	MA100
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

3.2.49.2 Purpose

The purpose with this query is to retrieve linked orders.

3.2.49.3 Structure

The MQ90 QUERY has the following structure:

```
struct query_order_private {
   struct transaction_type
   struct series  // Named struct no: 50000
   struct search_series  // Of type: SERIES ; Named struct no: 50000
   struct whose
   QUAD_WORD order_number_u  // Order Number
   UINT8_T bid_or_ask_c  // Bid or Ask
   char[3] filler_3_s  // Filler
}
```

3.2.49.4 Usage and conditions

Order Number

© The NASDAQ OMX Group, Inc. • 2013

is the order ID and is optional.

Search Series

Query for a specific instrument series or according to a wildcard filter.

Series

is used for routing.

Whose

specify participant or user as query filter.

3.2.49.5 Answer Structure

The MA100 VIA has the following structure:

```
struct answer_order_hdr {
   struct transaction_type
   struct next_series  // Of type: SERIES ; Named struct no: 50000
   QUAD_WORD next_order_number_u  // Order Number ; Of type: ORDER_NUMBER_U
   UINT8_T bid_or_ask_c  // Bid or Ask
   char[3] filler_3_s  // Filler
   UINT16_T items_n  // Items
   UINT16_T size_n  // Size
}
Sequence {
   struct item_hdr
   Sequence {
      struct sub_item_hdr
      Choice {
         struct order_number  // Named struct no: 34805
         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 order_owner  // Named struct no: 34804
         struct linked_order_base  // Named struct no: 34810
      }
   }
}
```

3.2.50 MQ95 [One Sided Auction QUERY]

3.2.50.1 Fingerprint

QUERY properties	
transaction type	MQ95
calling sequence	omniapi_query_ex
struct name	query_one_sided_auction

QUERY properties	
facility	EP0
partitioned	true
answers	MA95

ANSWER properties	
transaction type	MA95
struct name	answer_one_sided_auction
segmented	true

3.2.50.2 Related Messages

BI95

3.2.50.3 Purpose

The purpose of this query is to retrieve BI95 one-sided auction results.

3.2.50.4 Structure

The MQ95 QUERY has the following structure:

```
struct query_one_sided_auction {
   struct transaction_type
   struct series  // Named struct no: 50000
   UINT16_T segment_number_n  // Segment Number
   UINT8_T only_this_series_c  // Series, Only this
   CHAR filler_1_s  // Filler
}
```

3.2.50.5 Answer Structure

The MA95 ANSWER has the following structure:

```
struct answer_one_sided_auction {
   struct transaction_type
   struct series  // Named struct no: 50000
   UINT16_T segment_number_n  // Segment Number
   UINT16_T items_n  // Items
   Array ITEM [max no: 900] {
      struct series  // Named struct no: 50000
      struct timestamp  // Of type: TIME_SPEC
      INT32_T equilibrium_price  // Premium ; Of type: PREMIUM_I
      INT32_T high_price  // Premium ; Of type: PREMIUM_I
      INT32_T low_price  // Premium ; Of type: PREMIUM_I
      INT32_T vwap_match_price  // Premium ; Of type: PREMIUM_I
      INT64_T respondent_quantity  // Quantity ; Of type: QUANTITY_I
      INT64_T matching_quantity  // Quantity ; Of type: QUANTITY_I
      INT64_T imbalance_quantity  // Quantity ; Of type: QUANTITY_I
```


© The NASDAQ OMX Group, Inc. • 2013

```
      UINT16_T respondent_order_count  // Number of orders ; Of type:
NUMBER_OF_ORDERS_N
      UINT16_T matching_order_count  // Number of orders ; Of type:
NUMBER_OF_ORDERS_N
      UINT8_T is_preliminary_c  // Is Preliminary
      char[3] filler_3_s  // Filler
   }
}
```

3.2.51 MQ98 [Indicative Quotes Public QUERY]

3.2.51.1 Fingerprint

QUERY properties	
transaction type	MQ98
calling sequence	omniapi_query_ex
struct name	query_order_public
facility	EP0
partitioned	true
answers	MA98

VIA properties	
transaction type	MA98
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

3.2.51.2 Purpose

This transaction is used for querying all indicative quotes.

3.2.51.3 Structure

The MQ98 QUERY has the following structure:

```
struct query_order_public {
   struct transaction_type
   struct series  // Named struct no: 50000
   struct search_series  // Of type: 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
}
```

© The NASDAQ OMX Group, Inc. • 2013

3.2.51.4 Return Codes

An MQ98 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	-	-	
Transaction aborted	…	-	-

3.2.51.5 Answer Structure

The MA98 VIA has the following structure:

```
struct answer_order_hdr {
   struct transaction_type
   struct next_series  // Of type: SERIES ; Named struct no: 50000
   QUAD_WORD next_order_number_u  // Order Number ; Of type: ORDER_NUMBER_U
   UINT8_T bid_or_ask_c  // Bid or Ask
   char[3] filler_3_s  // Filler
   UINT16_T items_n  // Items
   UINT16_T size_n  // Size
}
Sequence {
   struct item_hdr
   Sequence {
      struct sub_item_hdr
      Choice {
         struct indicative_quote_base  // Named struct no: 34026
         struct indicative_quote_fixed_income  // Named struct no: 34027
      }
   }
}
```


© The NASDAQ OMX Group, Inc. • 2013

3.2.51.6 Answer, comments

Sequence Number

is a non-consecutive increasing number per series. It can be used to synchronize the answer to the MQ98 query with the BO98 broadcast flow.

After a successful MQ98 transaction, a list of indicative quotes 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.

3.2.52 MQ99 [Maximum Block Order Sizes QUERY]

3.2.52.1 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

3.2.52.2 Purpose

MQ99 provides the max exchange allowed limit for MO96 and MO36/MO420.

3.2.52.3 Structure

The MQ99 QUERY has the following structure:

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

3.2.52.4 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.

3.2.52.5 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
}
```

3.2.52.6 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.

3.2.53 MQ100 [Own Inactive Linked Order QUERY]

3.2.53.1 Fingerprint

QUERY properties	
transaction type	MQ100
calling sequence	omniapi_query_ex
struct name	query_order_private

© The NASDAQ OMX Group, Inc. • 2013

QUERY properties	
facility	EP0
partitioned	true
answers	MA100

VIA properties	
transaction type	MA100
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

3.2.53.2 Purpose

The purpose with this query is to retrieve inactive linked orders.

3.2.53.3 Structure

The MQ100 QUERY has the following structure:

```
struct query_order_private {
   struct transaction_type
   struct series  // Named struct no: 50000
   struct search_series  // Of type: SERIES ; Named struct no: 50000
   struct whose
   QUAD_WORD order_number_u  // Order Number
   UINT8_T bid_or_ask_c  // Bid or Ask
   char[3] filler_3_s  // Filler
}
```

3.2.53.4 Usage and conditions

Order Number

is the order ID and is optional.

Search Series

Query for a specific instrument series or according to a wildcard filter.

Series

is used for routing.

3.2.53.5 Answer Structure

The MA100 VIA has the following structure:

© The NASDAQ OMX Group, Inc. • 2013

```
struct answer_order_hdr {
   struct transaction_type
   struct next_series  // Of type: SERIES ; Named struct no: 50000
   QUAD_WORD next_order_number_u  // Order Number ; Of type: ORDER_NUMBER_U
   UINT8_T bid_or_ask_c  // Bid or Ask
   char[3] filler_3_s  // Filler
   UINT16_T items_n  // Items
   UINT16_T size_n  // Size
}
Sequence {
   struct item_hdr
   Sequence {
      struct sub_item_hdr
      Choice {
         struct order_number  // Named struct no: 34805
         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 order_owner  // Named struct no: 34804
         struct linked_order_base  // Named struct no: 34810
      }
   }
}
```

3.2.54 MQ151 [Order Broadcast QUERY]

3.2.54.1 Fingerprint

QUERY properties	
transaction type	MQ151
calling sequence	omniapi_query_ex
struct name	query_order_broadcast
facility	EPA
partitioned	true
answers	MA151

VIA properties	
transaction type	MA151
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

© The NASDAQ OMX Group, Inc. • 2013

3.2.54.2 Purpose

This query is used to retrieve missing order-book broadcasts, BO5, for the own participant.

3.2.54.3 Structure

The MQ151 QUERY has the following structure:

```
struct query_order_broadcast {
   struct transaction_type
   struct series  // Named struct no: 50000
   UINT8_T instance_c  // Instance, Number
   char[3] filler_3_s  // Filler
   char[8] yyyymmdd_s  // Date
   struct broadcast_type
   UINT32_T sequence_first_u  // Sequence First
   UINT32_T sequence_last_u  // Sequence Last
}
```

3.2.54.4 Usage and Conditions

Series

Series can be filled up to instrument type.

Instance, Number

The matching engine instance that issued the broadcast (required). This is normally equal to 1 in the first query. The value to use in a consecutive query is returned in the answer.

Date

The business date to gather information for (required). The period available is exchange specific.

Broadcast Type

The type of the broadcast to query history for. Mandatory.

Sequence First
Sequence Last

Sequence First and Sequence Last can optionally be set to zero respectively to retrieve all missing broadcasts.

Example

We want to query for all missing BO5 broadcasts for the current business date, which we assume to be 2005 01 27.

1st query

The query fields are populated as follows:

- Series = zero filled (wildcard)
- Instance Number = 1 (normally 1 in the first query).

© The NASDAQ OMX Group, Inc. • 2013

- Date = 20050127 (YYYYMMDD)
- Broadcast Type = BO5
- Sequence First = 0 (wildcard)
- Sequence Last = 0 (wildcard)

The first item in the answer will always be the structure **Order History Server, Next Query (query_order_broadcast_next, 34911)**. The next query will use this information:

- Series = zero filled (wildcard)
- Instance Number = Next Instance Number (received in answer)
- Date = 20050127 (YYYYMMDD)
- Broadcast Type = BO5
- Sequence First = Sequence First Next (received in answer. This is however blank in the case of a shift of instance.)
- Sequence Last = 0

The query is sent repeatedly until a Next Instance Number equal to zero is returned. This indicates that all BO5:s in the interval have been returned.

3.2.54.5 Return Codes

cstatus	txstat	ordidt	rcvbuf
Successful	MP_SUCCESS	-	-
Successful	MP_OHS_DATAPURGED Data has been purged	-	-
Successful	MP_OHS_DATAINCOMPLETE The return set of data is incomplete. Recovery in progress.	-	-
Successful	MP_OHS_DATAINCOMPLETE_NORECOV The return set of data is incomplete. Recovery is turned off.	-	-
Transaction aborted	MP_OHS_INVTIME An invalid time or date in a request or query.	-	-
Transaction aborted	MP_OHS_INVBDXTYPE Invalid broadcast type in query.	-	-
Transaction aborted	MP_OHS_INVINSTANCE Invalid instance in query.	-	-

3.2.54.6 Answer Structure

The MA151 VIA has the following structure:

© The NASDAQ OMX Group, Inc. • 2013

```
struct answer_hdr
Sequence {
   struct item_hdr
   Sequence {
      struct sub_item_hdr
      Choice {
         struct block_price_trans  // Named struct no: 34007
         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 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 order_price_change  // Named struct no: 34905
         struct order_return_info  // Named struct no: 34904
         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 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 query_order_broadcast_next  // Named struct no: 34911
         struct order_info  // Named struct no: 34917
         struct order_trade_info  // Named struct no: 34920
         struct order_leg_trade_info  // Named struct no: 34921
      }
   }
}
```

3.2.55 MQ154 [Order Broadcast Proxy QUERY]

3.2.55.1 Fingerprint

QUERY properties	
transaction type	MQ154
calling sequence	omniapi_query_ex
struct name	query_order_broadcast_p
facility	EPB
partitioned	true
answers	MA154

VIA properties	
transaction type	MA154

VIA 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.
segmented	true

3.2.55.2 Purpose

This query is used to retrieve missing order-book broadcasts, BO5 for a specified participant.

This is the query used by broker service providers to be able to query for order history for other customers.

3.2.55.3 Structure

The MQ154 QUERY has the following structure:

```
struct query_order_broadcast_p {
   struct transaction type
   struct party
   struct series  // Named struct no: 50000
   UINT8_T instance_c  // Instance, Number
   char[3] filler_3_s  // Filler
   char[8] yyyymmdd_s  // Date
   struct broadcast type
   UINT32_T sequence first u  // Sequence First
   UINT32_T sequence last u  // Sequence Last
}
```

3.2.55.4 Usage and Conditions

Series

Series can be filled up to instrument type.

Instance, Number

Denotes the matching engine partition that the broadcast originates from. Mandatory.

Date

Business date is mandatory.

Broadcast Type

The type of the broadcast to query history for. Mandatory.

Sequence First
Sequence Last

Sequence First and Sequence Last can optionally be set to zero to retrieve all missing broadcasts.

Example

© The NASDAQ OMX Group, Inc. • 2013

Query for all missing BO5 broadcast for participant PART1 for the current business date, which we assume to be 20050127.

The query is populated as follows:

- Party = PART1
- Series, zero filled (wildcard)
- Instance Number = 1, always 1 in the first query
- Date, YYYYMMDD = 20050127
- Broadcast Type = BO5
- Sequence First = 0
- Sequence Last = 0

The first item in the answer will always be the structure Order History Server, Next Query (query_order_broadcast_next, 34911)

The next query will use this information:

- Party = PART1
- Series, zero filled (wildcard)
- Instance Number = 1, Next Instance Number from query_order_broadcast_next
- Date, YYYYMMDD = 20050127
- Broadcast Type = BO5
- Sequence First = Sequence First Next from query_order_broadcast_next
- Sequence Last = 0

All BO5 broadcast have been received when the Next Instance Number is zero.

This query requires that the querying participant is authorized to see the BO5 stream of the participant PART1. This configuration is done by the exchange.

Party

Note:

All character fields must be space padded up to the total length of the field.

3.2.55.5 Answer Structure

The MA154 VIA has the following structure:

```
struct answer_hdr
Sequence {
   struct item_hdr
   Sequence {
      struct sub_item_hdr
      Choice {
         struct block_price_trans  // Named struct no: 34007
         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
```

© The NASDAQ OMX Group, Inc. • 2013

```
            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 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 order_price_change  // Named struct no: 34905
            struct order_return_info  // Named struct no: 34904
            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 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 query_order_broadcast_next  // Named struct no: 34911
            struct order_info  // Named struct no: 34917
            struct order_trade_info  // Named struct no: 34920
            struct order_leg_trade_info  // Named struct no: 34921
        }
    }
}
```

3.2.56 MQ392 [MQ8 With Trader ID QUERY]

3.2.56.1 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

3.2.56.2 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.

© The NASDAQ OMX Group, Inc. • 2013

3.2.56.3 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
}
```

3.2.56.4 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.

3.2.56.5 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
   }
}
```

© The NASDAQ OMX Group, Inc. • 2013

3.2.56.6 Answer, comments

The answer from the query is the same as for the base transaction, MQ8.

3.2.57 MQ393 [MQ9 With Trader ID QUERY]

3.2.57.1 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

3.2.57.2 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.

3.2.57.3 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
}
```

3.2.57.4 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 NASDAQ OMX Group, Inc. • 2013

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.

3.2.57.5 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
   }
}
```

3.2.57.6 Answer, comments

The answer from the query is the same as for the base transaction, MQ9.

3.2.58 MQ396 [Internal Own Linked Order Proxy QUERY]

3.2.58.1 Fingerprint

QUERY properties	
transaction type	MQ396
calling sequence	omniapi_query_ex
struct name	query_order_private
facility	EP0
partitioned	true

QUERY properties	
answers	MA100

VIA properties	
transaction type	MA100
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

3.2.58.2 Purpose

The purpose with this query is to retrieve linked orders.

3.2.58.3 Structure

The MQ396 QUERY has the following structure:

```
struct query_order_private {
   struct transaction type
   struct series  // Named struct no: 50000
   struct search series  // Of type: SERIES ; Named struct no: 50000
   struct whose
   QUAD_WORD order_number_u  // Order Number
   UINT8_T bid_or_ask_c  // Bid or Ask
   char[3] filler_3_s  // Filler
}
```

3.2.58.4 Usage and conditions

Order Number

is the order ID and is optional.

Search Series

Query for a specific instrument series or according to a wildcard filter.

Series

is used for routing.

Whose

specify participant or user as query filter.

3.2.58.5 Answer Structure

The MA100 VIA has the following structure:

© The NASDAQ OMX Group, Inc. • 2013

```
struct answer_order_hdr {
   struct transaction_type
   struct next_series  // Of type: SERIES ; Named struct no: 50000
   QUAD_WORD next_order_number_u  // Order Number ; Of type: ORDER_NUMBER_U
   UINT8_T bid_or_ask_c  // Bid or Ask
   char[3] filler_3_s  // Filler
   UINT16_T items_n  // Items
   UINT16_T size_n  // Size
}
Sequence {
   struct item_hdr
   Sequence {
      struct sub_item_hdr
      Choice {
         struct order_number  // Named struct no: 34805
         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 order_owner  // Named struct no: 34804
         struct linked_order_base  // Named struct no: 34810
      }
   }
}
```

3.2.59 MQ397 [Internal Own Inactive Linked Order Proxy QUERY]

3.2.59.1 Fingerprint

QUERY properties	
transaction type	MQ397
calling sequence	omniapi_query_ex
struct name	query_order_private
facility	EP0
partitioned	true
answers	MA100

VIA properties	
transaction type	MA100
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

© The NASDAQ OMX Group, Inc. • 2013

3.2.59.2 Purpose

The purpose with this query is to retrieve inactive linked orders.

3.2.59.3 Structure

The MQ397 QUERY has the following structure:

```
struct query_order_private {
   struct transaction_type
   struct series  // Named struct no: 50000
   struct search_series  // Of type: SERIES ; Named struct no: 50000
   struct whose
   QUAD_WORD order_number_u  // Order Number
   UINT8_T bid_or_ask_c  // Bid or Ask
   char[3] filler_3_s  // Filler
}
```

3.2.59.4 Usage and conditions

Order Number

is the order ID and is optional.

Search Series

Query for a specific instrument series or according to a wildcard filter.

Series

is used for routing.

3.2.59.5 Answer Structure

The MA100 VIA has the following structure:

```
struct answer_order_hdr {
   struct transaction_type
   struct next_series  // Of type: SERIES ; Named struct no: 50000
   QUAD_WORD next_order_number_u  // Order Number ; Of type: ORDER_NUMBER_U
   UINT8_T bid_or_ask_c  // Bid or Ask
   char[3] filler_3_s  // Filler
   UINT16_T items_n  // Items
   UINT16_T size_n  // Size
}
Sequence {
   struct item_hdr
   Sequence {
      struct sub_item_hdr
      Choice {
         struct order_number  // Named struct no: 34805
         struct time_in_force  // Named struct no: 34807
         struct exchange_info  // Named struct no: 50004
```


© The NASDAQ OMX Group, Inc. • 2013

```
            struct free_text  // Named struct no: 34801
            struct clearing_info  // Named struct no: 34802
            struct linked_order_leg  // Named struct no: 34803
            struct order_owner  // Named struct no: 34804
            struct linked_order_base  // Named struct no: 34810
         }
      }
   }
```

3.2.60 MQ398 [Internal Query Proxy Trade Reports, Unmatched QUERY]

3.2.60.1 Fingerprint

QUERY properties	
transaction type	MQ398
calling sequence	omniapi_query_ex
struct name	query_tot_order
facility	EP0
partitioned	true
answers	MA334

ANSWER properties	
transaction type	MA334
struct name	answer_trd_report
segmented	true

3.2.60.2 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).

3.2.60.3 Structure

The MQ398 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
}
```

3.2.60.4 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.

3.2.60.5 Answer Structure

The MA334 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
      struct exchange_info_s  // Internally overlayed structure:
OM_EXCHANGE_INFO
      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
   }
```


© The NASDAQ OMX Group, Inc. • 2013

```
}
```

3.2.60.6 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.

3.2.61 MQ416 [Proxy Total Session State Type Order QUERY]

3.2.61.1 Fingerprint

QUERY properties	
transaction type	MQ416
calling sequence	omniapi_query_ex
struct name	query_tot_order
facility	EP0
partitioned	true
answers	MA32

ANSWER properties	
transaction type	MA32
struct name	answer_tot_order
segmented	true

3.2.61.2 Related Messages

MQ32

BO5

3.2.61.3 Purpose

This query is a proxy version of MQ32 and is used to retrieve session state orders.

3.2.61.4 Structure

The MQ416 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
```

```
}
```

3.2.61.5 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 MQ416 differs from MQ32 is how the whose field is filled out.

> **Note:**
> All character fields must be space padded up to the total length of the field.

3.2.61.6 Return Codes

An MQ416 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** Illegal transaction at this time	-	-
Transaction aborted	**MP_QUERY_CUST_UND** Underlying or Customer is not fully defined in query.	-	-

See OMnet System Error Messages Reference for details on why transactions are aborted.

3.2.61.7 Answer Structure

The MA32 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
```


© The NASDAQ OMX Group, Inc. • 2013

```
        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
    }
}
```

3.2.61.8 Answer, comments

After a successful MQ416 transaction, a list of session state 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.

3.2.62 MQ432 [Proxy Query Own Stop Orders QUERY]

3.2.62.1 Fingerprint

QUERY properties	
transaction type	MQ432
calling sequence	omniapi_query_ex
struct name	query_tot_order_p
facility	EP0
partitioned	false
answers	MA48

ANSWER properties	
transaction type	MA48
struct name	answer_stop_order
segmented	false

3.2.62.2 Related Messages

This transaction has the same contents as MQ48 and MA48.

3.2.62.3 Purpose

This transaction is used for querying stop orders in the order book on behalf of someone else.

© The NASDAQ OMX Group, Inc. • 2013

3.2.62.4 Structure

The MQ432 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
}
```

3.2.62.5 Usage and Conditions

Whose

should be used to specify for whom you are querying the stop orders. It could be either the firm or a user. To specify a firm, the user field should be space padded.

3.2.62.6 Return Codes

cstatus	txstat	ordidt	rvcbuf
Successful	Normal	Transaction identification	List of stop 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	-	-

An MQ432 transaction may be aborted by the Marketplace. If that happens only the reason for the transaction being aborted is returned to the sender.

3.2.62.7 Answer Structure

The MA48 ANSWER has the following structure:

```
struct answer_stop_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
      struct party
      struct order
      struct stop_series
      struct timestamp_created  // Of type: TIME_SPEC
```


© The NASDAQ OMX Group, Inc. • 2013

```
        struct timestamp_in  // Of type: TIME_SPEC
        INT32_T limit_premium_i  // Premium, Limit
        INT64_T total_volume_i  // Total Volume
        INT64_T display_quantity_i  // Quantity, Display
    }
}
```

3.2.62.8 Answer, comments

After a successful MQ432 transaction, a list of own stop 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 next segment 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 has been reached.

3.2.63 MQ433 [Proxy Query Inact. Stop Orders QUERY]

3.2.63.1 Fingerprint

QUERY properties	
transaction type	MQ433
calling sequence	omniapi_query_ex
struct name	query_tot_order_p
facility	EP0
partitioned	false
answers	MA49

ANSWER properties	
transaction type	MA49
struct name	answer_stop_order
segmented	false

3.2.63.2 Related Messages

This transaction has the same contents as MQ49 and MA49.

3.2.63.3 Purpose

This transaction is used for querying inactive stop orders in the order book on behalf of s someone else.

3.2.63.4 Structure

The MQ433 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
}
```

3.2.63.5 Usage and Conditions

Whose

should be used to specify for whom you are querying the stop orders. It could be either the firm or a user. To specify a firm, the user field should be space padded.

3.2.63.6 Return Codes

cstatus	txstat	ordidt	rvcbuf
Successful	Normal	Transaction identification	List of stop 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	-	-

An MQ433 transaction may be aborted by the Marketplace. If that happens only the reason for the transaction being aborted is returned to the sender.

3.2.63.7 Answer Structure

The MA49 ANSWER has the following structure:

```
struct answer_stop_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
      struct party
      struct order
      struct stop_series
      struct timestamp_created  // Of type: TIME_SPEC
      struct timestamp_in  // Of type: TIME_SPEC
      INT32_T limit_premium_i  // Premium, Limit
      INT64_T total_volume_i  // Total Volume
      INT64_T display_quantity_i  // Quantity, Display
   }
}
```


© The NASDAQ OMX Group, Inc. • 2013

3.2.63.8 Answer, comments

After a successful MQ433 transaction, a list of own inactive stop 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 next segment 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 has been reached.

3.2.64 MQ434 [MQ50 With Trader ID QUERY]

3.2.64.1 Fingerprint

QUERY properties	
transaction type	MQ434
calling sequence	omniapi_query_ex
struct name	query_tot_order_p
facility	EP0
partitioned	true
answers	MA50

ANSWER properties	
transaction type	MA50
struct name	answer_tot_order
segmented	true

3.2.64.2 Purpose

This transaction is used to get all inactive orders placed on-behalf of all members.

3.2.64.3 Structure

The MQ434 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
}
```

3.2.64.4 Related Messages

- MQ50 is the external variant.
- MQ306 is the internal variant.

© The NASDAQ OMX Group, Inc. • 2013

3.2.64.5 Usage and conditions

Trading Code

must be filled with the trading code of all inactive orders which was placed on behalf.

3.2.64.6 Return Codes

Same as MQ8 (structure=answer_tot_order)

3.2.64.7 Answer Structure

The MA50 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
   }
}
```

3.2.65 MQ67 [Total Order Book Query for Issuer QUERY]

3.2.65.1 Fingerprint

QUERY properties	
transaction type	MQ67
calling sequence	omniapi_query_ex
struct name	query_tot_ob
facility	EP0
partitioned	true

© The NASDAQ OMX Group, Inc. • 2013

QUERY properties	
answers	MA42

ANSWER properties	
transaction type	MA42
struct name	answer_tot_ob
segmented	true

3.2.65.2 Purpose

This transaction is used for querying all orders in the Order Book. It is a proxy version of MQ67.

Note: This transaction can return different answer structures.

3.2.65.3 Structure

The MQ67 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
}
```

3.2.65.4 Return Codes

An MQ67 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)

cstatus	txstat	ordidt	rcvbuf
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	…	-	-

3.2.65.5 Answer Structure

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
   }
}
```

3.2.65.6 Answer, comments

If the trader identity is not public information, party is blanked.

3.2.66 MQ474 [Own Linked Order Proxy QUERY]

3.2.66.1 Fingerprint

QUERY properties	
transaction type	MQ474
calling sequence	omniapi_query_ex

© The NASDAQ OMX Group, Inc. • 2013

QUERY properties	
struct name	query_order_private
facility	EP0
partitioned	true
answers	MA100

VIA properties	
transaction type	MA100
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

3.2.66.2 Purpose

The purpose with this query is to retrieve linked orders.

3.2.66.3 Structure

The MQ474 QUERY has the following structure:

```
struct query_order_private {
   struct transaction_type
   struct series  // Named struct no: 50000
   struct search_series  // Of type: SERIES ; Named struct no: 50000
   struct whose
   QUAD_WORD order_number_u  // Order Number
   UINT8_T bid_or_ask_c  // Bid or Ask
   char[3] filler_3_s  // Filler
}
```

3.2.66.4 Answer Structure

The MA100 VIA has the following structure:

```
   struct answer_order_hdr {
      struct transaction_type
      struct next_series  // Of type: SERIES ; Named struct no: 50000
      QUAD_WORD next_order_number_u  // Order Number ; Of type: ORDER_NUMBER_U
      UINT8_T bid_or_ask_c  // Bid or Ask
      char[3] filler_3_s  // Filler
      UINT16_T items_n  // Items
      UINT16_T size_n  // Size
   }
   Sequence {
      struct item_hdr
      Sequence {
         struct sub_item_hdr
         Choice {
```

© The NASDAQ OMX Group, Inc. • 2013

```
            struct order_number  // Named struct no: 34805
            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 order_owner  // Named struct no: 34804
            struct linked_order_base  // Named struct no: 34810
        }
    }
}
```

3.2.67 MQ484 [Own Inactive Linked Order Proxy QUERY]

3.2.67.1 Fingerprint

QUERY properties	
transaction type	MQ484
calling sequence	omniapi_query_ex
struct name	query_order_private
facility	EP0
partitioned	true
answers	MA100

VIA properties	
transaction type	MA100
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

3.2.67.2 Purpose

The purpose with this query is to retrieve inactive linked orders.

3.2.67.3 Structure

The MQ484 QUERY has the following structure:

```
struct query_order_private {
   struct transaction_type
   struct series  // Named struct no: 50000
   struct search_series  // Of type: SERIES ; Named struct no: 50000
   struct whose
   QUAD_WORD order_number_u  // Order Number
   UINT8_T bid_or_ask_c  // Bid or Ask
```


© The NASDAQ OMX Group, Inc. • 2013

```
    char[3] filler_3_s  // Filler
}
```

3.2.67.4 Answer Structure

The MA100 VIA has the following structure:

```
struct answer_order_hdr {
   struct transaction_type
   struct next_series  // Of type: SERIES ; Named struct no: 50000
   QUAD_WORD next_order_number_u  // Order Number ; Of type: ORDER_NUMBER_U
   UINT8_T bid_or_ask_c  // Bid or Ask
   char[3] filler_3_s  // Filler
   UINT16_T items_n  // Items
   UINT16_T size_n  // Size
}
Sequence {
   struct item_hdr
   Sequence {
      struct sub_item_hdr
      Choice {
         struct order_number  // Named struct no: 34805
         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 order_owner  // Named struct no: 34804
         struct linked_order_base  // Named struct no: 34810
      }
   }
}
```

3.3 Trading and Market Information

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.

Since BD1 broadcasts are not sent out for deals with deal source Internal, Interbank, Correction, MPS, Reverse, or Basis, BD17 must be used to retrieve these deals.

Ticker applicability

Event	Action
Reception of BD1	Use it in ticker as usual.
Reception of BD17	Use it in ticker. Determine by means of the flag hidden_price_c whether to hide the price or not. 1 – Yes 2 – No

3.3.2 BD2 [Edited Price Information VIB]

3.3.2.1 Fingerprint

VIB properties	
transaction type	BD2
calling sequence	omniapi_read_event_ext_ex or omniapi_read_event_block

© The NASDAQ OMX Group, Inc. • 2013

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

3.3.2.2 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.

3.3.2.3 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_trd  // Named struct no: 33036
         struct market_info_omfi  // Named struct no: 33047
         struct ob_levels_closing  // Named struct no: 33031
      }
   }
}
```

3.3.2.4 Usage and Conditions

In order to maintain a real time database of the BD2 information the client application must use the IQ18 query to download a baseline of the information. Please refer to IQ18 for information on the sequence for this.

3.3.2.5 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, Trade Report

This structure is provided in the broadcast 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.

3.3.3 BD3 [Underlying Information BROADCAST]

3.3.3.1 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

© The NASDAQ OMX Group, Inc. • 2013

3.3.3.2 Purpose

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

3.3.3.3 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
}
```

3.3.3.4 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.

3.3.4 BD70 [Trade Ticker VIB]

3.3.4.1 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

© The NASDAQ OMX Group, Inc. • 2013

VIB properties	
segmented	true

3.3.4.2 Related Messages

TR70, BD71, TR71

3.3.4.3 Purpose

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

3.3.4.4 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_fixed_income_trade_ticker  // Named struct no: 34404
      struct_half_trade_ticker  // Named struct no: 34405
   }
}
```

3.3.4.5 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.

3.3.5 BD71 [Amended Trades VIB]

3.3.5.1 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

© The NASDAQ OMX Group, Inc. • 2013

3.3.5.2 Related Messages

TR70, BD70, TR71

3.3.5.3 Purpose

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

3.3.5.4 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
   }
}
```

3.3.5.5 Usage and conditions

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

Block Size

is not available in BD71 and will always be 0.

3.3.6 BI5 [Indices Information BROADCAST]

3.3.6.1 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

3.3.6.2 Purpose

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

© The NASDAQ OMX Group, Inc. • 2013

3.3.6.3 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
}
```

3.3.6.4 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.

3.3.7 BI9 [Price Information Heartbeat BROADCAST]

3.3.7.1 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

3.3.7.2 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.

© The NASDAQ OMX Group, Inc. • 2013

3.3.7.3 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
}
```

3.3.7.4 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.

3.3.8 BI63 [Preliminary Settlement Prices BROADCAST]

3.3.8.1 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

© The NASDAQ OMX Group, Inc. • 2013

3.3.8.2 Purpose

This subscription returns intra day calculated settlement prices.

3.3.8.3 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
   }
}
```

3.3.8.4 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.

© The NASDAQ OMX Group, Inc. • 2013

3.3.9 BO1 [Order Book Changes, with Identity BROADCAST]

3.3.9.1 Fingerprint

BROADCAST properties	
transaction type	BO1
calling sequence	omniapi_read_event_ext_ex or omniapi_read_event_block
struct name	ob_changes_id
info type	instrument class

3.3.9.2 Purpose

This broadcast will return all changes in the Order Book with reference to the specified Commodity Code regardless of Instrument Group.

3.3.9.3 Structure

The BO1 BROADCAST has the following structure:

```
struct ob_changes_id {
   struct broadcast_type
   struct changes
   QUAD_WORD order_number_u  // Order Number
   struct order_no_id
   struct party
}
```

3.3.9.4 Usage and Conditions

Additional information will be provided for markets that permit identities to be known.

The information describes the alteration made and refers the changed data.

It is recommended to ask for the event for the information needed and thereafter to send an order query transaction.

To obtain an Order Book mirror copy, all broadcasts should be stored until the query is completed. When the sequence number is higher than the sequence number for this series in the answer, the broadcast must be taken care of.

An Order Book change is either ADD, DELETE or ALTER. This is specified in the Order Book Command.

Information for an Order Book command equal to ADD should be interpreted as follows:

- Sequence Number is a consecutive number per series.
- Quantity difference is equal to the Quantity field for an ADD operation.

> *Example 5:*
>
> If you do an ADD operation the remaining orders are each moved to a higher number, i.e. lower position.

© The NASDAQ OMX Group, Inc. • 2013

ADD on order 3 will reposition the order from 3 to 4, order 4 to 5 etc.

Information for an Order Book command equal to DELETE is to be interpreted as follows:

- The deleted order is identified by the position (position in the Order Book) held in the Order Book and by the order number. Remaining fields contain redundant information.

Example 6:

If you do a DELETE operation the remaining orders are each moved to a lower number, i.e.. higher position.

DELETE on order 3 will reposition the order from 3 to 2 and order 2 to 1.

Information for an Order Book command equal to ALTER is to be interpreted as follows:

- The order that has changed (that is, the content has changed but the position in the order book remains) is defined by both order position and order number.
- Quantity difference is the difference between old and new quantity, if the quantity field is changed. (Quantity difference = new quantity - old quantity.)
- The fields that follow contain the values of the order after the alteration has taken place regardless of which field has been changed.

Note: Combination Order Mark has a non-zero value when the order is derived from a combination order (so-called bait order).

Order Type, Exchange

Order Type, Exchange is used to indicate exchange specific order information like ‘Market Bid’ order.

3.3.10 BO2 [Order Book Changes, without Identity BROADCAST]

3.3.10.1 Fingerprint

BROADCAST properties	
transaction type	BO2
calling sequence	omniapi_read_event_ext_ex or omniapi_read_event_block
struct name	ob_changes_no_id
info type	instrument class

3.3.10.2 Related Messages

BO1

© The NASDAQ OMX Group, Inc. • 2013

3.3.10.3 Purpose

This broadcast will return all changes in the Order Book with reference to the specified Commodity Code regardless of Instrument Group.

3.3.10.4 Structure

The BO2 BROADCAST has the following structure:

```
struct ob_changes_no_id {
   struct broadcast_type
   struct changes
   QUAD_WORD order_number_u  // Order Number
   struct order_no_id
}
```

3.3.10.5 Usage and conditions

Additional information will be provided for markets that permit identities to be known. The information describes the alteration made and refers the changed data.

It is recommended to ask for the event for the information needed and thereafter to send an order query transaction.

To obtain an Order Book mirror copy, all broadcasts should be stored until the query is completed. When the sequence number is higher than the sequence number for this series in the answer, the broadcast must be taken care of.

An Order Book change is either ADD, DELETE or ALTER. This is specified in the Order Book Command.

Information for an Order Book command equal to ADD should be interpreted as follows:

- Sequence Number is a consecutive number per series.
- Quantity difference is equal to the Quantity field for an ADD operation.

> *Example 7:*
>
> If you do an ADD operation the remaining orders are each moved to a higher number, i.e. lower position.
>
> ADD on order 3 will reposition the order from 3 to 4, order 4 to 5 etc.

Information for an Order Book command equal to DELETE is to be interpreted as follows:

- The deleted order is identified by the position (position in the Order Book) held in the Order Book and by the order number. Remaining fields contain redundant information.

> *Example 8:*
>
> If you do a DELETE operation the remaining orders are each moved to a lower number, i.e. higher position.
>
> DELETE on order 3 will reposition the order from 3 to 2 and order 2 to 1.

Information for an Order Book command equal to ALTER is to be interpreted as follows:

- The order that has changed (that is, the content has changed but the position in the order book remains) is defined by both order position and order number.

© The NASDAQ OMX Group, Inc. • 2013

- Quantity difference is the difference between old and new quantity, if the quantity field is changed. (Quantity difference = new quantity - old quantity.)
- The fields that follow contain the values of the order after the alteration has taken place regardless of which field has been changed.

Note: Combination Order Mark has a non-zero value when the order is derived from a combination order (so-called bait order).

Order Type, Exchange

Order Type, Exchange is used to indicate exchange specific order information like ‘Market Bid’ order.

3.3.11 BO10 [Equilibrium Price Update BROADCAST]

3.3.11.1 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
virtual underlying	true

3.3.11.2 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.

3.3.11.3 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
```


© The NASDAQ OMX Group, Inc. • 2013

```
    }
}
```

3.3.11.4 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.

3.3.12 BO14 [Order Book Levels VIB]

3.3.12.1 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

3.3.12.2 Related Messages

IQ10/IQ18

3.3.12.3 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.

© The NASDAQ OMX Group, Inc. • 2013

3.3.12.4 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
   }
}
```

3.3.12.5 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.

> **Note:** BO15 and IQ19 are not used by NASDAQ OMX Nordic.

3.3.12.6 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

© The NASDAQ OMX Group, Inc. • 2013

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)

© The NASDAQ OMX Group, Inc. • 2013

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)

© The NASDAQ OMX Group, Inc. • 2013

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, 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:

© The NASDAQ OMX Group, Inc. • 2013

- 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

3.3.13 BO15 [Order Book Levels VIB]

3.3.13.1 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

3.3.13.2 Related Messages

IQ19

3.3.13.3 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.

3.3.13.4 Structure

The BO15 VIB has the following structure:

© The NASDAQ OMX Group, Inc. • 2013

```
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
   }
}
```

3.3.13.5 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.

3.3.13.6 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

© The NASDAQ OMX Group, Inc. • 2013

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)

© The NASDAQ OMX Group, Inc. • 2013

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)

© The NASDAQ OMX Group, Inc. • 2013

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, 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:

© The NASDAQ OMX Group, Inc. • 2013

- 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

3.3.14 BO49 [Price Median VIB]

3.3.14.1 Fingerprint

VIB properties	
transaction type	BO49
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

3.3.14.2 Related Messages

IQ49

3.3.14.3 Purpose

This broadcast is used to distribute the Market by Median Bid and Ask price.

3.3.14.4 Structure

The BO49 VIB has the following structure:

```
struct broadcast_hdr
```

```
Sequence {
   struct_sub_item_hdr
   Choice {
      struct_price_median_id  // Named struct no: 33070
      struct_price_median  // Named struct no: 33071
   }
}
```

3.3.15 II12 [Underlying and indices QUERY]

3.3.15.1 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

3.3.15.2 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

3.3.15.3 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
}
```


© The NASDAQ OMX Group, Inc. • 2013

3.3.15.4 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.

3.3.15.5 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.	-

3.3.15.6 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
```

© The NASDAQ OMX Group, Inc. • 2013

```
        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
    }
}
```

3.3.15.7 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.

3.3.16 II17 [Preliminary Settlement Prices QUERY]

3.3.16.1 Fingerprint

QUERY properties	
transaction type	II17
calling sequence	omniapi_query_ex
struct name	query_prel_settlement
facility	EP0
partitioned	false
answers	IA17

ANSWER properties	
transaction type	IA17
struct name	answer_prel_settlement
segmented	true

3.3.16.2 Purpose

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

© The NASDAQ OMX Group, Inc. • 2013

3.3.16.3 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
}
```

3.3.16.4 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.

3.3.16.5 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.

© The NASDAQ OMX Group, Inc. • 2013

3.3.16.6 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
   }
}
```

3.3.16.7 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.

3.3.17 IQ12 [Total Equilibrium Prices QUERY]

3.3.17.1 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

© The NASDAQ OMX Group, Inc. • 2013

3.3.17.2 Purpose

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

3.3.17.3 Structure

The IQ12 QUERY has the following structure:

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

3.3.17.4 Usage and conditions

Series

must be filled with any valid series.

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

3.3.17.5 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.

3.3.17.6 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
    }
}
```

3.3.17.7 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.

3.3.18 IQ18 [Total Volumes and Prices VIQ]

3.3.18.1 Fingerprint

VIQ properties	
transaction type	IQ18
calling sequence	omniapi_query_ex

© The NASDAQ OMX Group, Inc. • 2013

VIQ 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.
facility	EP0
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

3.3.18.2 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.

3.3.18.3 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
   }
}
```

3.3.18.4 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 NASDAQ OMX Group, Inc. • 2013

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.

3.3.18.5 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:

© The NASDAQ OMX Group, Inc. • 2013

- 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.

3.3.18.6 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.

3.3.18.7 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_trd  // Named struct no: 33036
         struct market_info_omfi  // Named struct no: 33047
         struct ob_levels_closing  // Named struct no: 33031
      }
   }
}
```

3.3.18.8 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.

3.3.18.9 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

© The NASDAQ OMX Group, Inc. • 2013

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.

Market Info, Trade Report

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

Usage of fields in this structure:

© The NASDAQ OMX Group, Inc. • 2013

Last Trade Report Price	The value of MIN_INT is used to indicate an undefined value while binary zero indicates a price of zero.

3.3.18.10 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

© The NASDAQ OMX Group, Inc. • 2013

- 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

3.3.19 IQ19 [Total Volumes and Prices VIQ]

3.3.19.1 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

© The NASDAQ OMX Group, Inc. • 2013

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

3.3.19.2 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.

3.3.19.3 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
   }
}
```

3.3.19.4 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 NASDAQ OMX Group, Inc. • 2013

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.

3.3.19.5 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.)

© The NASDAQ OMX Group, Inc. • 2013

Block Size is not used and should be zero filled.

3.3.19.6 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.

3.3.19.7 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_trd  // Named struct no: 33036
         struct ob_levels_closing  // Named struct no: 33031
      }
   }
}
```

3.3.19.8 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

© The NASDAQ OMX Group, Inc. • 2013

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.

3.3.19.9 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

© The NASDAQ OMX Group, Inc. • 2013

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.

Market Info, Trade Report

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

Usage of fields in this structure:

Last Trade Report Price	The value of MIN_INT is used to indicate an undefined value while binary zero indicates a price of zero.

3.3.19.10 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.

© The NASDAQ OMX Group, Inc. • 2013

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)

© The NASDAQ OMX Group, Inc. • 2013

- 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

3.3.20 IQ42 [Trade Statistics QUERY]

3.3.20.1 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

3.3.20.2 Purpose

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

3.3.20.3 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
```


© The NASDAQ OMX Group, Inc. • 2013

```
    char[8] date_s  // Date
    char[2] filler_2_s  // Filler
}
```

3.3.20.4 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**.

3.3.20.5 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.

3.3.20.6 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
```

© The NASDAQ OMX Group, Inc. • 2013

```
        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
    }
}
```

3.3.20.7 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.

3.3.21 IQ49 [Price Median VIQ]

3.3.21.1 Fingerprint

VIQ properties	
transaction type	IQ49
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

© The NASDAQ OMX Group, Inc. • 2013

VIQ properties	
answers	IA49

VIA properties	
transaction type	IA49
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

3.3.21.2 Related Messages

BO49

3.3.21.3 Purpose

This query is used to retrieve Market by Median Bid and Ask.

3.3.21.4 Structure

The IQ49 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 price_median_id  // Named struct no: 33070
   }
}
```

3.3.21.5 Answer Structure

The IA49 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 price_median_id  // Named struct no: 33070
   Sequence {
      struct sub_item_hdr
      Choice {
         struct price_median  // Named struct no: 33071
      }
   }
}
```

© The NASDAQ OMX Group, Inc. • 2013

3.3.22 TQ1 [Historical Spread QUERY]

3.3.22.1 Fingerprint

QUERY properties	
transaction type	TQ1
calling sequence	omniapi_query_ex
struct name	query_spread_chk
facility	EP0
partitioned	false
answers	TA1

ANSWER properties	
transaction type	TA1
struct name	answer_spread_chk
segmented	true

3.3.22.2 Related Messages

None.

3.3.22.3 Purpose

The TQ1 query is used to download BBO (best bid and offer) data.

3.3.22.4 Structure

The TQ1 QUERY has the following structure:

```
struct query_spread_chk {
   struct transaction_type
   struct series  // Named struct no: 50000
   INT32_T start_time  // START_TIME
   INT32_T end_time  // END_TIME
   UINT16_T segment_number_n  // Segment Number
   char[2] filler_2_s  // Filler
}
```

3.3.22.5 Usage and Conditions

The download is done by specifying:

- An instrument of interest
- A start date

© The NASDAQ OMX Group, Inc. • 2013

- An end date
- A Segment number

The client should start with Segment Number 1 and increase it by one until Segment Number 0 is returned in the answer. At this point the client has received all BBO for given instrument within the specified time frame.

The query can be sent for current business day and a limited number of historical dates. Number of historical dates available is defined by the exchange.

The time format used is number of seconds since 1'st of January 1970.

3.3.22.6 Return Codes

cstatus	txstat	rcvbuf
Successful	Normal	List of BBO items
Transaction aborted	BADSEG - Segment number can not be Zero in an input query	–
Transaction aborted …	...	–

3.3.22.7 Answer Structure

The TA1 ANSWER has the following structure:

```
struct answer_spread_chk {
   struct transaction_type
   UINT16_T segment_number_n  // Segment Number
   UINT16_T items_n  // Items
   Array ITEM [max no: 100] {
      struct answer_spread {
         struct series  // Named struct no: 50000
         INT32_T best_bid_i  // BEST_BID_I
         INT32_T best_ask_i  // BEST_ASK_I
         INT32_T timestamp_best_bid  // TIMESTAMP_BEST_BID
         INT32_T timestamp_best_ask  // TIMESTAMP_BEST_ASK
      }
   }
}
```

3.3.22.8 Answer, comments

The answer returns a list of BBO items. Each BBO item includes the Bid and Ask price and the time when this BBO was established.

© The NASDAQ OMX Group, Inc. • 2013

3.3.23 TR70 [Trade Ticker QUERY]

3.3.23.1 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

3.3.23.2 Related Messages

BD70, BD71, TR71

3.3.23.3 Purpose

This query is used for recovering BD70.

3.3.23.4 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
}
```

3.3.23.5 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:

© The NASDAQ OMX Group, Inc. • 2013

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.

3.3.23.6 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 fixed_income_trade_ticker  // Named struct no: 34404
         struct half_trade_ticker  // Named struct no: 34405
      }
   }
}
```

3.3.23.7 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.

© The NASDAQ OMX Group, Inc. • 2013

3.3.24 TR71 [Amended Trades QUERY]

3.3.24.1 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

3.3.24.2 Related Messages

BD70, BD71, TR70

3.3.24.3 Purpose

This query is used for recovering BD71.

3.3.24.4 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
}
```

3.3.24.5 Answer Structure

The TA71 VIA has the following structure:

```
    struct answer_next_series_hdr {
       struct transaction_type
```


© The NASDAQ OMX Group, Inc. • 2013

```
      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
         }
      }
   }
```

3.4 Market Status

3.4.1 BI1 [Resumption and Suspension of Trading BROADCAST]

3.4.1.1 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

3.4.1.2 Purpose

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

3.4.1.3 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
}
```

3.4.2 BI41 [Instrument Status Information BROADCAST]

3.4.2.1 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

3.4.2.2 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.

3.4.2.3 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
   }
}
```

3.4.2.4 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 NASDAQ OMX Group, Inc. • 2013

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

© The NASDAQ OMX Group, Inc. • 2013

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

3.4.3 BI94 [Planned Instrument Session Info BROADCAST]

3.4.3.1 Fingerprint

BROADCAST properties	
transaction type	BI94
calling sequence	omniapi_read_event_ext_ex or omniapi_read_event_block
struct name	planned_inst_session_info
info type	general

3.4.3.2 Related Messages

UQ19

3.4.3.3 Purpose

This broadcast informs about upcoming session state changes when the normal Trading Session is abandoned.

3.4.3.4 Structure

The BI94 BROADCAST has the following structure:

```
struct planned_inst_session_info {
   struct broadcast_type
   UINT16_T items_n  // Items
   char[2] filler_2_s  // Filler
   Array ITEM [max no: 10] {
      struct series  // Named struct no: 50000
      INT64_T quantity_limit_q  // Quantity limit used for One sided auction
      INT32_T reference_price_i  // REFERENCE_PRICE_I
      INT32_T net_price_for_settlement_i  // Net Price for Settlement
      UINT16_T session_order_n  // Session Order
      UINT16_T state_number_n  // Trading State Number
      UINT16_T state_level_e  // Level
      char[8] next_planned_start_date_s  // Planned Start Date, Next
      char[6] next_planned_start_time_s  // Planned Start Time, Next
      char[8] date_settlement_s  // Date, Settlement
      char[80] warning_msg_s  // Warning Message
   }
}
```


© The NASDAQ OMX Group, Inc. • 2013

3.4.4 BI95 [One Sided Auction Result BROADCAST]

3.4.4.1 Fingerprint

BROADCAST properties	
transaction type	BI95
calling sequence	omniapi_read_event_ext_ex or omniapi_read_event_block
struct name	one_sided_auction_result
info type	dedicated

3.4.4.2 Related Messages

MQ95

3.4.4.3 Purpose

This broadcast contains the result from the one sided auction.

3.4.4.4 Structure

The BI95 BROADCAST has the following structure:

```
struct one_sided_auction_result {
   struct broadcast_type
   struct series  // Named struct no: 50000
   struct timestamp  // Of type: TIME_SPEC
   INT32_T equilibrium_price  // Premium ; Of type: PREMIUM_I
   INT32_T high_price  // Premium ; Of type: PREMIUM_I
   INT32_T low_price  // Premium ; Of type: PREMIUM_I
   INT32_T vwap_match_price  // Premium ; Of type: PREMIUM_I
   INT64_T respondent_quantity  // Quantity ; Of type: QUANTITY_I
   INT64_T matching_quantity  // Quantity ; Of type: QUANTITY_I
   INT64_T imbalance_quantity  // Quantity ; Of type: QUANTITY_I
   UINT16_T respondent_order_count  // Number of orders ; Of type:
NUMBER_OF_ORDERS_N
   UINT16_T matching_order_count  // Number of orders ; Of type:
NUMBER_OF_ORDERS_N
   UINT8_T is_preliminary_c  // Is Preliminary
   char[3] filler_3_s  // Filler
}
```

3.4.5 UC19 [Request Auction TRANSACTION]

3.4.5.1 Fingerprint

TRANSACTION properties	
transaction type	UC19
calling sequence	omniapi_tx_ex
struct name	request_auction
facility	EP0
partitioned	false

3.4.5.2 Related Messages

UC20

3.4.5.3 Purpose

The purpose of this transaction is for an Issuer, or someone acting on behalf of an Issuer, to request an Issuing or a Buy Back Auction.

3.4.5.4 Structure

The UC19 TRANSACTION has the following structure:

```
struct request_auction {
   struct transaction_type
   struct series  // Named struct no: 50000
   INT64_T quantity_limit_q  // Quantity limit used for One sided auction
   INT32_T reference_price_i  // REFERENCE_PRICE_I
   INT32_T net_price_for_settlement_i  // Net Price for Settlement
   UINT8_T auction_type_c  // Auction Type
   UINT8_T book_transparancy_c  // Book Transparancy
   char[8] date_settlement_s  // Date, Settlement
   char[8] auction_uncross_date_s  // Auction Uncross Date
   char[6] auction_uncross_time_s  // Auction Uncross Time
}
```

3.4.5.5 Usage and conditions

Series

Must be a valid Instrument-Series, for which this transaction is allowed.

Issuer Trading Code

© The NASDAQ OMX Group, Inc. • 2013

If transaction is sent on behalf of the issuer, this field holds the issuer.

Auction Type

Must be set to Issuing / Buy Back.

Book Transparency

Must be set to Open / Hidden.

Auction Uncross Date Time

Mandatory. This is when the uncross will be made.

Settlement Date

Optional. This is the settlement date which will be broadcast to participants.

Net price for settlement

Optional. This is the net price used when calculating the settlement price and it will be broadcast to participants.

3.4.6 UC20 [Finish Auction TRANSACTION]

3.4.6.1 Fingerprint

TRANSACTION properties	
transaction type	UC20
calling sequence	omniapi_tx_ex
struct name	finish_auction
facility	EP0
partitioned	false

3.4.6.2 Related Messages

UC19

3.4.6.3 Purpose

The purpose of this transaction is for an Issuer, or someone acting on behalf of an Issuer, to request a canceling of an ongoing issuing or buy back auction.

3.4.6.4 Structure

The UC20 TRANSACTION has the following structure:

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

© The NASDAQ OMX Group, Inc. • 2013

```
}
```

3.4.6.5 Usage and conditions

Series

Must be a valid Instrument Series, for which this transaction is allowed.

Trading Code

Normally, this is the issuer.

Issuer Trading Code

If transaction is sent on behalf of the issuer, the Trading Code field holds the sender, and this field holds the issuer.

3.4.7 UQ15 [Instrument Status QUERY]

3.4.7.1 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

3.4.7.2 Purpose

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

3.4.7.3 Structure

The UQ15 QUERY has the following structure:

```
struct query_instrument_status {
```


© The NASDAQ OMX Group, Inc. • 2013

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

3.4.7.4 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.

© The NASDAQ OMX Group, Inc. • 2013

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

3.4.7.5 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
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.

3.4.7.6 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
   }
}
```

3.4.7.7 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

© The NASDAQ OMX Group, Inc. • 2013

	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.

3.4.8 UQ19 [Planned Instrument Session QUERY]

3.4.8.1 Fingerprint

QUERY properties	
transaction type	UQ19
calling sequence	omniapi_query_ex
struct name	query_planned_inst_session
facility	EP0
partitioned	false
answers	UA19

ANSWER properties	
transaction type	UA19
struct name	answer_planned_inst_session
segmented	true

3.4.8.2 Related Messages

BI94

3.4.8.3 Purpose

The purpose of this query is to get information about upcoming session state changes for instrument series which do not follow the normal Trading Session schedule.

3.4.8.4 Structure

The UQ19 QUERY has the following structure:

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

3.4.8.5 Usage and conditions

Series

Must be filled down to Instrument Type.

3.4.8.6 Answer Structure

The UA19 ANSWER has the following structure:

```
struct answer_planned_inst_session {
   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
      INT64_T quantity_limit_q  // Quantity limit used for One sided auction
      INT32_T reference_price_i  // REFERENCE_PRICE_I
      INT32_T net_price_for_settlement_i  // Net Price for Settlement
      UINT16_T session_order_n  // Session Order
      UINT16_T state_number_n  // Trading State Number
      UINT16_T state_level_e  // Level
      char[8] next_planned_start_date_s  // Planned Start Date, Next
      char[6] next_planned_start_time_s  // Planned Start Time, Next
      char[8] date_settlement_s  // Date, Settlement
      char[80] warning_msg_s  // Warning Message
   }
}
```

3.5 Market Maker Messages

3.5.1 BL8 [Request with Volume BROADCAST]

3.5.1.1 Fingerprint

BROADCAST properties	
transaction type	BL8
calling sequence	omniapi_read_event_ext_ex or omniapi_read_event_block
struct name	dedic_quote_request_vol_info
info type	dedicated

© The NASDAQ OMX Group, Inc. • 2013

3.5.1.2 Related Messages

MC4, MI4

3.5.1.3 Purpose

The Dedicated Quote Request with Volume Info is sent after a valid Quote Request with Volume. The broadcast, unlike MI4, is sent when the Quote Request is supposed to be sent as a dedicated broadcast to either all Market Makers or only the responsible Market Makers. To whom a Quote Request with Volume Info should be sent, is configured on Instrument Type level in CDB. For more information on MI4, refer to that section.

3.5.1.4 Structure

The BL8 BROADCAST has the following structure:

```
struct dedic_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
}
```

3.5.1.5 Usage and Conditions

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.

3.5.2 BL22 [Dedicated Market Maker Alarm BROADCAST]

3.5.2.1 Fingerprint

BROADCAST properties	
transaction type	BL22
calling sequence	omniapi_read_event_ext_ex or omniapi_read_event_block
struct name	dedicated_mm_alarm_bl22
info type	dedicated

© The NASDAQ OMX Group, Inc. • 2013

3.5.2.2 Purpose

The Dedicated Designated Market Maker Alarm broadcast is sent if a customer has not fulfilled his responsibilities regarding price quotations.

3.5.2.3 Structure

The BL22 BROADCAST has the following structure:

```
struct dedicated_mm_alarm_bl22 {
   struct broadcast_type
   struct series  // Named struct no: 50000
   UINT32_T mmsup_status_u  // Alarm, Type
   UINT32_T alarm_status_u  // Alarm Status
   struct trading_code
   UINT32_T block_n  // Block Size
}
```

3.5.3 BO38 [Market Maker Protection Settings Information BROADCAST]

3.5.3.1 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

3.5.3.2 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.

3.5.3.3 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
}
```


© The NASDAQ OMX Group, Inc. • 2013

3.5.3.4 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.

3.5.4 LQ16 [Market Maker Underlying Price QUERY]

3.5.4.1 Fingerprint

QUERY properties	
transaction type	LQ16
calling sequence	omniapi_query_ex
struct name	query_mmsup_uv
facility	EP0
partitioned	false
answers	LA16

ANSWER properties	
transaction type	LA16
struct name	answer_mmsup_uv
segmented	false

3.5.4.2 Related Messages

MI5

3.5.4.3 Purpose

This query returns the current At-The-Money (ATM) value of the strike price that is used in the Market Maker Supervision.

3.5.4.4 Structure

The LQ16 QUERY has the following structure:

```
struct query_mmsup_uv {
   struct transaction_type
   struct series  // Named struct no: 50000
   struct ul_series
```

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

3.5.4.5 Usage and conditions

Series

is used for retrieving answers for the segmented query. In the initial query Series is zeroed.

Segment Number

is used for retrieving answers for the segmented query. In the initial query Segment Number is set to 1.

Commodity

may be zeroed (all underlyings) or used for selecting a specific commodity code or future (the future binary code).

is always set to zero.

3.5.4.6 Answer Structure

The LA16 ANSWER has the following structure:

```
struct answer_mmsup_uv {
   struct transaction_type
   struct series_next
   UINT16_T segment_number_n  // Segment Number
   UINT16_T items_n  // Items
   Array ITEM [max no: 1750] {
      struct ul_series
      INT32_T atm_price_i  // Price, At-The-Money
      INT32_T underlying_price_i  // Price, Underlying
      char[8] yyyymmdd_s  // Date
      char[6] hhmmss_s  // Time, External
      char[2] filler_2_s  // Filler
   }
}
```

3.5.4.7 Answer, comments

After a successful LQ16 transaction, a list of items with underlying, ATM value, and date is returned to the sender.The Client should confirm to the following logic in order to download data for all items:

1. From the answer structure, copy the Next Series to the series field in subsequent query. If the Segment Number in the answer is greater than zero the value should be incremented by 1 and copied to the Segment Number in the subsequent query, otherwise (received Segment Number is zero) the value of one should be copied.
2. Repeat step 1 until the Segment Number in the answer is zero. When series_next is zero filled, the last ATM value for the last partition is received.

© The NASDAQ OMX Group, Inc. • 2013

One ATM value is distributed per underlying and includes also a timestamp with the last update time (UTC).

3.5.5 MC4 [Quote Request with Volume TRANSACTION]

3.5.5.1 Fingerprint

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

3.5.5.2 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..

3.5.5.3 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
}
```

3.5.5.4 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.

3.5.5.5 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
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.

3.5.6 MI3 [Market established BROADCAST]

3.5.6.1 Fingerprint

BROADCAST properties	
transaction type	MI3
calling sequence	omniapi_read_event_ext_ex or omniapi_read_event_block
struct name	market_established
info type	dedicated

3.5.6.2 Purpose

When a trader sends a Crossing Quote request a subsequent Market Established broadcast is distributed if:

- the market is already established **or**
- a Market Maker sends a valid quote within a predefined time.

3.5.6.3 Structure

The MI3 BROADCAST has the following structure:

© The NASDAQ OMX Group, Inc. • 2013

```
struct market_established {
   struct broadcast_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
   INT32_T bid_premium_i  // Bid Premium
   INT32_T ask_premium_i  // Ask Premium
}
```

3.5.7 MI4 [Quote Request with Volume Information BROADCAST]

3.5.7.1 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

3.5.7.2 Related Messages

MC4, BL8

3.5.7.3 Purpose

The Quote Request with Volume Info is sent after a valid Quote Request with Volume. The broadcast, unlike BL8, is sent when the Quote Request is supposed to be sent to the entire market.

3.5.7.4 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
}
```

© The NASDAQ OMX Group, Inc. • 2013

3.5.7.5 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.

3.5.8 MI5 [Market Maker Underlying Price BROADCAST]

3.5.8.1 Fingerprint

BROADCAST properties	
transaction type	MI5
calling sequence	omniapi_read_event_ext_ex or omniapi_read_event_block
struct name	mmsup_uv
info type	dedicated

3.5.8.2 Related Messages

LQ16

3.5.8.3 Purpose

The At-The-Money (ATM) value of the strike price, that is used in the Supervision, is distributed with this broadcast. When the ATM value is changed in the Supervision a new broadcast is sent

3.5.8.4 Structure

The MI5 BROADCAST has the following structure:

```
struct mmsup_uv {
   struct broadcast_type
   UINT16_T items_n  // Items
   char[2] filler_2_s  // Filler
   Array ITEM [max no: 39] {
      struct ul_series
      INT32_T atm_price_i  // Price, At-The-Money
      INT32_T underlying_price_i  // Price, Underlying
      char[8] yyyymmdd_s  // Date
      char[6] hhmmss_s  // Time, External
```


© The NASDAQ OMX Group, Inc. • 2013

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

3.5.8.5 Usage and Conditions

ATM value

is distributed per underlying and includes also a timestamp with the last update time (UTC).

is always set to zero.

3.6 Trade and Position Management

3.6.1 BD6 [Dedicated Trade Information VIB]

3.6.1.1 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

3.6.1.2 Related Messages

CQ10

CQ11

3.6.1.3 Purpose

This is a dedicated trade broadcast distributed to the participants in real-time. The contents of the broadcast is exchange specific.

> **Note:** BD6 replaces BD4.

3.6.1.4 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
      struct cl_trade_trade_report_api  // Named struct no: 67
      struct cl_trade_fixed_income_api  // Named struct no: 68
      struct cl_trade_cancel_trade_api  // Named struct no: 70
   }
}
```

3.6.1.5 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)

Deferred Publication

In case deferred publication time is set to end-of-day in CDB, the resulting BD6 from a trade report with deferred publication will contain deferred_time_n=65535.

3.6.1.6 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

© The NASDAQ OMX Group, Inc. • 2013

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.

3.6.2 BD18 [Dedicated Delivery BROADCAST]

3.6.2.1 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

3.6.2.2 Related Messages

CQ52, CQ53

3.6.2.3 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.

3.6.2.4 Structure

The BD18 BROADCAST has the following structure:

```
struct directed_delivery {
   struct broadcast_type
   struct cl_delivery_api
}
```

3.6.2.5 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

© The NASDAQ OMX Group, Inc. • 2013

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**.

© The NASDAQ OMX Group, Inc. • 2013

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. If this number is above 200, this indicates that the delivery item is informational only, i.e. will not be included in any further settlement processing. The type of settlement is found by taking the class number and subtracting 200, so that if class-number is e.g. 202, this is an informational (200) clearing fee (2).

If this number is between 100 and 200, this indicates that the delivery item will be accumulated for settlement at a later date, i.e. not necessarily the settlement date specified in the delivery. The type of settlement is found by taking the class number and subtracting 100, so that if class-number is e.g. 102, this is a clearing fee (2) which will accrue (100).

3.6.3 BD29 [Directed Give Up BROADCAST]

3.6.3.1 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

© The NASDAQ OMX Group, Inc. • 2013

3.6.3.2 Related Messages

CQ61, CQ76

3.6.3.3 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.

3.6.3.4 Structure

The BD29 BROADCAST has the following structure:

```
struct directed_give_up {
   struct broadcast_type
   struct cl_give_up_api
}
```

3.6.3.5 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.”

© The NASDAQ OMX Group, Inc. • 2013

- 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.

3.6.4 BD39 [Dedicated Trade Change Information BROADCAST]

3.6.4.1 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

© The NASDAQ OMX Group, Inc. • 2013

3.6.4.2 Related Messages

Dedicated Trade Information Broadcast and CQ39

3.6.4.3 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.

3.6.4.4 Structure

The BD39 BROADCAST has the following structure:

```
struct directed_trade_change {
   struct broadcast_type
   struct cl_trade_change_api {
      struct series  // Named struct no: 50000
      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
   }
}
```

3.6.4.5 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.

3.6.5 BD41 [DC Holding Trade VIB]

3.6.5.1 Fingerprint

VIB properties	
transaction type	BD41
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.

© The NASDAQ OMX Group, Inc. • 2013

VIB properties	
info type	dedicated

3.6.5.2 Related Messages

CQ51

3.6.5.3 Purpose

This broadcast returns information on deals on hold in the market.

3.6.5.4 Structure

The BD41 VIB has the following structure:

```
struct broadcast_hdr
Sequence {
   struct sub_item_hdr
   Choice {
      struct dc_hold_deal_external  // Named struct no: 63
      struct dc_hold_trade_external  // Named struct no: 64
   }
}
```

3.6.5.5 Usage and conditions

When an On Hold deal is disapproved, a new BD41 with **State**= Rejected is sent.

3.6.6 BI27 [Clearing message BROADCAST]

3.6.6.1 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

3.6.6.2 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.

3.6.6.3 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
   }
}
```

3.6.6.4 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.

3.6.7 BI28 [Bond Index Parameters BROADCAST]

3.6.7.1 Fingerprint

BROADCAST properties	
transaction type	BI28
calling sequence	omniapi_read_event_ext_ex or omniapi_read_event_block
struct name	bond_index_params
info type	general

3.6.7.2 Purpose

This subscription returns duration and internal interest rate for a bond index underlying. This information is normally produced outside the Exchange and redistributed in the API.

3.6.7.3 Structure

The BI28 BROADCAST has the following structure:

```
struct bond_index_params {
   struct broadcast_type
   UINT16_T commodity_n  // Commodity Code
   char[2] filler_2_s  // Filler
```


© The NASDAQ OMX Group, Inc. • 2013

```
    INT32_T duration_i  // Duration
    INT32_T internal_interest_rate_i  // Internal Interest Rate
}
```

3.6.7.4 Usage and conditions

The availability of this information depends on the Exchange policy.

3.6.8 CB3 [Directed OTC Trade Report VIB]

3.6.8.1 Fingerprint

VIB properties	
transaction type	CB3
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

3.6.8.2 Purpose

This broadcast is a directed broadcast containing information about a trade report and it is sent to both the member and the counter party member.

> **Note:**
> This broadcast is deprecated and will be replaced by KB1.

3.6.8.3 Structure

The CB3 VIB has the following structure:

```
struct directed_trade_report {
   struct broadcast_type
   UINT8_T broadcast_reason_c  // Broadcast Reason
   char[3] filler_3_s  // Filler
   UINT16_T items_n  // Items
   UINT16_T size_n  // Size
}
Sequence {
   struct sub_item_hdr
   Choice {
      struct fi_trade_report  // Named struct no: 13
      struct fx_trade_report  // Named struct no: 7
      struct cash_trade_report  // Named struct no: 8
      struct agreement_trade_report  // Named struct no: 9
      struct ssi_trade_report  // Named struct no: 10
```

```
            struct equity_trade_report  // Named struct no: 12
            struct fra_trade_report  // Named struct no: 11
            struct fi_repo_trade_report  // Named struct no: 14
            struct ir_swap_trade_report  // Named struct no: 15
            struct xcur_swap_trade_report  // Named struct no: 16
            struct cash_transfer_group_otc  // Named struct no: 22
            struct cash_transfer_trade_report  // Named struct no: 23
            struct otc_clearing_info  // Named struct no: 83
        }
    }
```

3.6.8.4 Usage and Conditions

The broadcast is sent every time a field is changed.

3.6.9 CB146 [CL OTC Trade Operation Rejected VIB]

3.6.9.1 Fingerprint

VIB properties	
transaction type	CB146
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

3.6.9.2 Related Messages

CQ146

3.6.9.3 Purpose

This broadcast will be sent when a Trade Operation for an OTC Trade in other instruments than swaps or TM FRA’s, has been "Rejected" by the clearinghouse due to Clearinghouse Collateral Checks.

3.6.9.4 Structure

The CB146 VIB has the following structure:

```
struct bdx_cl_otc_trade_op_on_hold {
   struct broadcast_type
   UINT16_T items_n  // Items
   UINT16_T size_n  // Size
}
Sequence {
   struct sub_item_hdr
   Choice {
```


© The NASDAQ OMX Group, Inc. • 2013

```
        struct cl_otc_operation_info  // Named struct no: 95
        struct cl_otc_trade_operation  // Named struct no: 96
        struct risk_exposure_limit_vim  // Named struct no: 50010
    }
}
```

3.6.10 CC10 [Rectify Exercise TRANSACTION]

3.6.10.1 Fingerprint

TRANSACTION properties	
transaction type	CC10
calling sequence	omniapi_tx_ex
struct name	cl_rectify_exercise
facility	EP0
partitioned	false

3.6.10.2 Purpose

This transaction is used to rectify an exercise request.

3.6.10.3 Structure

The CC10 TRANSACTION has the following structure:

```
struct cl_rectify_exercise {
   struct transaction_type
   struct series  // Named struct no: 50000
   INT32_T exercise_number_i  // Exercise, Request Number
}
```

3.6.10.4 Usage and conditions

Only exercise request can be rectified. General automatic exercise cannot be rectified.

3.6.11 CC11 [Cancel Holding Rectify Trade TRANSACTION]

3.6.11.1 Fingerprint

TRANSACTION properties	
transaction type	CC11
calling sequence	omniapi_tx_ex
struct name	confirm_rectify_t
facility	EP3

© The NASDAQ OMX Group, Inc. • 2013

TRANSACTION properties	
partitioned	false

3.6.11.2 Related Messages

CQ14, CQ15

3.6.11.3 Purpose

This transaction is used to cancel a previously sent rectify trade request.

3.6.11.4 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
}
```

3.6.11.5 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.

3.6.12 CC12 [Cancel Holding Rectify Deal TRANSACTION]

3.6.12.1 Fingerprint

TRANSACTION properties	
transaction type	CC12
calling sequence	omniapi_tx_ex
struct name	confirm_rectify_d
facility	EP3
partitioned	false

© The NASDAQ OMX Group, Inc. • 2013

3.6.12.2 Related Messages

CQ16, CQ17

3.6.12.3 Purpose

This transaction is used to cancel a previously sent rectify deal request.

3.6.12.4 Structure

The CC12 TRANSACTION has the following structure:

```
struct confirm_rectify_d {
   struct transaction_type
   struct series  // Named struct no: 50000
   INT64_T rectify_deal_number_q  // Rectify Deal Number
   UINT8_T operation_c  // Operation
   UINT8_T confirm_reject_c  // Confirm or Reject
   char[2] filler_2_s  // Filler
}
```

3.6.12.5 Usage and conditions

Series

must be set to Series from the Original deal.

Rectify Deal Number

must be set to the rectify deal number identifying the deal rectification in question.

Operation

is set to Delete.

Confirm or Reject

is set to Reject.

3.6.13 CC13 [Exercise Request TRANSACTION]

3.6.13.1 Fingerprint

TRANSACTION properties	
transaction type	CC13
calling sequence	omniapi_tx_ex
struct name	exercise_req
facility	EP3

TRANSACTION properties	
partitioned	false

3.6.13.2 Purpose

The purpose of this transaction is to request an exercise.

3.6.13.3 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
}
```

3.6.13.4 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.

3.6.14 CC14 [Deny Exercise Request TRANSACTION]

3.6.14.1 Fingerprint

TRANSACTION properties	
transaction type	CC14
calling sequence	omniapi_tx_ex
struct name	set_deny_exercise
facility	EP3
partitioned	false

3.6.14.2 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.

© The NASDAQ OMX Group, Inc. • 2013

3.6.14.3 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
}
```

3.6.15 CC15 [Cancel Exercise Request TRANSACTION]

3.6.15.1 Fingerprint

TRANSACTION properties	
transaction type	CC15
calling sequence	omniapi_tx_ex
struct name	annul_exercise_req
facility	EP3
partitioned	false

3.6.15.2 Related Messages

CQ21

3.6.15.3 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**.

3.6.15.4 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
}
```

3.6.15.5 Usage and conditions

Series

© The NASDAQ OMX Group, Inc. • 2013

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.

3.6.16 CC19 [Cancel Trade TRANSACTION]

3.6.16.1 Fingerprint

TRANSACTION properties	
transaction type	CC19
calling sequence	omniapi_tx_ex
struct name	cancel_trade
facility	EP3
partitioned	false

3.6.16.2 Purpose

This transaction is used for canceling your own part of a trade. The trade is not actually canceled before both parties enter a trade cancellation transaction.

3.6.16.3 Structure

The CC19 TRANSACTION has the following structure:

```
struct cancel_trade {
   struct transaction_type
   struct series  // Named struct no: 50000
   UINT8_T instance_c  // Instance, Number
   UINT8_T bought_or_sold_c  // Bought or Sold
   char[2] filler_2_s  // Filler
   INT32_T trade_number_i  // Trade Number
   INT32_T ext_seq_nbr_i  // External Clearinghouse, Sequence Number
}
```

3.6.16.4 Usage and Conditions

Identification of Trade

Either submit:

- **Series**, **Sequence Number, External Clearinghouse**, **Bought or Sold**, and **Instance Number**, or:
- **Series, Trade Number** and **Instance Number**.

© The NASDAQ OMX Group, Inc. • 2013

Cancel Trade

When the system has received a Cancel Trade transaction from each of the two parties involved in a trade, the system will, if the time limit has been complied with, roll back the trade and send out reversing trade messages (BD6) to the two parties as a confirmation. A public cancellation message will also be distributed (BD71).

3.6.17 CC38 [Confirm Give up Request TRANSACTION]

3.6.17.1 Fingerprint

TRANSACTION properties	
transaction type	CC38
calling sequence	omniapi_tx_ex
struct name	confirm_give_up_request
facility	EP3
partitioned	false

3.6.17.2 Related Messages

CQ61

3.6.17.3 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.

3.6.17.4 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
   }
}
```

3.6.17.5 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.

3.6.17.6 Return Codes

Even if a Confirm Give Up transaction is accepted by the system, it is possible that it will not be executed immediately. The below statuses give more information:

Completion	Cstatus	TxStat (reason code)
The Give Up operation is performed.	Successful	CL_OMN_NORMAL
The Give Up operation is subject to collateral checks. If rejected, please refer to broadcast BD29. If approved, please refer to broadcast BD29 and BD6.	Successful	CL_OMN_COLLCHECK

Please refer to the *OMex System's Error Messages* for details about why transactions are aborted.

3.6.18 CC40 [Reject Give up Request TRANSACTION]

3.6.18.1 Fingerprint

TRANSACTION properties	
transaction type	CC40
calling sequence	omniapi_tx_ex
struct name	reject_give_up_request
facility	EP3
partitioned	false

3.6.18.2 Related Messages

CQ61

© The NASDAQ OMX Group, Inc. • 2013

3.6.18.3 Purpose

This transaction is used to reject a give-up request. Use CQ61 to retrieve information on give-up trades in holding state.

3.6.18.4 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
}
```

3.6.18.5 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.

3.6.19 CC45 [Change account state TRANSACTION]

3.6.19.1 Fingerprint

TRANSACTION properties	
transaction type	CC45
calling sequence	omniapi_tx_ex
struct name	change_account_state
facility	EP5
partitioned	false

3.6.19.2 Purpose

The purpose of this transaction is to change the account state of an account.

3.6.19.3 Structure

The CC45 TRANSACTION has the following structure:

© The NASDAQ OMX Group, Inc. • 2013

```
struct change_account_state {
   struct transaction_type
   struct series  // Named struct no: 50000
   struct account
   UINT8_T acc_state_c  // Account State
   char[3] filler_3_s  // Filler
}
```

3.6.19.4 Usage and Conditions

CC45 supports the following account state changes:

- From Registered to Active/Inactive
- From Active to Inactive

Series

must be set to zero.

Account state

Possible values are:

- 2 (Inactive)
- 3 (Active)

3.6.20 CC51 [Deny Real Time TRANSACTION]

3.6.20.1 Fingerprint

TRANSACTION properties	
transaction type	CC51
calling sequence	omniapi_tx_ex
struct name	set_deny_exercise
facility	EP3
partitioned	false

3.6.20.2 Purpose

The purpose of this transaction is to close down a long position in real time, which in other case would be exercised in an automatic exercise request. The transaction will close down the chosen long position and will randomly pick a short position within the same series, which also will be closed down.

3.6.20.3 Structure

The CC51 TRANSACTION has the following structure:

© The NASDAQ OMX Group, Inc. • 2013

```
struct set_deny_exercise {
   struct transaction_type
   struct series  // Named struct no: 50000
   struct account
   INT64_T deny_exercise_q  // Deny Exercise
}
```

3.6.20.4 Usage and conditions

The request must be done on a long position of an option.

3.6.21 CC54 [Cancel OTC Trade Report TRANSACTION]

3.6.21.1 Fingerprint

TRANSACTION properties	
transaction type	CC54
calling sequence	omniapi_tx_ex
struct name	cancel_trade_report
facility	EP0
partitioned	false

3.6.21.2 Purpose

The purpose of this transaction is to cancel a trade report.

> **Note:**
> This transaction is deprecated and will be replaced by KC2.

3.6.21.3 Structure

The CC54 TRANSACTION has the following structure:

```
struct cancel_trade_report {
   struct transaction_type
   struct series  // Named struct no: 50000
   UINT64_T trade_report_nbr_q  // Trade report number
   char[32] name_s  // Name
   UINT8_T confirm_reject_c  // Confirm or Reject
   char[3] filler_3_s  // Filler
}
```

3.6.21.4 Usage and Conditions

If a Trade Report is in a Pending Cancellation sub state, the pending cancellation can be rejected. Either side of the Trade Report can reject a pending cancellation on its own Trade Report. This means that a user

© The NASDAQ OMX Group, Inc. • 2013

can reject his outgoing cancellation because he changed his mind or made a typing error. The user receiving an incoming cancellation can also reject this if he doesn't wish to cancel the Trade Report.

No fields can be edited.

When cancelling an equity trade report, different conditions apply depending on the current state of the trade report.

UnMatched	The user who entered the report may cancel the trade report without restrictions.
matched	Once matched the trade report cannot be cancelled. Instead it can be fully terminated.

3.6.21.5 Return Codes

Even if a Cancel OTC Trade Report transaction is accepted by the system, it is possible that it will not be executed immediately. The below statuses give more information:

Completion	Cstatus	TxStat (reason code)
The deal is subject to collateral checks. Please refer to KB1 broadcast for result.	Successful	OTC_COLLCHECK

Please refer to the *OMex System's Error Messages* for details about why transactions are aborted.

3.6.22 CC57 [Confirm/ Reject OTC Trade Report TRANSACTION]

3.6.22.1 Fingerprint

TRANSACTION properties	
transaction type	CC57
calling sequence	omniapi_tx_ex
struct name	confirm_reject_trade_report
facility	EP0
partitioned	false

3.6.22.2 Purpose

The purpose of this transaction is to confirm/reject a trade report.

3.6.22.3 Structure

The CC57 TRANSACTION has the following structure:

```
struct confirm_reject_trade_report {
   struct transaction_type
   struct series  // Named struct no: 50000
   UINT64_T party_trade_report_nbr_q  // Party trade report number
   struct account
```


© The NASDAQ OMX Group, Inc. • 2013

```
    char[32] name_s  // Name
    char[32] passthrough_s  // Passthrough Information
    char[80] participant_info_s  // Participant Info
    char[120] buy_si_s  // Buy Settlement Instruction
    char[24] cash_account_s  // Account, Cash
    char[24] security_account_s  // Account, Security
    UINT8_T confirm_reject_c  // Confirm or Reject
    UINT8_T settle_domestic_currency_c  // Settlement Domestic Currency
    UINT8_T settle_foreign_currency_c  // Settlement Foreign Currency
    UINT8_T use_ssi_c  // Use SSI
}
```

3.6.22.4 Usage and Conditions

A trade report can be entered in two ways; either via the Enter Trade Report transaction or via this confirm transaction if the user is the counterpart.

No fields can be edited.

If the Trade Report is rejected, it will be set to Rejected state.

3.6.22.5 Return Codes

Even if a Confirm OTC Trade Report transaction is accepted by the system, it is possible that it will not be executed immediately. The below statuses give more information:

Completion	Cstatus	TxStat (reason code)
The deal is novated.	Successful	OTC_NORMAL
The deal is subject to collateral checks. Please refer to KB1 broadcast for result.	Successful	OTC_COLLCHECK

Please refer to the *OMex System's Error Messages* for details about why transactions are aborted.

3.6.23 CC63 [Rectify FRA Trade Report TRANSACTION]

3.6.23.1 Fingerprint

TRANSACTION properties	
transaction type	CC63
calling sequence	omniapi_tx_ex
struct name	rectify_fra_trade_report
facility	EP0
partitioned	false

3.6.23.2 Purpose

The purpose of this transaction is to rectify an FRA trade report.

© The NASDAQ OMX Group, Inc. • 2013

3.6.23.3 Structure

The CC63 TRANSACTION has the following structure:

```
struct rectify_fra_trade_report {
   struct transaction_type
   struct series  // Named struct no: 50000
   struct fra  // Named struct no: 85
   UINT64_T trade_report_nbr_q  // Trade report number
}
```

3.6.23.4 Usage and Conditions

Unmatched trade reports can be rectified without restriction.

Only non-matching fields can be rectified for Matched or Novated trade reports.

3.6.23.5 Return Codes

Even if a Rectify FRA Trade Report transaction is accepted by the system, it is possible that it will not be executed immediately. The below statuses give more information:

Completion	Cstatus	TxStat (reason code)
The deal is subject to collateral checks. Please refer to KB1 broadcast for result.	Successful	OTC_COLLCHECK

Please refer to the *OMex System's Error Messages* for details about why transactions are aborted.

3.6.24 CC68 [Rectify IR Swap Trade Report TRANSACTION]

3.6.24.1 Fingerprint

TRANSACTION properties	
transaction type	CC68
calling sequence	omniapi_tx_ex
struct name	rectify_ir_swap_trade_report
facility	EP0
partitioned	false

3.6.24.2 Purpose

The purpose of this transaction is to rectify an Interest Rate Swap Trade Report.

3.6.24.3 Structure

The CC68 TRANSACTION has the following structure:

© The NASDAQ OMX Group, Inc. • 2013

```
struct rectify_ir_swap_trade_report {
   struct transaction_type
   struct series  // Named struct no: 50000
   UINT64_T trade_report_nbr_q  // Trade report number
   struct ir_swap
   UINT16_T items_n  // Items
   char[2] filler_2_s  // Filler
   Array ITEM [max no: 500] {
      struct swap_flow
   }
}
```

3.6.24.4 Usage and Conditions

Unmatched trade reports can be rectified without restriction.

Only non-matching fields can be rectified for Matched or Novated trade reports.

3.6.24.5 Return Codes

Even if a Rectify IR Swap Trade Report transaction is accepted by the system, it is possible that it will not be executed immediately. The below statuses give more information:

Completion	Cstatus	TxStat (reason code)
The deal is subject to collateral checks. Please refer to KB1 broadcast for result.	Successful	OTC_COLLCHECK

Please refer to the *OMex System's Error Messages* for details about why transactions are aborted.

3.6.25 CC73 [Terminate Swap TRANSACTION]

3.6.25.1 Fingerprint

TRANSACTION properties	
transaction type	CC73
calling sequence	omniapi_tx_ex
struct name	terminate_swap
facility	EP0
partitioned	false

3.6.25.2 Purpose

The purpose of this transaction is to terminate a Swap Trade Report.

3.6.25.3 Structure

The CC73 TRANSACTION has the following structure:

© The NASDAQ OMX Group, Inc. • 2013

```
struct terminate_swap {
   struct transaction_type
   struct series  // Named struct no: 50000
   struct swap_termination {
      struct series  // Named struct no: 50000
      struct account
      char[32] name_s  // Name
      UINT64_T trade_report_nbr_q  // Trade report number
      char[8] termination_agree_date_s  // Termination Agree Date
      INT64_T notional_amount_q  // Notional amount
      INT64_T second_notional_amount_q  // Notional amount ; Of type:
NOTIONAL_AMOUNT_Q
      struct first_currency  // Of type: SERIES ; Named struct no: 50000
      struct second_currency  // Of type: SERIES ; Named struct no: 50000
      struct termination_payer  // Of type: PAYMENT
      char[80] termination_info_s  // Termination Info
      UINT8_T full_termination_c  // Full Termination
      char[3] filler_3_s  // Filler
   }
   UINT32_T termination_number_u  // Termination Number
   UINT8_T termination_operation_c  // Termination Operation
   char[3] filler_3_s  // Filler
}
```

3.6.25.4 Usage and Conditions

A trade report can be totally or partially terminated. For a totally terminated trade report, the trade report object and all flows that are not yet paid are set into state Terminated. For a partially terminated trade report, the flows that are not yet paid are modified and the termination state for the trade report and affected flows is set to Partially terminated. A termination payment can be specified in both cases. A partially terminated trade report can be partially terminated again and/or fully terminated.

When the termination transaction is handled (on the termination agree date), the trade report (for both the party and counterparty) will get substate Pending Termination and will remain in that state until the termination object has been matched by the counterparty. The purpose of the Pending Termination substate is to let the user see that something is happening to the trade report.

Termination records must be matched with the counterparty's record before they become valid. All details of the Termination, except Termination Information, must match.

Note that no broadcast is sent to the counterparty when a termination is registered or matched.

3.6.25.5 Return Codes

Even if a Terminate Swap Trade Report transaction is accepted by the system, it is possible that it will not be executed immediately. The below statuses give more information:

Completion	Cstatus	TxStat (reason code)
The deal is subject to collateral checks. Please refer to KB1 broadcast for result.	Successful	OTC_COLLCHECK

Please refer to the *OMex System's Error Messages* for details about why transactions are aborted.

© The NASDAQ OMX Group, Inc. • 2013

3.6.26 CC99 [Clearing Member Accept or Reject OTC trade TRANSACTION]

3.6.26.1 Fingerprint

TRANSACTION properties	
transaction type	CC99
calling sequence	omniapi_tx_ex
struct name	accept_reject_trade_report_for_clearing
facility	EP0
partitioned	false

3.6.26.2 Related Messages

CB3, CQ80, CQ81, CQ82

3.6.26.3 Purpose

This transaction is used by the Clearing Member to either accept or reject OTC trades which have been automatically given up to him.

> **Note:**
> This transaction is deprecated and will be replaced by KC5.

3.6.26.4 Structure

The CC99 TRANSACTION has the following structure:

```
struct accept_reject_trade_report_for_clearing {
   struct transaction_type
   struct series  // Named struct no: 50000
   UINT64_T trade_report_nbr_q  // Trade report number
   char[32] name_s  // Name
   UINT8_T confirm_reject_c  // Confirm or Reject
   char[3] filler_3_s  // Filler
}
```

3.6.26.5 Usage and conditions

When a trade is automatically given up for clearing, it is possible for the Clearing Member to require a possibility to either accept or reject the trade before it's taken up. A trade propagating into a clearing account where confirmation is required will remain in an unmatched state, with a sub state "Waiting for Clearing Member Accept" until it has been accepted. If the trade is accepted, it will continue its processing where if was put in a waiting state. If the trade is rejected by the Clearing Member, it will be set in a reject state.

© The NASDAQ OMX Group, Inc. • 2013

> **Note:**
> This transaction may be rejected, in case one is trying to act on a trade for which one is not entitled to perform this action.

3.6.26.6 Return Codes

Even if a Clearing Member confirmes an OTC Trade Report and the confirmation is accepted by the system, it is possible that it will not be executed immediately. The below statuses give more information:

Completion	Cstatus	TxStat (reason code)
The deal is subject to collateral checks. Please refer to KB1 broadcast for result.	Successful	OTC_COLLCHECK

Please refer to the *OMex System's Error Messages* for details about why transactions are aborted.

3.6.27 CD4 [Transitory Account Trades TRANSACTION]

3.6.27.1 Fingerprint

TRANSACTION properties	
transaction type	CD4
calling sequence	omniapi_tx_ex
struct name	cl_reregistration
facility	EP3
partitioned	false

3.6.27.2 Related Messages

CD5

3.6.27.3 Purpose

Two types of transactions are used to transfer trades from the daily account to the client account.

The CD4 transaction type is used by the Trader and identifies the trade that is to be moved by the Order Number, the Deal Price, and the Series (not a Combination Series).

The other transaction type, CD5, is used by the Back Office (application) and identifies a trade by using the unique Trade Number.

3.6.27.4 Structure

The CD4 TRANSACTION has the following structure:

```
struct cl_reregistration {
   struct transaction type
```


© The NASDAQ OMX Group, Inc. • 2013

```
    struct series  // Named struct no: 50000
    char[12] reserved_12_s  // Reserved
    UINT8_T items_c  // Item
    char[3] filler_3_s  // Filler
    Array ITEM [max no: 100] {
       QUAD_WORD order_number_u  // Order Number
       INT32_T deal_price_i  // Price, Deal
       INT64_T deal_quantity_i  // Quantity, Deal
       char[10] ex_client_s  // Client
       UINT8_T open_close_req_c  // Open  Close Request
       char[15] customer_info_s  // Customer, Information
       char[2] filler_2_s  // Filler
    }
}
```

3.6.27.5 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. For this transaction, an asynchronous transaction, the possible choices are close and normal. Close will be treated according to the rules for the destination account. Information about the synchronous transaction i.e. Daily Account Trades Transaction used by Back Office, see **CD5**.

If client information is omitted, the client identity in the original trade will be used.

The transaction can fail for a number of reasons. For this type of transaction only a consistency check is made. It is up to the Trader to check with his Back Office that the transfer was made.

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 (CD32).

A Daily Account Trades transaction can only be canceled on the same business day as it is created.

3.6.27.6 Return Codes

Even if a transfer operation is accepted by the system, it is possible that it will not be executed immediately. The below statuses give more information:

Completion	Cstatus	TxStat (reason code)
The transfer operation is performed.	Successful	CL_OMN_NORMAL
The transfer operation is subject to collateral checks. If rejected, please refer to broadcast CB146. If approved, please refer to broadcast BD39 and BD6.	Successful	CL_OMN_COLLCHECK

Please refer to the *OMex System's Error Messages* for details about why transactions are aborted.

3.6.28 CD5 [Transitory Account Trades TRANSACTION]

3.6.28.1 Fingerprint

TRANSACTION properties	
transaction type	CD5
calling sequence	omniapi_tx_ex
struct name	cl_reregistration_bo
facility	EP3
partitioned	false

3.6.28.2 Related Messages

CD4

3.6.28.3 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.

Another transaction type, CD4, is used by the Trader and identifies the trade that is to be moved by the Order Number, the Deal Price, and the Series (not a Combination Series).

3.6.28.4 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
   }
}
```


© The NASDAQ OMX Group, Inc. • 2013

3.6.28.5 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 a close order cannot be executed for CD5, an error message will be returned. For information about the asynchronous transaction i.e. the Daily Account Trades Transaction used by Trader, see CD4.

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.

3.6.28.6 Return Codes

Even if a transfer operation is accepted by the system, it is possible that it will not be executed immediately. The below statuses give more information:

Completion	Cstatus	TxStat (reason code)
The transfer operation is performed.	Successful	CL_OMN_NORMAL
The transfer operation is subject to collateral checks. If rejected, please refer to broadcast CB146. If approved, please refer to broadcast BD39 and BD6.	Successful	CL_OMN_COLLCHECK

Please refer to the *OMex System's Error Messages* for details about why transactions are aborted.

3.6.29 CD8 [Countersign Trade TRANSACTION]

3.6.29.1 Fingerprint

TRANSACTION properties	
transaction type	CD8
calling sequence	omniapi_tx_ex
struct name	countersign_trade
facility	EP3
partitioned	false

© The NASDAQ OMX Group, Inc. • 2013

3.6.29.2 Related Messages

CQ13

3.6.29.3 Purpose

This transaction is used to countersign trades that are in holding state. Use CQ13 to retrieve information on trades in holding state.

3.6.29.4 Structure

The CD8 TRANSACTION has the following structure:

```
struct countersign_trade {
   struct transaction_type
   struct series  // Named struct no: 50000
   INT32_T trade_number_i  // Trade Number
   INT32_T ext_status_i  // Return Status
   UINT8_T confirm_reject_c  // Confirm or Reject
   char[3] filler_3_s  // Filler
}
```

3.6.29.5 Usage and conditions

Series

the Series of the trade to be countersigned.

Trade number

the Trade number of the trade to be countersigned.

Confirm or Reject

should be set to either Rejected or Confirmed.

3.6.30 CD27 [Rectify Trade (Open/Close) TRANSACTION]

3.6.30.1 Fingerprint

TRANSACTION properties	
transaction type	CD27
calling sequence	omniapi_tx_ex
struct name	rectify_trade
facility	EP3
partitioned	false

© The NASDAQ OMX Group, Inc. • 2013

3.6.30.2 Related Messages

CD28

3.6.30.3 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.

3.6.30.4 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
   }
}
```

3.6.30.5 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

© The NASDAQ OMX Group, Inc. • 2013

these fields must be identical to that of the trade to be rectified.

3.6.31 CD28 [Rectify Trade TRANSACTION]

3.6.31.1 Fingerprint

TRANSACTION properties	
transaction type	CD28
calling sequence	omniapi_tx_ex
struct name	rectify_trade
facility	EP3
partitioned	false

3.6.31.2 Related Messages

CD27

3.6.31.3 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.

3.6.31.4 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
   }
}
```

3.6.31.5 Usage and conditions

Series

© The NASDAQ OMX Group, Inc. • 2013

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.

3.6.31.6 Return Codes

Even if a rectify transaction is accepted by the system, it is possible that it will not be executed immediately. The below statuses give more information:

Completion	Cstatus	TxStat (reason code)
The rectify operation is performed.	Successful	CL_OMN_NORMAL
The rectify operation is subject to manual checks, and will not go through until manually approved. If approved, please refer to broadcast BD39 and BD6.	Successful	CL_OMN_REQHOLDING
The rectify operation is subject to collateral checks. If rejected, please refer to broadcast CB146. If approved, please refer to broadcast BD39 and BD6.	Successful	CL_OMN_COLLCHECK

Please refer to the *OMex System's Error Messages* for details about why transactions are aborted.

3.6.32 CD31 [Rectify Deal TRANSACTION]

3.6.32.1 Fingerprint

TRANSACTION properties	
transaction type	CD31
calling sequence	omniapi_tx_ex
struct name	rectify_deal
facility	EP3
partitioned	false

© The NASDAQ OMX Group, Inc. • 2013

3.6.32.2 Purpose

A deal rectification transaction is used for changing a whole deal or to cancel it.

3.6.32.3 Structure

The CD31 TRANSACTION has the following structure:

```
struct rectify_deal {
   struct transaction_type
   struct series  // Named struct no: 50000
   UINT8_T instance_c  // Instance, Number
   UINT8_T operation_c  // Operation
   UINT16_T items_n  // Items
   struct other_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
   }
   INT32_T deal_price_i  // Price, Deal
   INT32_T deal_number_i  // Deal Number
   Array ITEM [max no: 255] {
      INT32_T trade_number_i  // Trade Number
      INT64_T trade_quantity_i  // Quantity, Trade
      UINT8_T bid_or_ask_c  // Bid or Ask
      CHAR reserved_1_c  // Reserved
      char[2] reserved_2_s  // Reserved
   }
}
```

3.6.32.4 Usage and conditions

All trades in the deal must belong to the customer's own accounts. The externally allowed number of days for rectification for the instrument type is checked before the operation is carried through.

Deal Cancellation

The transaction may be used to cancel a deal. This is useful for canceling an Average Price Trade transaction (CD32) or for canceling a Daily Account Trades transaction (CD4, CD5). These transactions can only be canceled on the same business day as they were originally created.

In order to cancel a deal, one transaction is used.

In the first transaction:

Operation

must be set to delete.

Series, Other

© The NASDAQ OMX Group, Inc. • 2013

Price, Deal Item

fields must be set to zero or in other words, the trades in the deal must not be specified.

Instance, Number

is ignored.

> **Note:** In case the average price trade, resulting from the Average Price Trade transaction to be canceled, has been subject to Daily Account Trades transaction(s), these must first be canceled before the Average Price Trade transaction itself can be canceled.

Deal Rectification

In order to rectify a deal, two transactions must be used. Series and price may be altered for the deal. Quantity and bid/ask may be altered for the trades in the deal. The new values for these characteristics must be specified in both the first and the second transaction even if unchanged from the original deal.

In the first transaction:

Operation

must be set to delete.

Series

must be set to the series for the deal replacing the faulty deal .

Series, Other

is set to the series for the deal replacing the faulty deal.

Instance, Number

is ignored.

In the second transaction:

Operation

must be set to create.

Series

must be set to the series for the deal replacing the faulty deal.

Series, Other

© The NASDAQ OMX Group, Inc. • 2013

must be set to the series in the original deal.

Instance, Number

is ignored.

> **Note:** The functionality to change series is currently limited to series handled within the same clearing partition.

3.6.32.5 Return Codes

Even if a rectify transaction is accepted by the system, it is possible that it will not be executed immediately. The below statuses give more information:

Completion	Cstatus	TxStat (reason code)
The rectify operation is performed.	Successful	CL_OMN_NORMAL
The rectify operation is subject to manual checks, and will not go through until manually approved. If approved, please refer to broadcast BD39 and BD6.	Successful	CL_OMN_REQHOLDING
The rectify operation is subject to collateral checks. If rejected, please refer to broadcast CB146. If approved, please refer to broadcast BD39 and BD6.	Successful	CL_OMN_COLLCHECK

Please refer to the *OMex System's Error Messages* for details about why transactions are aborted.

3.6.33 CD32 [Average Price Trade TRANSACTION]

3.6.33.1 Fingerprint

TRANSACTION properties	
transaction type	CD32
calling sequence	omniapi_tx_ex
struct name	average_price_trade
facility	EP3
partitioned	false

3.6.33.2 Related Messages

CQ16, CQ17, CC12

© The NASDAQ OMX Group, Inc. • 2013

3.6.33.3 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.

3.6.33.4 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
   }
}
```

3.6.33.5 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.

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 may be canceled. This is achieved by canceling the final deal, at the average price, created by the Average Price Trade transaction. The deal is canceled by use of the Rectify Deal transaction (CD31).

A rectify deal transaction must be confirmed before the operation is carried through. To retrieve information on rectify deals put on hold, use CQ16 or CQ17, and to confirm or reject the transaction, use CC12.

An Average Price Trade transaction can only be canceled on the same business day as it is created.

> **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**.

3.6.33.6 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.

3.6.34 CD34 [Transfer Position TRANSACTION]

3.6.34.1 Fingerprint

TRANSACTION properties	
transaction type	CD34
calling sequence	omniapi_tx_ex
struct name	cl_transfer_position
facility	EP5
partitioned	false

3.6.34.2 Purpose

The purpose of this transaction is to let a participant transfer positions from one account to another account.

3.6.34.3 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
}
```


© The NASDAQ OMX Group, Inc. • 2013

3.6.34.4 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.

3.6.34.5 Return Codes

Even if a transfer operation is accepted by the system, it is possible that it will not be executed immediately. The below statuses give more information:

Completion	Cstatus	TxStat (reason code)
The transfer operation is subject to collateral checks. If rejected, please refer to broadcast CB146. If approved, please refer to broadcast BD39 and BD6.	Successful	CL_OMN_COLLCHECK

Please refer to the *OMex System's Error Messages* for details about why transactions are aborted.

3.6.35 CD35 [Give up Request TRANSACTION]

3.6.35.1 Fingerprint

TRANSACTION properties	
transaction type	CD35
calling sequence	omniapi_tx_ex
struct name	give_up_request
facility	EP3
partitioned	false

3.6.35.2 Purpose

This transaction is used to give up a trade to another member.

3.6.35.3 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
}
```

3.6.35.4 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.

3.6.36 CD38 [Long Position Adjustment TRANSACTION]

3.6.36.1 Fingerprint

TRANSACTION properties	
transaction type	CD38
calling sequence	omniapi_tx_ex
struct name	long_position_adj
facility	EP3
partitioned	false

© The NASDAQ OMX Group, Inc. • 2013

3.6.36.2 Purpose

The purpose of this transaction is to net a position by closing an equal amount of long and short contracts respectively.

CD38 will be replaced by CD54.

3.6.36.3 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
   Array ITEM [max no: 1500] {
      struct account
      struct series  // Named struct no: 50000
      INT32 T long adjustment i  // Long Adjustment
   }
}
```

3.6.36.4 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.

3.6.37 CD54 [Position Closeout QUERY]

3.6.37.1 Fingerprint

QUERY properties	
transaction type	CD54
calling sequence	omniapi_query_ex
struct name	position_closeout
facility	EP3
partitioned	true

© The NASDAQ OMX Group, Inc. • 2013

QUERY properties	
answers	CA54

ANSWER properties	
transaction type	CA54
struct name	position_closeout_status
segmented	false

3.6.37.2 Related Messages

CQ122, CQ123, CD55

3.6.37.3 Purpose

The purpose of this transaction is to allow closeout of a collection of positions.

3.6.37.4 Structure

The CD54 QUERY has the following structure:

```
struct position_closeout {
   struct transaction_type
   struct series  // Named struct no: 50000
   UINT16_T items_n  // Items
   char[2] filler_2_s  // Filler
   Array ITEM [max no: 950] {
      struct account
      struct series  // Named struct no: 50000
      INT64_T final_held_q  // Held/Long position, After closeout
      INT64_T closeout_qty_i  // Quantity, Close out
      char[8] date_s  // Date
   }
}
```

3.6.37.5 Usage and conditions

CD54 is implemented as a query in order to be able to return an answer. The answer indicates for each individual position closeout request whether it was successfully processed or not.

Series

identifies together with account the position.

Account

identifies together with Series the position.

Closeout Quantity

- The quantity by which the position should be closed out.

© The NASDAQ OMX Group, Inc. • 2013

- If Closeout quantity is set to zero, the position will be closed out down to the requested Final held position. This is only allowed for closeout of current business date positions.

Final Held

- The requested held/ long position after position close-out.
- Final held must be zero if Closeout quantity is non-zero.

Date

is the Clearing date for which the position should be closed out.

3.6.37.6 Answer Structure

The CA54 ANSWER has the following structure:

```
struct position_closeout_status {
   struct transaction_type
   UINT16_T items_n  // Items
   char[2] filler_2_s  // Filler
   Array ITEM [max no: 950] {
      struct account
      struct series  // Named struct no: 50000
      INT64_T final_held_q  // Held/Long position, After closeout
      INT64_T closeout_qty_i  // Quantity, Close out
      INT32_T closeout_status_i  // Status, Close out
      char[8] date_s  // Date
   }
}
```

3.6.38 CO7 [Enter FRA Trade Report TRANSACTION]

3.6.38.1 Fingerprint

TRANSACTION properties	
transaction type	CO7
calling sequence	omniapi_tx_ex
struct name	enter_fra_trade_report
facility	EP0
partitioned	false

3.6.38.2 Purpose

The purpose of this transaction is to enter an FRA trade report.

3.6.38.3 Structure

The CO7 TRANSACTION has the following structure:

```
struct enter_fra_trade_report {
   struct transaction_type
   struct series  // Named struct no: 50000
   struct fra  // Named struct no: 85
}
```

3.6.38.4 Usage and Conditions

Passthrough info

is called Participant reference and contains an identification used by an external system, for example a BIC code.

Date, settlement

is used for Effective date (when payment is delivered, and is also the start date of the FRA period).

Date, as of

is Trade date.

Name

holds an end user identity, typically the NT-user.

Sell SI

contains an settlement instruction and is considered applicable **only** if **Payment settled by CSD Y/NandUse SSI** are both no**and** the trade report is on the sell side. It is otherwise neglected but considered an error if the trade report is on the buy side.

Bought or Sold

The mapping is as follows:

Bought = FIXED/FLOAT meaning 'buying Floating for Fixed' (also called 'lend').

Sold = FLOAT/FIXED meaning 'selling Floating for Fixed' (also called 'borrow').

Novation

Currently not applicable to FRA trade reports.

Floating rate index

is the index used for the floating rate in the contract. May be empty if a non-standard rate is applied.

Notional amount

is the amount to which all considerations and interest rates relate. The amount is purely imaginative as far as the FRA contract is concerned.

Date, termination

© The NASDAQ OMX Group, Inc. • 2013

is the date ending the FRA contract period.

3.6.39 CO9 [Enter IR Swap Trade Report TRANSACTION]

3.6.39.1 Fingerprint

TRANSACTION properties	
transaction type	CO9
calling sequence	omniapi_tx_ex
struct name	enter_ir_swap_trade_report
facility	EP0
partitioned	false

3.6.39.2 Purpose

The purpose of this transaction is to enter an Interest Rate Swap Trade Report.

3.6.39.3 Structure

The CO9 TRANSACTION has the following structure:

```
struct enter_ir_swap_trade_report {
   struct transaction_type
   struct series  // Named struct no: 50000
   struct ir_swap
   UINT16_T items_n  // Items
   char[2] filler_2_s  // Filler
   Array ITEM [max no: 500] {
      struct swap_flow
   }
}
```

3.6.39.4 Usage and Conditions

The trade report and swap flow objects are matched against existing objects in state Unmatched and their states are set to Unmatched, Matched or Novated, as appropriate.

When a trade report is created it can not enter state Matched until all swap flow objects have been matched, the trade report and flow objects have been authorized (if required), and the trade report condition has been confirmed by the counterparty.

When the trade report and all its swap flows are in state Matched the trade report is said to be “fully” matched.

© The NASDAQ OMX Group, Inc. • 2013

3.6.40 CQ3 [Position QUERY]

3.6.40.1 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

3.6.40.2 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.

3.6.40.3 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
}
```

3.6.40.4 Usage and conditions

Series

© The NASDAQ OMX Group, Inc. • 2013

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.

3.6.40.5 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
   }
}
```

3.6.40.6 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).

3.6.41 CQ8 [Fixing Values QUERY]

3.6.41.1 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

3.6.41.2 Purpose

This transaction retrieves fixing value for cash settled products (on a daily basis, when they are exercised or when they are closed).

3.6.41.3 Structure

The CQ8 QUERY has the following structure:

```
struct query_fixing_val {
```


© The NASDAQ OMX Group, Inc. • 2013

```
    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
}
```

3.6.41.4 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.

3.6.41.5 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
    }
}
```

3.6.41.6 Answer, comments

If the maximum number of items for one transaction is returned, the query should be repeated with incremented segment number.

© The NASDAQ OMX Group, Inc. • 2013

3.6.42 CQ10 [Query missing trade QUERY]

3.6.42.1 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

3.6.42.2 Related Messages

BD6 (Dedicated Trade Information VIB)

CQ11 (Query Missing Trade, Historical Query).

3.6.42.3 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.

3.6.42.4 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
}
```


© The NASDAQ OMX Group, Inc. • 2013

3.6.42.5 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.

3.6.42.6 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
         struct cl_trade_trade_report_api  // Named struct no: 67
         struct cl_trade_fixed_income_api  // Named struct no: 68
         struct cl_trade_cancel_trade_api  // Named struct no: 70
      }
   }
}
```

© The NASDAQ OMX Group, Inc. • 2013

3.6.42.7 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. The variable record layout is:

Broadcast Header

- Broadcast type
- Items (no of subitems), items_n
- Size (total size in bytes of broadcast including the header), size_n

Sub-item Header

- Named struct no (number of structure following), named_struct_n
- Size (total size in bytes of sub-item including the sub-item header), size_n

Data Structure

- Data structure, Any Named Structure

In practice, when retrieving trades disseminated with VIB's, the actual data structure is a sequence of:

- cl_trade_base_api (named struct no = 3), followed by
- cl_trade_secur_part (named struct no = 20).

It may be useful to remember the relation:

cl_trade_api_t + exchange_info_s = cl_trade_base_api_t + cl_trade_secur_part_t

3.6.43 CQ11 [Query missing trade, historical QUERY]

3.6.43.1 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

© The NASDAQ OMX Group, Inc. • 2013

VIA 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.
segmented	false

3.6.43.2 Related Messages

BD6 (Dedicated Trade Information VIB) and CQ10 (Query Missing Trade Query).

3.6.43.3 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.

3.6.43.4 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
}
```

3.6.43.5 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.

3.6.43.6 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
         struct cl_trade_trade_report_api  // Named struct no: 67
         struct cl_trade_fixed_income_api  // Named struct no: 68
         struct cl_trade_cancel_trade_api  // Named struct no: 70
      }
   }
}
```

3.6.43.7 Answer, comments

See CQ10.

3.6.44 CQ13 [Holding Trade QUERY]

3.6.44.1 Fingerprint

QUERY properties	
transaction type	CQ13
calling sequence	omniapi_query_ex
struct name	query_holding_trade
facility	EP3
partitioned	true
answers	CA13

ANSWER properties	
transaction type	CA13
struct name	answer_trade

© The NASDAQ OMX Group, Inc. • 2013

ANSWER properties	
segmented	false

3.6.44.2 Related Messages

CD8

3.6.44.3 Purpose

This query is used to retrieve trades that are in holding state.

3.6.44.4 Structure

The CQ13 QUERY has the following structure:

```
struct query_holding_trade {
   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
}
```

3.6.44.5 Usage and conditions

A trade is put in holding state if the account on which the trade is done, is set up to require Countersign on trades. The query is used by the countersign responsible. The trades leave the holding state after a confirm or a reject. A reject will put the trade on the Customer reject account. Trades that are still in holding state at the end of the day are rejected. To countersign the trade, use CD8.

Series

must be completed with **Country Number**, **Market Code** and **Instrument Group**.

Search Series
Account

identifies the trades to be returned in the answer.

Segment Number

is one for the first query and then incremented.

3.6.44.6 Answer Structure

The CA13 ANSWER has the following structure:

```
struct answer_trade {
   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: 155] {
       struct cl_trade_api  // Named struct no: 1
    }
}
```

3.6.44.7 Answer, comments

If the maximum number of items for one transaction is returned, the query should be repeated with incremented segment number.

Apart from the header each trade in the response contains the same information as directed_trade_t.

3.6.45 CQ14 [Holding Rectify Trade QUERY]

3.6.45.1 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

3.6.45.2 Related Messages

CQ15, CC11

3.6.45.3 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.

© The NASDAQ OMX Group, Inc. • 2013

3.6.45.4 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
}
```

3.6.45.5 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.

3.6.45.6 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
```

© The NASDAQ OMX Group, Inc. • 2013

```
            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
        }
    }
}
```

3.6.45.7 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

© The NASDAQ OMX Group, Inc. • 2013

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.

3.6.46 CQ15 [Detailed Holding Rectify Trade QUERY]

3.6.46.1 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

ANSWER properties	
struct name	answer_rectify_ext_cont
segmented	false

3.6.46.2 Related Messages

CQ14, CC11

3.6.46.3 Purpose

This query is used for receiving detailed information about a holding rectify trade.

3.6.46.4 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
}
```

3.6.46.5 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**.

3.6.46.6 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
   }
}
```


© The NASDAQ OMX Group, Inc. • 2013

3.6.47 CQ16 [Holding Rectify Deal QUERY]

3.6.47.1 Fingerprint

QUERY properties	
transaction type	CQ16
calling sequence	omniapi_query_ex
struct name	query_rectify_d
facility	EP3
partitioned	true
answers	CA16

ANSWER properties	
transaction type	CA16
struct name	answer_rectify_d
segmented	false

3.6.47.2 Related Messages

CQ17, CC12

3.6.47.3 Purpose

The purpose of this query is to list rectified deals that are in holding state or that have been in holding state and now are completed etc.

3.6.47.4 Structure

The CQ16 QUERY has the following structure:

```
struct query_rectify_d {
   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
}
```

3.6.47.5 Usage and conditions

Only deals where all trades included are registred on the same customer can be rectified by that customer. The customer can use this transaction to obtain information on possible rectify deals on hold and then use this information to either confirm or reject the rectify.

© The NASDAQ OMX Group, Inc. • 2013

Use CQ17 to get detailed information regarding a holding rectify deal. Use CC12 to confirm or reject the request to rectify the deal.

Series

must be completed with **Country Number**, **Market Code** and **Instrument Group**.

Search Series

identifies the positions to be returned in the answer.

Segment Number

is one for the first query and then incremented.

3.6.47.6 Answer Structure

The CA16 ANSWER has the following structure:

```
struct answer_rectify_d {
   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
   char[2] filler_2_s  // Filler
   Array ITEM [max no: 100] {
      struct orig_deal_part {
         struct series  // Named struct no: 50000
         char[8] asof_date_s  // Date, As Of
         char[6] asof_time_s  // Time, As Of
         char[2] filler_2_s  // Filler
         INT32_T deal_price_i  // Price, Deal
         INT32_T deal_number_i  // Deal Number
         INT64_T deal_quantity_i  // Quantity, Deal
      }
      struct rectify_deal_part {
         struct new_series
         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 rectify_deal_number_q  // Rectify Deal Number
         struct trading_code
         struct ex_user_code
         INT32_T state_i  // State, Product
         INT32_T new_deal_price_i  // Price, New Deal
      }
   }
}
```


© The NASDAQ OMX Group, Inc. • 2013

3.6.47.7 Answer, comments

If the maximum number of items for one transaction is returned, the query should be repeated with incremented segment number.

3.6.48 CQ17 [Detailed Rectify Deal QUERY]

3.6.48.1 Fingerprint

QUERY properties	
transaction type	CQ17
calling sequence	omniapi_query_ex
struct name	query_rectify_d_cont
facility	EP3
partitioned	false
answers	CA17

ANSWER properties	
transaction type	CA17
struct name	answer_rectify_d_cont
segmented	false

3.6.48.2 Related Messages

CQ16, CC12

3.6.48.3 Purpose

This transaction gives detailed information of the trades included in a specified rectified deal in state holding.

3.6.48.4 Structure

The CQ17 QUERY has the following structure:

```
struct query_rectify_d_cont {
   struct transaction type
   struct series  // Named struct no: 50000
   INT64_T rectify_deal_number_q  // Rectify Deal Number
}
```

3.6.48.5 Usage and conditions

Only deals where all trades included are registred on the same customer can be rectified by that customer. The customer can use this transaction to obtain information on possible rectify deals on hold and then use

© The NASDAQ OMX Group, Inc. • 2013

this information to either confirm or reject the rectify. Use CQ16 to obtain rectify deal number and original series to be supplied as query parameters when using CQ17.

Use CQ16 to get information regarding a holding rectify deal. Use CC12 to confirm or reject the request to rectify the deal.

Series

must be completed with **Country Number**,**Market Code** and **Instrument Group**.

3.6.48.6 Answer Structure

The CA17 ANSWER has the following structure:

```
struct answer_rectify_d_cont {
   struct transaction type
   UINT16_T items_n  // Items
   char[2] filler_2_s  // Filler
   Array ITEM [max no: 255] {
      struct series  // Named struct no: 50000
      INT32_T trade_number_i  // Trade Number
      UINT8_T bid_or_ask_c  // Bid or Ask
      char[3] filler_3_s  // Filler
      INT64_T trade_quantity_i  // Quantity, Trade
   }
}
```

3.6.49 CQ19 [Account Propagation QUERY]

3.6.49.1 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

© The NASDAQ OMX Group, Inc. • 2013

3.6.49.2 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.

3.6.49.3 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
}
```

3.6.49.4 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

3.6.49.5 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
   }
}
```

3.6.49.6 Answer, comments

If the maximum number of items for one transaction is returned, the query should be repeated with incremented segment number.

© The NASDAQ OMX Group, Inc. • 2013

3.6.50 CQ20 [Open Interest QUERY]

3.6.50.1 Fingerprint

QUERY properties	
transaction type	CQ20
calling sequence	omniapi_query_ex
struct name	query_open_interest
facility	EP3
partitioned	true
answers	CA20

ANSWER properties	
transaction type	CA20
struct name	answer_open_interest
segmented	false

3.6.50.2 Purpose

The purpose of this query is to retrieve the Open Interest per series. The Open Interest for a series is calculated once a day by summarizing the positions for all accounts.

This query is only available when the signal BI7, Information Type 1 has been sent.

See also CQ72 that returns more.

3.6.50.3 Structure

The CQ20 QUERY has the following structure:

```
struct query_open_interest {
   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
}
```

3.6.50.4 Usage and conditions

Series

© The NASDAQ OMX Group, Inc. • 2013

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

must be filled with current business date.

3.6.50.5 Answer Structure

The CA20 ANSWER has the following structure:

```
struct answer_open_interest {
   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 final open interest q  // Final Open Interest
   }
}
```

3.6.50.6 Answer, comments

If the maximum number of items for one transaction is returned, the query should be repeated with incremented segment number.

3.6.51 CQ21 [Pending Exercise Request QUERY]

3.6.51.1 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

© The NASDAQ OMX Group, Inc. • 2013

ANSWER properties	
struct name	answer_exercise_req
segmented	false

3.6.51.2 Related Messages

CC15

3.6.51.3 Purpose

The purpose of this query is to retrieve all pending exercise requests. Use CC15 to either confirm or reject the pending exercise request.

3.6.51.4 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
}
```

3.6.51.5 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.

3.6.51.6 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
```


© The NASDAQ OMX Group, Inc. • 2013

```
   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
   }
}
```

3.6.51.7 Answer, comments

If the maximum number of items for one transaction is returned, the query should be repeated with incremented segment number.

3.6.52 CQ22 [Error Message QUERY]

3.6.52.1 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

3.6.52.2 Related Messages

BD6

3.6.52.3 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.

3.6.52.4 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
}
```

3.6.52.5 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.

3.6.52.6 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
```


© The NASDAQ OMX Group, Inc. • 2013

```
        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
    }
}
```

3.6.52.7 Answer, comments

If the maximum number of items for one transaction is returned, the query should be repeated with incremented segment number.

3.6.53 CQ31 [Simulate Fee QUERY]

3.6.53.1 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

3.6.53.2 Purpose

This query calculates the fees for a particular trade. The fees are returned as delivery information (see Answer below).

3.6.53.3 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
}
```

3.6.53.4 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

3.6.53.5 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
   }
}
```

3.6.53.6 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

© The NASDAQ OMX Group, Inc. • 2013

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.

3.6.54 CQ36 [Average Price Trade QUERY]

3.6.54.1 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

3.6.54.2 Purpose

This query returns the trade number of the trades that are part of an average price trade.

3.6.54.3 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
}
```

3.6.54.4 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.

3.6.54.5 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
   }
}
```

3.6.54.6 Answer, comments

If the maximum number of items for one transaction is returned, the query should be repeated with incremented segment number.

3.6.55 CQ38 [Account QUERY]

3.6.55.1 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

© The NASDAQ OMX Group, Inc. • 2013

ANSWER properties	
segmented	true

3.6.55.2 Purpose

The purpose of this query is to retrieve account information for own accounts.

3.6.55.3 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
}
```

3.6.55.4 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.

3.6.55.5 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
    }
}
```

3.6.55.6 Answer, comments

If the maximum number of items for one transaction is returned, the query should be repeated with incremented segment number.

3.6.56 CQ39 [Trade Change QUERY QUERY]

3.6.56.1 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

3.6.56.2 Related Messages

CQ10, BD39

3.6.56.3 Purpose

The purpose of this query is to retrieve missing trade change broadcasts.

3.6.56.4 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
```


© The NASDAQ OMX Group, Inc. • 2013

```
   INT32_T sequence_last_i  // Number, Last Sequential
   char[8] date_s  // Date
}
```

3.6.56.5 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).

3.6.56.6 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 {
         struct series  // Named struct no: 50000
         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
      }
   }
}
```

3.6.57 CQ51 [DC Holding Trade QUERY]

3.6.57.1 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

3.6.57.2 Related Messages

BD41, MO75, MO76, MO459

3.6.57.3 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.

3.6.57.4 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
}
```

3.6.57.5 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

© The NASDAQ OMX Group, Inc. • 2013

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.

3.6.57.6 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.

3.6.57.7 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
         }
      }
   }
```

3.6.57.8 Answer, comments

If the maximum number of items for one transaction is returned, the query should be repeated with incremented segment number.

© The NASDAQ OMX Group, Inc. • 2013

3.6.58 CQ52 [Delivery QUERY]

3.6.58.1 Fingerprint

QUERY properties	
transaction type	CQ52
calling sequence	omniapi_query_ex
struct name	query_missing_delivery
facility	EP3
partitioned	true
answers	CA52

ANSWER properties	
transaction type	CA52
struct name	answer_missing_delivery
segmented	false

3.6.58.2 Related Messages

BD18, CQ53

3.6.58.3 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.

3.6.58.4 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
}
```

3.6.58.5 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 business date.

3.6.58.6 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
   }
}
```

3.6.58.7 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.

Class Number

is a number indicating type of settlement for a delivery item. If this number is above 200, this indicates that the delivery item is informational only, i.e. will not be included in any further settlement processing. The type of settlement is found by taking the class number and subtracting 200, so that if class-number is e.g. 202, this is an informational (200) clearing fee (2).

If this number is between 100 and 200, this indicates that the delivery item will be accumulated for settlement at a later date, i.e. not necessarily the settlement date specified in the delivery. The type of settlement is

© The NASDAQ OMX Group, Inc. • 2013

found by taking the class number and subtracting 100, so that if class-number is e.g. 102, this is a clearing fee (2) which will accrue (100).

3.6.59 CQ53 [Delivery History QUERY]

3.6.59.1 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

3.6.59.2 Related Messages

BD18, CQ52

3.6.59.3 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.

3.6.59.4 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
}
```

© The NASDAQ OMX Group, Inc. • 2013

3.6.59.5 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 historical dates compared to current business 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.

3.6.59.6 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
   }
}
```

3.6.59.7 Answer, comments

Date

contains the date on which this delivery was created.

Class Number

is a number indicating type of settlement for a delivery item. If this number is above 200, this indicates that the delivery item is informational only, i.e. will not be included in any further settlement processing. The type of settlement is found by taking the class number and subtracting 200, so that if class-number is e.g. 202, this is an informational (200) clearing fee (2).

If this number is between 100 and 200, this indicates that the delivery item will be accumulated for settlement at a later date, i.e. not necessarily the settlement date specified in the delivery. The type of settlement is

© The NASDAQ OMX Group, Inc. • 2013

found by taking the class number and subtracting 100, so that if class-number is e.g. 102, this is a clearing fee (2) which will accrue (100).

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.

3.6.60 CQ61 [Holding Give Up Request QUERY]

3.6.60.1 Fingerprint

QUERY properties	
transaction type	CQ61
calling sequence	omniapi_query_ex
struct name	query_give_up_request
facility	EP3
partitioned	true
answers	CA61

ANSWER properties	
transaction type	CA61
struct name	answer_give_up_request
segmented	true

© The NASDAQ OMX Group, Inc. • 2013

3.6.60.2 Related Messages

CC38

CC40

BD29

CQ76

CQ77

3.6.60.3 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.

3.6.60.4 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
}
```

3.6.60.5 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

© The NASDAQ OMX Group, Inc. • 2013

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

© The NASDAQ OMX Group, Inc. • 2013

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.

3.6.60.6 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
   }
}
```

3.6.60.7 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

© The NASDAQ OMX Group, Inc. • 2013

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.

3.6.61 CQ62 [Confirm Give Up Request QUERY]

3.6.61.1 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

3.6.61.2 Related Messages

CC38, CQ61

3.6.61.3 Purpose

This query returns the give-up items sent when a giveup was confirmed. This query can only be sent for a confirmed giveup.

3.6.61.4 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
}
```

3.6.61.5 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.

3.6.61.6 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
```


© The NASDAQ OMX Group, Inc. • 2013

```
        char[15] customer_info_s  // Customer, Information
    }
}
```

3.6.61.7 Answer, comments

This information is the same information as sent in the Confirm Give-Up Trade Transaction, see **CC38**.

3.6.62 CQ65 [Level Position QUERY]

3.6.62.1 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

3.6.62.2 Related Messages

CQ3

3.6.62.3 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.

3.6.62.4 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
}
```

3.6.62.5 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.

© The NASDAQ OMX Group, Inc. • 2013

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.

3.6.62.6 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
   }
}
```

3.6.62.7 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.

3.6.63 CQ68 [Clearing Date QUERY]

3.6.63.1 Fingerprint

QUERY properties	
transaction type	CQ68
calling sequence	omniapi_query_ex

© The NASDAQ OMX Group, Inc. • 2013

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

3.6.63.2 Purpose

The purpose of this query is to retrieve information on the current and the next clearing date for instrument types.

3.6.63.3 Structure

The CQ68 QUERY has the following structure:

```
struct query_clearing_date {
   struct transaction_type
   struct series  // Named struct no: 50000
   struct search_series
}
```

3.6.63.4 Usage and conditions

Series, Search

may be zeroed to retrieve clearing date information on all instrument types handled by a particular clearing server.

3.6.63.5 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
```


© The NASDAQ OMX Group, Inc. • 2013

```
        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
    }
}
```

3.6.63.6 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.

3.6.64 CQ72 [Net Open Interest QUERY]

3.6.64.1 Fingerprint

QUERY properties	
transaction type	CQ72
calling sequence	omniapi_query_ex
struct name	query_open_interest_ext
facility	EP3
partitioned	true
answers	CA72

ANSWER properties	
transaction type	CA72
struct name	answer_open_interest_ext
segmented	false

3.6.64.2 Related Messages

CQ20 – Open Interest

3.6.64.3 Purpose

The purpose of this query is to retrieve the net and gross market open interest per series. This query is only available when the signal BI7, Information Type 1 has been sent.

3.6.64.4 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
}
```

3.6.64.5 Usage and conditions

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 complete up to **Country Number** and **Market Code**.

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.

3.6.64.6 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
   }
}
```


© The NASDAQ OMX Group, Inc. • 2013

3.6.64.7 Answer, comments

If the maximum number of items for one transaction is returned, the query should be repeated with incremented segment number.

3.6.65 CQ76 [Give Up QUERY]

3.6.65.1 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

3.6.65.2 Related Messages

BD29

3.6.65.3 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.

3.6.65.4 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
}
```

3.6.65.5 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.

3.6.65.6 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
   }
}
```

3.6.65.7 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.

3.6.66 CQ77 [Give Up History QUERY]

3.6.66.1 Fingerprint

QUERY properties	
transaction type	CQ77
calling sequence	omniapi_query_ex
struct name	query_api_give_up
facility	EP5

© The NASDAQ OMX Group, Inc. • 2013

QUERY properties	
partitioned	true
answers	CA77

ANSWER properties	
transaction type	CA77
struct name	answer_api_give_up
segmented	false

3.6.66.2 Related Messages

CQ76

3.6.66.3 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.

3.6.66.4 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
}
```

3.6.66.5 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

© The NASDAQ OMX Group, Inc. • 2013

is the last item to get for **Clearing Date, To**. Zero means the last item for that date.

3.6.66.6 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
   }
}
```

3.6.66.7 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.

© The NASDAQ OMX Group, Inc. • 2013

3.6.67 CQ78 [Consideration QUERY]

3.6.67.1 Fingerprint

QUERY properties	
transaction type	CQ78
calling sequence	omniapi_query_ex
struct name	query_consideration
facility	EP0
partitioned	false
answers	CA78

ANSWER properties	
transaction type	CA78
struct name	answer_consideration
segmented	false

3.6.67.2 Purpose

Use this query to retrieve considerations.

3.6.67.3 Structure

The CQ78 QUERY has the following structure:

```
struct query_consideration {
   struct transaction_type
   struct series  // Named struct no: 50000
   INT64_T face_value_q  // Face Value
   INT32_T yield_i  // YIELD_I
   char[8] settlement_date_s  // Date, Settlement
}
```

3.6.67.4 Answer Structure

The CA78 ANSWER has the following structure:

```
struct answer_consideration {
   struct transaction_type
   struct series  // Named struct no: 50000
   INT64_T face_value_q  // Face Value
   INT64_T consideration_q  // Consideration
   UINT64_T clean_price_q  // Price, Clean
   UINT64_T dirty_price_q  // DIRTY_PRICE_Q
   INT32_T yield_i  // YIELD_I
```

© The NASDAQ OMX Group, Inc. • 2013

```
    char[8] settlement_date_s  // Date, Settlement
    UINT16_T dec_in_clean_price_n  // DEC_IN_CLEAN_PRICE_N
    UINT16_T dec_in_dirty_price_n  // DEC_IN_DIRTY_PRICE_N
}
```

3.6.68 CQ80 [OTC Trade Report QUERY]

3.6.68.1 Fingerprint

QUERY properties	
transaction type	CQ80
calling sequence	omniapi_query_ex
struct name	query_trade_report
facility	EP0
partitioned	true
answers	CA80

VIA properties	
transaction type	CA80
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

3.6.68.2 Purpose

This query returns a list of trade reports.

Note:

This transaction is deprecated and will be replaced by KQ1.

3.6.68.3 Structure

The CQ80 QUERY has the following structure:

```
struct query_trade_report {
    struct transaction_type
    struct series  // Named struct no: 50000
    struct party
    struct account
    UINT16_T segment_number_n  // Segment Number
    char[2] filler_2_s  // Filler
    char[8] from_settlement_date_s  // From Settlement Date
    char[8] to_settlement_date_s  // To Settlement Date
```


© The NASDAQ OMX Group, Inc. • 2013

```
    char[32] passthrough_s  // Passthrough Information
    char[32] series_id_s  // Series, Identity
    UINT8_T trade_report_state_c  // Trade Report State
    UINT8_T trade_report_category_c  // Trade Report Category
    UINT8_T bought_or_sold_c  // Bought or Sold
    UINT8_T novation_c  // Novation
    UINT32_T trade_report_type_i  // Trade Report Type
    UINT8_T open_contract_c  // Open Contract
    char[3] filler_3_s  // Filler
}
```

3.6.68.4 Answer Structure

The CA80 VIA has the following structure:

```
struct answer_trade_report {
   struct transaction_type
   struct partition_low
   struct partition_high
   UINT16_T segment_number_n  // Segment Number
   UINT16_T items_n  // Items
}
Sequence {
   struct item_hdr
   Sequence {
      struct sub_item_hdr
      Choice {
         struct fi_trade_report  // Named struct no: 13
         struct fx_trade_report  // Named struct no: 7
         struct cash_trade_report  // Named struct no: 8
         struct agreement_trade_report  // Named struct no: 9
         struct ssi_trade_report  // Named struct no: 10
         struct equity_trade_report  // Named struct no: 12
         struct fra_trade_report  // Named struct no: 11
         struct fi_repo_trade_report  // Named struct no: 14
         struct ir_swap_trade_report  // Named struct no: 15
         struct xcur_swap_trade_report  // Named struct no: 16
         struct cash_transfer_trade_report  // Named struct no: 23
         struct otc_clearing_info  // Named struct no: 83
      }
   }
}
```

3.6.68.5 Answer, comments

The content is different depending on what type of trade report it is.

The answer is any sequence of trade reports according to the various numbered structures referenced above.

© The NASDAQ OMX Group, Inc. • 2013

3.6.69 CQ81 [OTC Trade Report QUERY]

3.6.69.1 Fingerprint

QUERY properties	
transaction type	CQ81
calling sequence	omniapi_query_ex
struct name	query_missing_trade_report
facility	EP0
partitioned	false
answers	CA81

VIA properties	
transaction type	CA81
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

3.6.69.2 Purpose

This query returns missing trade reports.

Note:

This transaction is deprecated and will be replaced by KQ2.

3.6.69.3 Structure

The CQ81 QUERY has the following structure:

```
struct query_missing_trade_report {
   struct transaction_type
   struct series  // Named struct no: 50000
   UINT32_T sequence_first_u  // Sequence First
   UINT32_T sequence_last_u  // Sequence Last
   char[8] timestamp_date_s  // Timestamp, Date
}
```

3.6.69.4 Answer Structure

The CA81 VIA has the following structure:

```
    struct answer_missing_trade_report {
```


© The NASDAQ OMX Group, Inc. • 2013

```
      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 fi_trade_report  // Named struct no: 13
            struct fx_trade_report  // Named struct no: 7
            struct cash_trade_report  // Named struct no: 8
            struct agreement_trade_report  // Named struct no: 9
            struct ssi_trade_report  // Named struct no: 10
            struct equity_trade_report  // Named struct no: 12
            struct fra_trade_report  // Named struct no: 11
            struct fi_repo_trade_report  // Named struct no: 14
            struct ir_swap_trade_report  // Named struct no: 15
            struct xcur_swap_trade_report  // Named struct no: 16
            struct cash_transfer_trade_report  // Named struct no: 23
            struct otc_clearing_info  // Named struct no: 83
         }
      }
   }
```

3.6.70 CQ82 [OTC Trade Report Version QUERY]

3.6.70.1 Fingerprint

QUERY properties	
transaction type	CQ82
calling sequence	omniapi_query_ex
struct name	query_trade_report_version
facility	EP0
partitioned	false
answers	CA82

VIA properties	
transaction type	CA82
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

3.6.70.2 Purpose

This query returns all versions of a Trade Report.

© The NASDAQ OMX Group, Inc. • 2013

Note:
This transaction is deprecated and will be replaced by KQ3.

3.6.70.3 Structure

The CQ82 QUERY has the following structure:

```
struct query_trade_report_version {
   struct transaction_type
   struct series  // Named struct no: 50000
   UINT64_T trade_report_nbr_q  // Trade report number
   UINT16_T segment_number_n  // Segment Number
   UINT16_T items_n  // Items
}
```

3.6.70.4 Answer Structure

The CA82 VIA has the following structure:

```
   struct answer_trade_report {
      struct transaction_type
      struct partition_low
      struct partition_high
      UINT16_T segment_number_n  // Segment Number
      UINT16_T items_n  // Items
   }
   Sequence {
      struct item_hdr
      Sequence {
         struct sub_item_hdr
         Choice {
            struct fi_trade_report  // Named struct no: 13
            struct fx_trade_report  // Named struct no: 7
            struct cash_trade_report  // Named struct no: 8
            struct agreement_trade_report  // Named struct no: 9
            struct ssi_trade_report  // Named struct no: 10
            struct equity_trade_report  // Named struct no: 12
            struct fra_trade_report  // Named struct no: 11
            struct fi_repo_trade_report  // Named struct no: 14
            struct ir_swap_trade_report  // Named struct no: 15
            struct xcur_swap_trade_report  // Named struct no: 16
            struct cash_transfer_group_otc  // Named struct no: 22
            struct cash_transfer_trade_report  // Named struct no: 23
            struct otc_clearing_info  // Named struct no: 83
         }
      }
   }
```


© The NASDAQ OMX Group, Inc. • 2013

3.6.71 CQ86 [OTC Netting Request QUERY]

3.6.71.1 Fingerprint

QUERY properties	
transaction type	CQ86
calling sequence	omniapi_query_ex
struct name	query_otc_netting_req
facility	EP0
partitioned	true
answers	CA86

ANSWER properties	
transaction type	CA86
struct name	answer_otc_netting_req
segmented	true

3.6.71.2 Purpose

This query returns the status of OTC netting requests.

3.6.71.3 Structure

The CQ86 QUERY has the following structure:

```
struct query_otc_netting_req {
   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 query_type_c  // Query type
   CHAR filler_1_s  // Filler
}
```

3.6.71.4 Answer Structure

The CA86 ANSWER has the following structure:

```
struct answer_otc_netting_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: 200] {
```

© The NASDAQ OMX Group, Inc. • 2013

```
        struct otc_netting_req {
           struct series  // Named struct no: 50000
           UINT32_T netting_req_nbr_u  // Netting request number
           char[8] settlement_date_s  // Date, Settlement
           struct trading_code
           INT32_T state_i  // State, Product
           char[100] status_description_s  // Status Description
           char[8] created_date_s  // Date, Created
           char[6] created_time_s  // Time, Created
           UINT8_T instrument_level_c  // INSTRUMENT_LEVEL_C
           CHAR filler_1_s  // Filler
        }
     }
  }
```

3.6.72 CQ90 [Generate IR Swap Flow QUERY]

3.6.72.1 Fingerprint

QUERY properties	
transaction type	CQ90
calling sequence	omniapi_query_ex
struct name	query_generate_ir_swap_flow
facility	EP0
partitioned	true
answers	CA90

ANSWER properties	
transaction type	CA90
struct name	answer_generate_swap_flow
segmented	false

3.6.72.2 Related Messages

CQ91

3.6.72.3 Purpose

This query returns data that can be used as input in an Enter IR Swap transaction.

3.6.72.4 Structure

The CQ90 QUERY has the following structure:

```
struct query_generate_ir_swap_flow {
```


© The NASDAQ OMX Group, Inc. • 2013

```
   struct transaction_type
   struct series  // Named struct no: 50000
   char[8] settlement_date_s  // Date, Settlement
   char[8] date_termination_s  // Date, Maturity
   INT64_T notional_amount_q  // Notional amount
   char[5] first_holiday_id_s  // First State Holiday ID
   UINT8_T rate_reset_c  // Rate Reset
   UINT8_T reset_days_c  // Reset Days
   UINT8_T payment_set_c  // Payment Set
   char[5] second_holiday_id_s  // Second State Holiday ID
   UINT8_T business_day_conv_c  // BUSINESS_DAY_CONV_C
   char[2] filler_2_s  // Filler
   struct member_pay  // Of type: IR_SWAP_LEG
   struct counterparty_pay  // Of type: IR_SWAP_LEG
}
```

3.6.72.5 Usage and Conditions

Settlement Day

is Effective Date.

3.6.72.6 Answer Structure

The CA90 ANSWER has the following structure:

```
struct answer_generate_swap_flow {
   struct transaction_type
   struct partition_low
   struct partition_high
   UINT16_T items_n  // Items
   char[2] filler_2_s  // Filler
   Array ITEM [max no: 500] {
      struct swap_flow
   }
}
```

3.6.72.7 Answer, comments

Note that this query only contains the swap_flow data, and not the swap_flow_leg as for query swap flow (CQ91/CQ347). The reason is that this query is used to generate data that can be used as input in an Enter IR Swap transaction, not to show data about a flow.

3.6.73 CQ91 [Swap Flow QUERY]

3.6.73.1 Fingerprint

QUERY properties	
transaction type	CQ91

© The NASDAQ OMX Group, Inc. • 2013

QUERY properties	
calling sequence	omniapi_query_ex
struct name	query_swap_flow
facility	EP0
partitioned	true
answers	CA91

ANSWER properties	
transaction type	CA91
struct name	answer_swap_flow
segmented	false

3.6.73.2 Related Messages

CQ90

3.6.73.3 Purpose

This query returns data about a Swap flow.

3.6.73.4 Structure

The CQ91 QUERY has the following structure:

```
struct query_swap_flow {
   struct transaction_type
   struct series  // Named struct no: 50000
   UINT64_T trade_report_nbr_q  // Trade report number
   UINT16_T trade_report_version_n  // Trade report version
   UINT16_T segment_number_n  // Segment Number
   UINT8_T flow_state_c  // FLOW_STATE_C
   char[3] filler_3_s  // Filler
}
```

3.6.73.5 Answer Structure

The CA91 ANSWER has the following structure:

```
struct answer_swap_flow {
   struct transaction_type
   struct partition_low
   struct partition_high
   UINT16_T items_n  // Items
   UINT16_T segment_number_n  // Segment Number
   Array ITEM [max no: 250] {
      struct swap_flow_leg {
         struct flow  // Of type: SWAP_FLOW
        struct float_rate_series  // Of type: SERIES ; Named struct no: 50000
```


© The NASDAQ OMX Group, Inc. • 2013

```
            UINT8_T authorization_state_c  // Authorization State
            UINT8_T trade_report_state_c  // Trade Report State
            UINT16_T trade_report_version_n  // Trade report version
            UINT32_T delivery_unit_u  // Delivery Unit
            UINT32_T netting_req_nbr_u  // Netting request number
            UINT32_T pay_calc_req_nbr_u  // Pay calc request number
            UINT32_T orig_flow_number_start_u  // Original Flow Number, Start
Date
            UINT32_T orig_flow_number_end_u  // Original Flow Number, End Date
            char[8] asof_date_s  // Date, As Of
            char[6] asof_time_s  // Time, As Of
            char[2] filler_2_s  // Filler
            char[8] timestamp_date_s  // Timestamp, Date
            char[6] timestamp_time_s  // Timestamp, Time
            UINT8_T termination_state_c  // Termination State
            UINT8_T state_c  // State
            UINT8_T flow_operation_c  // FLOW_OPERATION_C
            char[3] filler_3_s  // Filler
        }
    }
}
```

3.6.73.6 Answer, comments

Note that this query only contains the swap_flow_leg, and not the swap_flow data as for Query Generate IR Swap Flow (CQ90). The reason is that this query is used to show data about a flow, not to generate data that can be used as input in an Enter IR Swap transaction.

3.6.74 CQ92 [Swap Termination QUERY]

3.6.74.1 Fingerprint

QUERY properties	
transaction type	CQ92
calling sequence	omniapi_query_ex
struct name	query_swap_termination
facility	EP0
partitioned	true
answers	CA92

ANSWER properties	
transaction type	CA92
struct name	answer_swap_termination
segmented	false

3.6.74.2 Purpose

This query returns all termination records for a Swap trade report.

3.6.74.3 Structure

The CQ92 QUERY has the following structure:

```
struct query_swap_termination {
   struct transaction_type
   struct series  // Named struct no: 50000
   struct party
   struct account
   UINT64_T trade_report_nbr_q  // Trade report number
   char[8] from_termination_agree_date_s  // From Termination Agree Date
   char[8] to_termination_agree_date_s  // To Termination Agree Date
   char[32] series_id_s  // Series, Identity
   UINT32_T termination_number_u  // Termination Number
   UINT8_T trade_report_state_c  // Trade Report State
   UINT8_T state_c  // State
   UINT8_T termination_search_c  // Termination search option
   CHAR filler_1_s  // Filler
   UINT16_T segment_number_n  // Segment Number
   char[2] filler_2_s  // Filler
}
```

3.6.74.4 Answer Structure

The CA92 ANSWER has the following structure:

```
struct answer_swap_termination {
   struct transaction_type
   struct partition_low
   struct partition_high
   UINT16_T items_n  // Items
   UINT16_T segment_number_n  // Segment Number
   Array ITEM [max no: 100] {
      struct swap_termination_leg {
         struct termination {  // Of type: SWAP_TERMINATION
            struct series  // Named struct no: 50000
            struct account
            char[32] name_s  // Name
            UINT64_T trade_report_nbr_q  // Trade report number
            char[8] termination_agree_date_s  // Termination Agree Date
            INT64_T notional_amount_q  // Notional amount
            INT64_T second_notional_amount_q  // Notional amount ; Of type:
NOTIONAL_AMOUNT_Q
            struct first_currency  // Of type: SERIES ; Named struct no: 50000
            struct second_currency  // Of type: SERIES ; Named struct no: 50000
            struct termination_payer  // Of type: PAYMENT
            char[80] termination_info_s  // Termination Info
            UINT8_T full_termination_c  // Full Termination
            char[3] filler_3_s  // Filler
         }
```


© The NASDAQ OMX Group, Inc. • 2013

```
            struct trading_code
            struct user_code
            struct auth_by_whom
            UINT32_T termination_number_u  // Termination Number
            struct party
            UINT8_T authorization_state_c  // Authorization State
            UINT8_T trade_report_state_c  // Trade Report State
            UINT16_T trade_report_version_n  // Trade report version
            UINT32_T delivery_unit_u  // Delivery Unit
            char[8] timestamp_date_s  // Timestamp, Date
            char[6] timestamp_time_s  // Timestamp, Time
            UINT8_T state_c  // State
            CHAR filler_1_s  // Filler
        }
    }
}
```

3.6.74.5 Answer, comments

Included are records for all partial terminations, any full termination, and termination records that are waiting to be matched.

3.6.75 CQ105 [Invalid Settlement Date QUERY]

3.6.75.1 Fingerprint

QUERY properties	
transaction type	CQ105
calling sequence	omniapi_query_ex
struct name	query_invalid_settle_dates
facility	EP0
partitioned	true
answers	CA105

ANSWER properties	
transaction type	CA105
struct name	answer_invalid_settle_dates
segmented	false

3.6.75.2 Purpose

This query returns a list of invalid settlement dates.

3.6.75.3 Structure

The CQ105 QUERY has the following structure:

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

3.6.75.4 Answer Structure

The CA105 ANSWER has the following structure:

```
struct answer_invalid_settle_dates {
   struct transaction_type
   UINT16_T items_n  // Items
   char[2] filler_2_s  // Filler
   Array ITEM [max no: 90] {
      UINT64_T trade_report_nbr_q  // Trade report number
      UINT64_T party_trade_report_nbr_q  // Party trade report number
      struct account
      struct party
      struct series  // Named struct no: 50000
      INT32_T deal_number_i  // Deal Number
      UINT32_T trade_report_type_i  // Trade Report Type
      UINT8_T trade_report_state_c  // Trade Report State
      UINT8_T trade_report_sub_state_c  // Trade Report Substate
      UINT16_T settle_date_items_n  // Items ; Of type: ITEMS_N
      Array ITEM [max no: 30] {
         struct settlement_dates {
            char[8] invalid_settle_date  // Date ; Of type: YYYYMMDD_S
            char[8] new_settle_date  // Date ; Of type: YYYYMMDD_S
            UINT8_T type_of_date_c  // Type of Date
            char[3] filler_3_s  // Filler
         }
      }
   }
}
```

3.6.76 CQ106 [Settlement Accumulation QUERY]

3.6.76.1 Fingerprint

QUERY properties	
transaction type	CQ106
calling sequence	omniapi_query_ex
struct name	query_otc_netting
facility	EP0
partitioned	true
answers	CA106

© The NASDAQ OMX Group, Inc. • 2013

VIA properties	
transaction type	CA106
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

3.6.76.2 Purpose

This query returns settlement accumulation information.

3.6.76.3 Structure

The CQ106 QUERY has the following structure:

```
struct query_otc_netting {
   struct transaction_type
   struct series  // Named struct no: 50000
   struct account
   char[8] settlement_date_s  // Date, Settlement
   UINT32_T delivery_unit_u  // Delivery Unit
   UINT32_T netting_req_nbr_u  // Netting request number
   UINT16_T segment_number_n  // Segment Number
   char[2] filler_2_s  // Filler
}
```

3.6.76.4 Usage and Conditions

Each query request must have an Account, Settlement Date and Series specified, no wildcards are permitted.

Settlement Date

can be for today, historical or future dates.

Delivery Unit Number
Netting Request Number.

can hold wildcards (including 0).

3.6.76.5 Answer Structure

The CA106 VIA has the following structure:

```
struct answer_otc_netting {
   struct transaction_type
   UINT16_T segment_number_n  // Segment Number
   UINT16_T items_n  // Items
}
Sequence {
   struct item_hdr
   Sequence {
```

```
        struct sub_item_hdr
        Choice {
           struct netting_swap  // Named struct no: 45
           struct netting_fra  // Named struct no: 46
           struct netting_fx  // Named struct no: 47
        }
     }
  }
```

3.6.76.6 Answer, structure contents

CA106 contains one transaction header structure followed by one or more variable structures.

Variable structure includes:

- Sub-item header including:
  - Named structure (a number that shows which structure that follows).
  - Size (total number of bytes in variable structure including this header).
- The actual data with structure type as given in the sub-item header.

Netting Swap

Fields usage in this structure:

Swap Side	is either Fixed or Float.
Payment Notional Amount	is the Notional Amount of the Swap Flow.
Payment	Pay amounts displayed as negative, receipt amounts as positive

Netting FRA

Fields usage in this structure:

Rate	is either the Fixed or Float Rate depending on Rate Type.
Side	Buy or Sell.
Consideration	is either Fixed or Float depending on Rate Type.
Difference	is a sign calculated from Side. For Side Buy the Float Difference is positive and Fixed Difference is negative. For Side Sell the Float Difference is negative and, Fixed Difference is positive.

Netting FX

Fields usage in this structure:

FX Side	is Buy or Sell depending on whether the payment amount is the Buy Amount or Sell Amount.
Amount	is either positive (Buy Amount) or negative (Sell Amount).

© The NASDAQ OMX Group, Inc. • 2013

Payment is same as Amount.

3.6.77 CQ128 [Query Account VIM QUERY]

3.6.77.1 Fingerprint

QUERY properties	
transaction type	CQ128
calling sequence	omniapi_query_ex
struct name	query_account
facility	EP5
partitioned	false
answers	CA128

VIA properties	
transaction type	CA128
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

3.6.77.2 Purpose

The purpose of this query is to retrieve account information for own accounts.

3.6.77.3 Structure

The CQ128 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
}
```

3.6.77.4 Usage and Conditions

Series

© The NASDAQ OMX Group, Inc. • 2013

is not relevant in this query and should be zero filled.

Segment Number

is one for the first query and then incremented.

A query can be executed 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 wild cards, and *Account Id* with wildcard or a value ending with wildcard. The answer contains all accounts according to the criteria.
3. Using *Date* as search criteria. The answer contains all accounts modified since the given *Business Date*. The field *Query on Date* must be set to true.

3.6.77.5 Answer Structure

The CA128 VIA has the following structure:

```
struct answer_account_hdr {
   struct transaction_type
   UINT16_T segment_number_n  // Segment Number
   UINT16_T items_n  // Items
}
Sequence {
   struct item_hdr
   Sequence {
      struct sub_item_hdr
      Choice {
         struct cl_account_base_api  // Named struct no: 81
         struct cl_account_risk_attribute_api  // Named struct no: 82
         struct cl_account_collateral_attribute_api  // Named struct no:
86
         struct cl_account_base_collateral_api  // Named struct no: 94
         struct cl_account_intraday_funding_api  // Named struct no: 97
      }
   }
}
```

3.6.77.6 Answer, comments

Item

If the maximum number of items for one transaction is returned, the query should be repeated with incremented segment number.

Margin class

© The NASDAQ OMX Group, Inc. • 2013

The effective margin class is returned. Margin class can be set on Account, Account type, Participant or Clearinghouse.

Answer contains one VIM item per account. Each Vim Item consists of three sub_items. One sub_item, **CL_ACCOUNT_BASE_API** (vim 81) holds basic information on the account. Next sub struct, **CL_ACCOUNT_RISK_ATTRIBUTE_API** (vim 82) holds risk parameters possible to set for an account, **CL_ACCOUNT_COLLATERAL_ATTRIBUTE_API** (vim 86) holds parameters related to collateral handling for the account and the last sub struct **CL_ACCOUNT_BASE_COLLATERAL_API** (vim 94) holds information related to base collateral

3.6.78 CQ146 [Query CL OTC Trade Operation QUERY]

3.6.78.1 Fingerprint

QUERY properties	
transaction type	CQ146
calling sequence	omniapi_query_ex
struct name	query_cl_otc_trade_operation
facility	EP5
partitioned	false
answers	CA146

VIA properties	
transaction type	CA146
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

3.6.78.2 Related Messages

CB146

3.6.78.3 Purpose

This query is used to retrieve Trade Operations for non Cash Flow OTC trades that have been subject to Clearinghouse Collateral Checks.

3.6.78.4 Structure

The CQ146 QUERY has the following structure:

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

© The NASDAQ OMX Group, Inc. • 2013

```
    char[8] business_date_s  // Date, Business
    UINT16_T segment_number_n  // Segment Number
    char[32] series_id_s  // Series, Identity
    UINT8_T le_state_c  // Type, Legal Event
    CHAR filler_1_s  // Filler
    struct account
}
```

3.6.78.5 Usage and Conditions

For a given business date, retrieve trade operations performed on trades that the user is eligible to see. It is possible to filter on a specific series.

Trade Operations can have state "Novated" or "Rejected", and sub state "Pending" if collateral check is just ongoing.

3.6.78.6 Answer Structure

The CA146 VIA has the following structure:

```
struct answer_cl_otc_trade_operation {
   struct transaction type
   UINT16_T segment_number_n  // Segment Number
   char[8] business_date_s  // Date, Business
   UINT16_T items_n  // Items
   UINT16_T size_n  // Size
}
Sequence {
   struct item_hdr
   Sequence {
      struct sub_item_hdr
      Choice {
         struct cl_otc_operation_info  // Named struct no: 95
         struct cl_otc_trade_operation  // Named struct no: 96
         struct risk_exposure_limit_vim  // Named struct no: 50010
      }
   }
}
```

3.6.78.7 Amswer, comments

One VIM item (and sub item) is returned per trade operation.

3.6.79 KB1 [Directed OTC Trade Report VIB]

3.6.79.1 Fingerprint

VIB properties	
transaction type	KB1
calling sequence	omniapi_read_event_ext_ex or omniapi_read_event_block

© The NASDAQ OMX Group, Inc. • 2013

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	dedicated

3.6.79.2 Related Messages

KQ1, KQ2, KQ3

3.6.79.3 Purpose

This broadcast will be sent when a Trade report is accepted and for every change that occurs with the trade report (e.g. content or state changes). For every change the trade report version is incremented.

3.6.79.4 Structure

The KB1 VIB has the following structure:

```
struct directed_trade_report {
   struct broadcast_type
   UINT8_T broadcast_reason_c  // Broadcast Reason
   char[3] filler_3_s  // Filler
   UINT16_T items_n  // Items
   UINT16_T size_n  // Size
}
Sequence {
   struct sub_item_hdr
   Choice {
      struct otc_trade_report_data  // Named struct no: 38002
      struct otc_base_trade_report  // Named struct no: 38001
      struct standard_trade_report  // Named struct no: 38009
      struct otc_fra_data  // Named struct no: 38004
      struct otc_fra_trade_report  // Named struct no: 38003
      struct otc_irs_data  // Named struct no: 38005
      struct otc_irs_trade_report  // Named struct no: 38006
      struct irs_member_pay  // Named struct no: 38007
      struct irs_counterparty_pay  // Named struct no: 38008
      struct otc_clearing_info  // Named struct no: 83
   }
}
```

3.6.79.5 Usage and Conditions

KB1 is a Variable Information Broadcast which sub items depends on the instrument.

It has some general sub item with general information applicable for all type of instruments:

- OTC trade report data
- OTC base trade report (was earlier embedded in the instrument specific structs)
- OTC Clearing info

© The NASDAQ OMX Group, Inc. • 2013

A number of other sub items are also included specific for different types of instrument.

For standard type of instrument:

- Standard trade report

For OTC FRA:

- OTC FRA trade report
- OTC FRA (was earlier within the OTC FRA trade report struct)

For IR SWAP:

- OTC IR SWAP trade report
- OTC IR SWAP (was earlier within the OTC IR SWAP trade report struct)
- IR SWAP Member Pay
- IR SWAP Counterparty Pay

3.6.80 KB10 [OTC Trade Operation on Hold VIB]

3.6.80.1 Fingerprint

VIB properties	
transaction type	KB10
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

3.6.80.2 Related Messages

KQ10

3.6.80.3 Purpose

This broadcast will be sent when a Trade Operation for an OTC Trade has been "Rejected" by the clearinghouse due to Clearinghouse Collateral Checks.

3.6.80.4 Structure

The KB10 VIB has the following structure:

```
struct broadcast_hdr
Sequence {
   struct sub_item_hdr
   Choice {
      struct otc_operation_info  // Named struct no: 38012
      struct otc_trade_operation  // Named struct no: 38013
      struct risk_exposure_limit_vim  // Named struct no: 50010
```


© The NASDAQ OMX Group, Inc. • 2013

```
    }
}
```

3.6.81 KC1 [Rectify OTC Trade Report VIT]

3.6.81.1 Fingerprint

VIT properties	
transaction type	KC1
calling sequence	omniapi_tx_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	false

3.6.81.2 Related Messages

If the transaction succeeds, a KB1 is sent out as confirmation of the change.

3.6.81.3 Purpose

This transaction is used to rectify a given trade report.

3.6.81.4 Structure

The KC1 VIT has the following structure:

```
struct rectify_otc_trade_report {
   struct transaction_type
   struct series  // Named struct no: 50000
   UINT64_T trade_report_nbr_q  // Trade report number
   UINT16_T items_n  // Items
   UINT16_T size_n  // Size
}
Sequence {
   struct item_hdr
   Sequence {
      struct sub_item_hdr
      Choice {
         struct otc_base_trade_report  // Named struct no: 38001
         struct standard_trade_report  // Named struct no: 38009
      }
   }
}
```

© The NASDAQ OMX Group, Inc. • 2013

3.6.81.5 Usage and Conditions

A successful rectification creates a new version of the trade report which is distributed to the parties by a KB1 broadcast.

The transaction contsists of the sub item “OTC Base trade report”

Followed by a the number of other sub items that are specific for the different types of instrument.

For standard type of instrument:

- Standard trade report

There are restrictions to what can be changed depending on the state of the trade report.

not yet matched

All fields may be rectified.

matched

- Free text fields such as Participant reference (Passthrough)
- Open/close request
- Account id. Here only the Account Id and not the Member part of the account can be changed.

Once matched the trade report cannot be rectified this way, since DvP instructions are created in real-time. However, If not yet settled, changes might be possible via the payment system or by manipulating (recall/split/resend) the DvP instructions.

Novated

It is not possible to change a standard trade report when in novated state. It is thereafter handled in the same way as for exchange traded trades.

3.6.81.6 Return Codes

Even if a rectify transaction is accepted by the system, it is possible that it will not be executed immediately. The below statuses give more information:

Completion	Cstatus	TxStat (reason code)
The rectify operation is performed.	Successful	OTC_NORMAL
The rectify operation is subject to collateral checks. If rejected, please refer to broadcast KB10. If approved, please refer to broadcast KB1.	Successful	OTC_COLLCHECK

Please refer to the *OMex System’s Error Messages* for details about why transactions are aborted.

© The NASDAQ OMX Group, Inc. • 2013

3.6.82 KC2 [Cancel OTC Trade Report TRANSACTION]

3.6.82.1 Fingerprint

TRANSACTION properties	
transaction type	KC2
calling sequence	omniapi_tx_ex
struct name	cancel_trade_report
facility	EP0
partitioned	false

3.6.82.2 Purpose

The purpose of this transaction is to cancel a trade report.

3.6.82.3 Structure

The KC2 TRANSACTION has the following structure:

```
struct cancel_trade_report {
   struct transaction_type
   struct series  // Named struct no: 50000
   UINT64_T trade_report_nbr_q  // Trade report number
   char[32] name_s  // Name
   UINT8_T confirm_reject_c  // Confirm or Reject
   char[3] filler_3_s  // Filler
}
```

3.6.82.4 Usage and Conditions

If a Trade Report is in a Pending Cancellation sub state, the pending cancellation can be rejected. Either side of the Trade Report can reject a pending cancellation on its own Trade Report. This means that a user can reject his outgoing cancellation because he changed his mind or made a typing error. The user receiving an incoming cancellation can also reject this if he doesn't wish to cancel the Trade Report.

No fields can be edited.

When cancelling an equity trade report, different conditions apply depending on the current state of the trade report.

unmatched	The user who entered the report may cancel the trade report without restrictions.
matched	Once matched, a cancellation is possible only prior to the Settlement Day and only if the Counterparty agrees (by confirming the cancellation). It is thus possible for the counterparty to reject a cancellation. In that case, or if the awaiting cancellation is never agreed upon, the original deal stays.

© The NASDAQ OMX Group, Inc. • 2013

novated	Once novated the trade report cannot be cancelled. Instead it can be fully terminated.

3.6.82.5 Return Codes

Even if a rectify transaction is accepted by the system, it is possible that it will not be executed immediately. The below statuses give more information:

Completion	Cstatus	TxStat (reason code)
The rectify operation is performed.	Successful	OTC_NORMAL
The rectify operation is subject to collateral checks. If rejected, please refer to broadcast KB10. If approved, please refer to broadcast KB1.	Successful	OTC_COLLCHECK

Please refer to the *OMex System's Error Messages* for details about why transactions are aborted.

3.6.83 KC5 [Clearing Member Accept or Reject OTC trade TRANSACTION]

3.6.83.1 Fingerprint

TRANSACTION properties	
transaction type	KC5
calling sequence	omniapi_tx_ex
struct name	accept_reject_trade_report_for_clearing
facility	EP0
partitioned	false

3.6.83.2 Related Messages

CB3, CQ80, CQ81, CQ82, KB1, KQ1, KQ2, KQ3

3.6.83.3 Purpose

This transaction is used by the Clearing Member to either accept or reject OTC trades which have been automatically given up to him.

3.6.83.4 Structure

The KC5 TRANSACTION has the following structure:

```
struct accept_reject_trade_report_for_clearing {
   struct transaction_type
   struct series  // Named struct no: 50000
   UINT64_T trade_report_nbr_q  // Trade report number
   char[32] name_s  // Name
   UINT8_T confirm_reject_c  // Confirm or Reject
```


© The NASDAQ OMX Group, Inc. • 2013

```
    char[3] filler_3_s  // Filler
}
```

3.6.83.5 Usage and conditions

When a trade is automatically given up for clearing, it is possible for the Clearing Member to require a possibility to either accept or reject the trade before it's taken up. A trade propagating into a clearing account where confirmation is required will remain in an unmatched state, with a sub state "Waiting for Clearing Member Accept" until it has been accepted. If the trade is accepted, it will continue its processing where if was put in a waiting state. If the trade is rejected by the Clearing Member, it will be set in a reject state.

Note:

This transaction may be rejected, in case one is trying to act on a trade for which one is not entitled to perform this action.

3.6.83.6 Return Codes

Even if a rectify transaction is accepted by the system, it is possible that it will not be executed immediately. The below statuses give more information:

Completion	Cstatus	TxStat (reason code)
The rectify operation is performed.	Successful	OTC_NORMAL
The rectify operation is subject to collateral checks. If rejected, please refer to broadcast KB10. If approved, please refer to broadcast KB1.	Successful	OTC_COLLCHECK

Please refer to the *OMex System's Error Messages* for details about why transactions are aborted.

3.6.84 KQ1 [OTC Trade Report QUERY]

3.6.84.1 Fingerprint

QUERY properties	
transaction type	KQ1
calling sequence	omniapi_query_ex
struct name	query_trade_report_otc
facility	EP0
partitioned	true
answers	KA1

VIA properties	
transaction type	KA1

© The NASDAQ OMX Group, Inc. • 2013

VIA 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.
segmented	true

3.6.84.2 Related Messages

KB1, KQ2, KQ3

3.6.84.3 Purpose

This query is used to retrieve Trade Reports.

3.6.84.4 Structure

The KQ1 QUERY has the following structure:

```
struct query_trade_report_otc {
   struct transaction_type
   struct series  // Named struct no: 50000
   struct party
   struct account
   UINT16_T segment_number_n  // Segment Number
   char[2] filler_2_s  // Filler
   char[8] from_date_s  // Date, From
   char[8] to_date_s  // Date, To
   char[32] passthrough_s  // Passthrough Information
   char[32] series_id_s  // Series, Identity
   UINT32_T trade_report_type_i  // Trade Report Type
   UINT8_T trade_report_state_c  // Trade Report State
   UINT8_T bought_or_sold_c  // Bought or Sold
   UINT8_T date_span_type_c  // Date Span Type
   CHAR filler_1_s  // Filler
}
```

3.6.84.5 Usage and Conditions

For a given business date, retrieve trade operations performed on trades that the user is eligible to see. It is possible to filter on a specific series, account, settlement date range, passthrough and trade report state.

3.6.84.6 Answer Structure

The KA1 VIA has the following structure:

```
struct answer_trade_report {
   struct transaction_type
   struct partition_low
   struct partition_high
   UINT16_T segment_number_n  // Segment Number
   UINT16_T items_n  // Items
```


© The NASDAQ OMX Group, Inc. • 2013

```
}
Sequence {
   struct item hdr
   Sequence {
      struct sub item hdr
      Choice {
         struct otc trade report data  // Named struct no: 38002
         struct otc base trade report  // Named struct no: 38001
         struct standard trade report  // Named struct no: 38009
         struct otc fra data  // Named struct no: 38004
         struct otc fra trade report  // Named struct no: 38003
         struct otc irs data  // Named struct no: 38005
         struct otc irs trade report  // Named struct no: 38006
         struct irs member pay  // Named struct no: 38007
         struct irs counterparty pay  // Named struct no: 38008
         struct otc clearing info  // Named struct no: 83
      }
   }
}
```

3.6.84.7 Answer, Comments

A VIM item (and sub item) is returned per trade report.

The sub items depends on the instrument with a number of general items applicable for all type of instruments:

- OTC trade report data
- OTC base trade report (was earlier embedded in the instrument specific structs)
- OTC Clearing info

A number of other sub items are also included specific for different types of instrument.

For standard type of instrument:

- Standard trade report

For OTC FRA:

- OTC FRA trade report
- OTC FRA (was earlier within the OTC FRA trade report struct)

For IR SWAP:

- OTC IR SWAP trade report
- OTC IR SWAP (was earlier within the OTC IR SWAP trade report struct)
- IR SWAP Member Pay
- IR SWAP Counterparty Pay

© The NASDAQ OMX Group, Inc. • 2013

3.6.85 KQ2 [OTC Trade Report QUERY]

3.6.85.1 Fingerprint

QUERY properties	
transaction type	KQ2
calling sequence	omniapi_query_ex
struct name	query_missing_trade_report
facility	EP0
partitioned	false
answers	KA2

VIA properties	
transaction type	KA2
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

3.6.85.2 Related Messages

KB1, KQ1, KQ3

3.6.85.3 Purpose

This query is used to retrieve missing Trade Reports.

3.6.85.4 Structure

The KQ2 QUERY has the following structure:

```
struct query_missing_trade_report {
   struct transaction_type
   struct series  // Named struct no: 50000
   UINT32_T sequence_first_u  // Sequence First
   UINT32_T sequence_last_u  // Sequence Last
   char[8] timestamp_date_s  // Timestamp, Date
}
```

3.6.85.5 Answer Structure

The KA2 VIA has the following structure:

```
    struct answer_missing_trade_report {
```


© The NASDAQ OMX Group, Inc. • 2013

```
      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 otc_trade_report_data  // Named struct no: 38002
            struct otc_base_trade_report  // Named struct no: 38001
            struct standard_trade_report  // Named struct no: 38009
            struct otc_fra_data  // Named struct no: 38004
            struct otc_fra_trade_report  // Named struct no: 38003
            struct otc_irs_data  // Named struct no: 38005
            struct otc_irs_trade_report  // Named struct no: 38006
            struct irs_member_pay  // Named struct no: 38007
            struct irs_counterparty_pay  // Named struct no: 38008
            struct otc_clearing_info  // Named struct no: 83
         }
      }
   }
```

3.6.85.6 Answer, Comments

See KQ1.

3.6.86 KQ3 [OTC Trade Report Version QUERY]

3.6.86.1 Fingerprint

QUERY properties	
transaction type	KQ3
calling sequence	omniapi_query_ex
struct name	query_trade_report_version
facility	EP0
partitioned	false
answers	KA3

VIA properties	
transaction type	KA3
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

© The NASDAQ OMX Group, Inc. • 2013

3.6.86.2 Related Messages

KB1, KQ1, KQ2

3.6.86.3 Purpose

This query returns all versions of a Trade Report.

3.6.86.4 Structure

The KQ3 QUERY has the following structure:

```
struct query_trade_report_version {
   struct transaction_type
   struct series  // Named struct no: 50000
   UINT64_T trade_report_nbr_q  // Trade report number
   UINT16_T segment_number_n  // Segment Number
   UINT16_T items_n  // Items
}
```

3.6.86.5 Answer Structure

The KA3 VIA has the following structure:

```
    struct answer_trade_report {
       struct transaction_type
       struct partition_low
       struct partition_high
       UINT16_T segment_number_n  // Segment Number
       UINT16_T items_n  // Items
    }
    Sequence {
       struct item_hdr
       Sequence {
          struct sub_item_hdr
          Choice {
             struct otc_trade_report_data  // Named struct no: 38002
             struct otc_base_trade_report  // Named struct no: 38001
             struct standard_trade_report  // Named struct no: 38009
             struct otc_fra_data  // Named struct no: 38004
             struct otc_fra_trade_report  // Named struct no: 38003
             struct otc_irs_data  // Named struct no: 38005
             struct otc_irs_trade_report  // Named struct no: 38006
             struct irs_member_pay  // Named struct no: 38007
             struct irs_counterparty_pay  // Named struct no: 38008
             struct otc_clearing_info  // Named struct no: 83
          }
       }
    }
```

3.6.86.6 Answer, Comments

See KQ1.

© The NASDAQ OMX Group, Inc. • 2013

3.6.87 KQ10 [Query OTC Trade Operation, External QUERY]

3.6.87.1 Fingerprint

QUERY properties	
transaction type	KQ10
calling sequence	omniapi_query_ex
struct name	query_otc_trade_operation
facility	EP5
partitioned	false
answers	KA10

VIA properties	
transaction type	KA10
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

3.6.87.2 Related Messages

KB10

3.6.87.3 Purpose

This query is used to retrieve Trade Operations that have been subject to Clearinghouse Collateral Checks.

3.6.87.4 Structure

The KQ10 QUERY has the following structure:

```
struct query_otc_trade_operation {
   struct transaction_type
   struct series  // Named struct no: 50000
   char[8] business_date_s  // Date, Business
   UINT16_T segment_number_n  // Segment Number
   char[2] filler_2_s  // Filler
}
```

3.6.87.5 Usage and Conditions

For a given business date, retrieve trade operations performed on trades that the user is eligible to see. It is possible to filter on a specific series.

© The NASDAQ OMX Group, Inc. • 2013

Trade Operations can have state "Novated" or "Rejected", and sub state "Pending" if collateral check is just ongoing.

3.6.87.6 Answer Structure

The KA10 VIA has the following structure:

```
struct answer_otc_trade_operation {
   struct transaction_type
   UINT16_T segment_number_n  // Segment Number
   char[8] business_date_s  // Date, Business
   UINT16_T items_n  // Items
   UINT16_T size_n  // Size
}
Sequence {
   struct item_hdr
   Sequence {
      struct sub_item_hdr
      Choice {
         struct otc_operation_info  // Named struct no: 38012
         struct otc_trade_operation  // Named struct no: 38013
         struct risk_exposure_limit_vim  // Named struct no: 50010
         struct otc_trade  // Named struct no: 38014
      }
   }
}
```

3.6.87.7 Answer, Comments

One VIM item (and sub item) is returned per trade operation.

3.6.88 VC1 [Register Physical Delivery TRANSACTION]

3.6.88.1 Fingerprint

TRANSACTION properties	
transaction type	VC1
calling sequence	omniapi_tx_ex
struct name	reg_physical_delivery
facility	EP5
partitioned	false

3.6.88.2 Purpose

The purpose of this transaction is to register (connect to a synthetic delivery) or rectify physical deliveries. To rectify registration (change/add/remove) of physical deliveries to a synthetic delivery all new physical deliveries must be registered again. The old ones are disconnected.

© The NASDAQ OMX Group, Inc. • 2013

3.6.88.3 Structure

The VC1 TRANSACTION has the following structure:

```
struct reg_physical_delivery {
   struct transaction_type
   struct series  // Named struct no: 50000
   INT32_T event_origin_i  // Event, Origin
   INT32_T class_no_i  // Class Number
   INT32_T sequence_no_i  // Number, Sequence
   UINT16_T items_n  // Items
   char[2] filler_2_s  // Filler
   Array ITEM [max no: 20] {
      struct physical_series
      INT64_T deliv_base_quantity_q  // Quantity, Delivery Base
   }
}
```

3.6.89 VQ1 [Underlying Delivery QUERY]

3.6.89.1 Fingerprint

QUERY properties	
transaction type	VQ1
calling sequence	omniapi_query_ex
struct name	query_cl_underlying_delivery
facility	EP5
partitioned	false
answers	VA1

ANSWER properties	
transaction type	VA1
struct name	answer_cl_underlying_delivery
segmented	false

3.6.89.2 Purpose

The purpose of this query is to retrieve all physical underlyings which are aimed for delivery instead of a synthetic underlying.

3.6.89.3 Structure

The VQ1 QUERY has the following structure:

```
struct query_cl_underlying_delivery {
```

© The NASDAQ OMX Group, Inc. • 2013

```
    struct transaction_type
    struct series  // Named struct no: 50000
    UINT8_T state_c  // State
    char[3] filler_3_s  // Filler
}
```

3.6.89.4 Usage and conditions

The series must be fully specified. It is possible to ask for only active physical underlyings (state active) or only rejected physical underlyings (state rejected.)

3.6.89.5 Answer Structure

The VA1 ANSWER has the following structure:

```
struct answer_cl_underlying_delivery {
   struct transaction_type
   UINT16_T items_n  // Items
   char[2] filler_2_s  // Filler
   Array ITEM [max no: 300] {
      char[8] created_date_s  // Date, Created
      char[8] rejected_date_s  // Date, Rejected
      struct trading_code
      struct physical_series
      struct series  // Named struct no: 50000
      INT32_T bond_quotation_i  // Bond Quotation
      UINT16_T dec_in_bq_n  // Decimals, Bond Quotation
      UINT8_T state_c  // State
      CHAR filler_1_s  // Filler
   }
}
```

3.6.89.6 Answer, comments

The response received is a list of all physical underlyings for a specific synthetic underlying.

3.6.90 VQ2 [Physical Delivery QUERY]

3.6.90.1 Fingerprint

QUERY properties	
transaction type	VQ2
calling sequence	omniapi_query_ex
struct name	query_physical_delivery
facility	EP5
partitioned	false
answers	VA2

© The NASDAQ OMX Group, Inc. • 2013

ANSWER properties	
transaction type	VA2
struct name	answer_physical_delivery
segmented	false

3.6.90.2 Purpose

The purpose of this query is to retrieve all physical deliveries for a specific synthetic delivery or all physical deliveries for a specific closing date.

3.6.90.3 Structure

The VQ2 QUERY has the following structure:

```
struct query_physical_delivery {
   struct transaction_type
   struct series  // Named struct no: 50000
   UINT16_T segment_number_n  // Segment Number
   char[2] filler_2_s  // Filler
   INT32_T event_origin_i  // Event, Origin
   char[8] as_of_date_s  // Date, As Of
}
```

3.6.90.4 Usage and conditions

There are two ways to ask for physical deliveries:

1. Fill in synthetic series and the event origin number, i.e. the event identifier number from the synthetic delivery. The answer contains physical deliveries for a specific synthetic delivery. (Wildcards are not allowed for series).
2. Fill in the closing date. The answer contains all physical deliveries for that specific closing date.

3.6.90.5 Answer Structure

The VA2 ANSWER has the following structure:

```
struct answer_physical_delivery {
   struct transaction_type
   UINT16_T segment_number_n  // Segment Number
   UINT16_T items_n  // Items
   Array ITEM [max no: 200] {
      struct confirmed_by {
         char[2] country_id_s  // Name, Country
         char[5] ex_customer_s  // Customer, Identity
         char[5] user_id_s  // User
      }
      struct trading_code
      struct physical_series
      struct series  // Named struct no: 50000
```

© The NASDAQ OMX Group, Inc. • 2013

```
        struct account_from {
           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 account_to {
           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
        }
        INT64_T deliv_base_quantity_q  // Quantity, Delivery Base
        INT32_T bond_quotation_i  // Bond Quotation
        INT32_T event_type_i  // Stimuli Event
        INT32_T event_origin_i  // Event, Origin
        INT32_T sequence_no_i  // Number, Sequence
        INT32_T class_no_i  // Class Number
        UINT16_T dec_in_bq_n  // Decimals, Bond Quotation
        char[8] created_date_s  // Date, Created
        char[8] as_of_date_s  // Date, As Of
        char[8] modified_date_s  // Date, Modified
        char[20] csd_account_from_s  // CSD Account, From
        char[20] csd_account_to_s  // CSD Account, To
        char[8] settlement_date_s  // Date, Settlement
        char[3] currency_s  // Currency
        UINT8_T state_c  // State
        char[2] filler_2_s  // Filler
     }
  }
```

3.6.90.6 Answer, comments

The response received is a list of all physical deliveries with respect to the selection.

3.7 Risk Management

3.7.1 CQ41 [Query cash flow for sim VIQ]

3.7.1.1 Fingerprint

VIQ properties	
transaction type	CQ41
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	EP6
partitioned	true

© The NASDAQ OMX Group, Inc. • 2013

VIQ properties	
answers	CA41

VIA properties	
transaction type	CA41
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

3.7.1.2 Related Messages

JQ71

3.7.1.3 Purpose

This query is used to query for cash flows for each OTC trade to be included in a margin simulation. The cash flows returned in this answer should then be used as input to query JQ71, including a sequential trade number which ties each flow to its trade. Cash flows may be retrieved for swap trades and TM FRA trades.

3.7.1.4 Structure

The CQ41 VIQ has the following structure:

```
struct query_cash_flows_for_sim {
   struct transaction_type
   struct series  // Named struct no: 50000
   UINT16_T segment_number_n  // Segment Number
   char[2] filler_2_s  // Filler
   UINT16_T items_n  // Items
   UINT16_T size_n  // Size
}
Sequence {
   struct item_hdr
   Sequence {
      struct sub_item_hdr
      Choice {
         struct ir_swap_flow_for_sim  // Named struct no: 75
         struct fra  // Named struct no: 85
      }
   }
}
```

3.7.1.5 Usage and conditions

The query is a VIQ query, which can either be filled with data for a swap or data for a Tailor made FRA.

3.7.1.6 Answer Structure

The CA41 VIA has the following structure:

© The NASDAQ OMX Group, Inc. • 2013

```
struct answer_cash_flows_for_sim {
   struct transaction_type
   UINT16_T segment_number_n  // Segment Number
   char[2] filler_2_s  // Filler
   UINT16_T items_n  // Items
   UINT16_T size_n  // Size
}
Sequence {
   struct item_hdr
   Sequence {
      struct sub_item_hdr
      Choice {
         struct otc_cash_flow_base  // Named struct no: 65
         struct otc_cash_flow_info  // Named struct no: 66
      }
   }
}
```

3.7.1.7 Answer, comments

Cash flows are returned in a format that can later be used as input to the Margin Simulation Query, JQ71.

3.7.2 EQ10 [Yield Curve Names QUERY]

3.7.2.1 Fingerprint

QUERY properties	
transaction type	EQ10
calling sequence	omniapi_query_ex
struct name	query_yield_curve_names
facility	EP5
partitioned	false
answers	EA10

VIA properties	
transaction type	EA10
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

3.7.2.2 Purpose

This query is used to retrieve all defined curve ID’s and their respective names, and data.

© The NASDAQ OMX Group, Inc. • 2013

3.7.2.3 Structure

The EQ10 QUERY has the following structure:

```
struct query_yield_curve_names {
   struct transaction_type
   struct series  // Named struct no: 50000
   UINT16_T segment_number_n  // Segment Number
   char[2] filler_2_s  // Filler
   char[16] filler_16_s  // Filler
}
```

3.7.2.4 Usage and Conditions

Series

should be zero filled.

3.7.2.5 Answer Structure

The EA10 VIA has the following structure:

```
struct answer_yield_curve_names {
   struct transaction_type
   UINT16_T segment_number_n  // Segment Number
   UINT16_T items_n  // Items
   UINT16_T size_n  // Size
}
Sequence {
   struct item_hdr
   Sequence {
      struct sub_item_hdr
      Choice {
         struct yield_curve_names  // Named struct no: 20000
      }
   }
}
```

3.7.2.6 Answer, Comments

For each defined curve, a name and an id is returned. The id is then to be used in queries for curve information etc.

3.7.3 JB1 [Margin Calculation Runs VIB]

3.7.3.1 Fingerprint

VIB properties	
transaction type	JB1

© The NASDAQ OMX Group, Inc. • 2013

VIB properties	
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

3.7.3.2 Related Messages

JQ1

3.7.3.3 Purpose

This broadcast gives information on new Margin Calculation Runs. A Margin calculation run will calculate margins for all accounts.

3.7.3.4 Structure

The JB1 VIB has the following structure:

```
struct bdx_marg_calc_runs {
   struct broadcast_type
   char[8] business_date_s  // Date, Business
   UINT16_T items_n  // Items
   UINT16_T size_n  // Size
}
Sequence {
   struct sub_item_hdr
   Choice {
      struct marg_calc_runs  // Named struct no: 21000
   }
}
```

3.7.3.5 Usage and Conditions

This broadcast give information about a new calculation run, for which a sequence number is returned. This sequence number can be used when querying for specific data for this run.

3.7.4 JB2 [Margin Calculation Runs, dedicated VIB]

3.7.4.1 Fingerprint

VIB properties	
transaction type	JB2
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.

© The NASDAQ OMX Group, Inc. • 2013

VIB properties	
info type	dedicated

3.7.4.2 Related Messages

JQ1

3.7.4.3 Purpose

This broadcast gives information on new Margin Calculation Runs which have been done for a single account or for a subset of accounts within the same participant.

3.7.4.4 Structure

The JB2 VIB has the following structure:

```
struct bdx_marg_calc_runs {
   struct broadcast_type
   char[8] business_date_s  // Date, Business
   UINT16_T items_n  // Items
   UINT16_T size_n  // Size
}
Sequence {
   struct sub_item_hdr
   Choice {
      struct marg_calc_runs  // Named struct no: 21000
      struct account_vim  // Named struct no: 50005
   }
}
```

3.7.4.5 Usage and Conditions

This broadcast give information about a new calculation run, for which a sequence number is returned. This sequence number can be used when querying for specific data for this run.

Margin Calculation Runs for a single account, shows the account filled in explicitly. Margin Calculation Runs for a subset of accounts, shows the account struct with information about the member only, and the account field is set to '*'.

3.7.5 JQ1 [Margin Calculation Runs QUERY]

3.7.5.1 Fingerprint

QUERY properties	
transaction type	JQ1
calling sequence	omniapi_query_ex
struct name	query_marg_calc_runs
facility	EP5

© The NASDAQ OMX Group, Inc. • 2013

QUERY properties	
partitioned	false
answers	JA1

VIA properties	
transaction type	JA1
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

3.7.5.2 Purpose

This query is used to retrieve information for margin calculation runs. A Margin calculation run will calculate margins for all accounts. For each returned calculation run, a sequence number is returned, and this sequence number can be used when querying for specific data for this run.

3.7.5.3 Structure

The JQ1 QUERY has the following structure:

```
struct query_marg_calc_runs {
   struct transaction_type
   struct series  // Named struct no: 50000
   char[8] business_date_s  // Date, Business
   UINT16_T segment_number_n  // Segment Number
   UINT8_T run_type_c  // Run Type
   char[12] clh_id_s  // Clearinghouse
   CHAR filler_1_s  // Filler
}
```

3.7.5.4 Usage and Conditions

Series

should be filled with zero.

Date, Business

should be filled in with today's business date or a previous date.

Run Type

should be set to zero or set to:

- **End Of Day**: answer will hold calculation runs of type End Of Day.
- **Intraday**: answer will hold calculation runs of type Intraday.
- **Call**: answer will hold calculation runs of type Call.
- **Preliminary**: answer will hold calculation runs of type Preliminary.

© The NASDAQ OMX Group, Inc. • 2013

If set to zero answer will hold calculation runs of all types.

Clearinghouse

is not used and can be left blank.

3.7.5.5 Answer Structure

The JA1 VIA has the following structure:

```
struct answer_marg_calc_runs {
   struct transaction type
   UINT16_T segment_number_n  // Segment Number
   char[8] business_date_s  // Date, Business
   UINT16_T items_n  // Items
   UINT16_T size_n  // Size
}
Sequence {
   struct item_hdr
   Sequence {
      struct sub_item_hdr
      Choice {
         struct marg_calc_runs  // Named struct no: 21000
         struct account_vim  // Named struct no: 50005
      }
   }
}
```

3.7.5.6 Answer, Comments

Query will return data for the clearinghouse that the user is connected to.

When a Margin Calculation Runs have been done for a single account an account struct shows the account filled in explicitly. When a Margin Calculation Run has been done for a subset of accounts, an account struct shows information about the member only, and the account field is set to '*'.

3.7.6 JQ15 [Stress factors for Yield Curve QUERY]

3.7.6.1 Fingerprint

QUERY properties	
transaction type	JQ15
calling sequence	omniapi_query_ex
struct name	query_stress_factors_for_yield_curve
facility	EP5
partitioned	false
answers	JA15

VIA properties	
transaction type	JA15
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

3.7.6.2 Related Messages

JQ1, JQ16

3.7.6.3 Purpose

This query is used to retrieve the stressing factors and curve correlation object for a Curve Stressing object.

3.7.6.4 Structure

The JQ15 QUERY has the following structure:

```
struct query_stress_factors_for_yield_curve {
   struct transaction_type
   struct series  // Named struct no: 50000
   INT32_T sequence_number_n  // Sequence Number
   char[12] stress_crv_id_s  // Stress Curve Id
   char[8] business_date_s  // Date, Business
   char[12] clh_id_s  // Clearinghouse
   UINT16_T segment_number_n  // Segment Number
   UINT8_T run_type_c  // Run Type
   CHAR filler_1_s  // Filler
}
```

3.7.6.5 Usage and Conditions

Series

should be zero filled, or with a series for which the curve is used as forcasting or discounting curve. If these two are different, answer will contain two curves.

Date, Business

should be filled in with today’s business date or a previous date. If set to blank, data for the latest available date is returned.

Curve Stressing ID

should be filled with the requested curve stressing ID, or if series is given, set to blank.

Sequence Number

© The NASDAQ OMX Group, Inc. • 2013

If given, the parameters for a specific batch run is returned. If set to blank, the parameters for the latest available run are returned.

Run Type

should be filled with **EndOfDay, Intraday, Call, Preliminary** or **None**.

- **EndOfDay**: sequence number is N/A, answer will hold latest available end of day data for given business date.
- **Intraday**: for given sequence number. Sequence number = 0 gives data for latest available intraday run.
- **Call**: for given sequence number. Sequence number = 0 gives data for latest available call run.
- **None**: sequence number and business date are N/A, current setting, independent if it has been used in any run or not, is returned.

Clearinghouse

is not used and can be left blank.

3.7.6.6 Answer Structure

The JA15 VIA has the following structure:

```
struct answer_stress_factors_for_yield_curve {
   struct transaction_type
   INT32_T sequence_number_n  // Sequence Number
   char[8] business_date_s  // Date, Business
   UINT16_T segment_number_n  // Segment Number
   UINT16_T items_n  // Items
   UINT16_T size_n  // Size
   UINT8_T run_type_c  // Run Type
   CHAR filler_1_s  // Filler
}
Sequence {
   struct item_hdr
   Sequence {
      struct sub_item_hdr
      Choice {
         struct stress_factors_for_yield_curve  // Named struct no: 21001
         struct principal_factors  // Named struct no: 21002
      }
   }
}
```

3.7.7 JQ16 [Curve Correlation Parameters QUERY]

3.7.7.1 Fingerprint

QUERY properties	
transaction type	JQ16

QUERY properties	
calling sequence	omniapi_query_ex
struct name	query_rm_crvcorr_param
facility	EP5
partitioned	false
answers	JA16

VIA properties	
transaction type	JA16
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

3.7.7.2 Purpose

This query is used to retrieve information about a given curve correlation cube object, either at a given margin calculation run, or the current setting.

3.7.7.3 Structure

The JQ16 QUERY has the following structure:

```
struct query_rm_crvcorr_param {
   struct transaction_type
   struct series  // Named struct no: 50000
   char[8] business_date_s  // Date, Business
   char[12] ccc_id_s  // Curve Correlation Cube
   INT32_T sequence_number_n  // Sequence Number
   UINT16_T segment_number_n  // Segment Number
   char[3] margin_class_s  // Margin class
   char[12] clh_id_s  // Clearinghouse
   UINT8_T margin_class_filter_c  // Margin Class Filter
   UINT8_T run_type_c  // Run Type
   CHAR filler_1_s  // Filler
}
```

3.7.7.4 Usage and Conditions

Series

should be filled with zero.

Date, Business

© The NASDAQ OMX Group, Inc. • 2013

should be filled in with today's business date or a previous date. If set to blank, data for the latest available date is returned.

Sequence Number

If non-zero, the parameters for a margin calculation run is returned. If set to zero, the parameters for the latest available run are returned.

Run Type

should be filled with **End Of Day**, **Intraday**, **Call** or **Preliminary**.

- **End Of Day**: sequence number is N/A, answer will hold latest available data for given business date.
- **Intraday**: for given sequence number. Sequence number = 0 gives data for latest available intraday run.
- **Call**: for given sequence number. Sequence number = 0 gives data for latest available call run.

Curve Correlation Cube

If given, only data for the given Curve Correlation Cube ID is returned, and Margin Class Filter and Margin Class are ignored. If set to blank, data matching the Margin Class Filter and Margin Class is returned.

Margin Class Filter

should be filled in with one of the below

- **Specific**: for given Margin Class.
- **Relevant for me**: Margin Class field is N/A. Data for all margin classes applicable for the Participant is returned, that is, also margin classes used for any accounts under the participant.
- **All**: Margin Class field is N/A. Data for all margin classes is returned.
- **Default**: this is what will be used by the backwards compatible API, where it is not possible to specify any margin class parameters at all. Margin Class field is N/A. Data for the margin class applicable for the Participant is returned, but not margin classes specified for certain accounts under the participant.

Margin Class

should be filled in if Margin Class Filter is set to Specific, otherwise blank.

Clearinghouse

is not used and can be left blank.

3.7.7.5 Answer Structure

The JA16 VIA has the following structure:

```
struct answer_rm_crvcorr_param {
   struct transaction_type
   INT32_T sequence_number_n  // Sequence Number
   char[8] business_date_s  // Date, Business
   UINT16_T segment_number_n  // Segment Number
   UINT16_T items_n  // Items
   UINT16_T size_n  // Size
   UINT8_T run_type_c  // Run Type
```

© The NASDAQ OMX Group, Inc. • 2013

```
      CHAR filler_1_s  // Filler
   }
   Sequence {
      struct item_hdr
      Sequence {
         struct sub_item_hdr
         Choice {
            struct crvcorr_param  // Named struct no: 21013
         }
      }
   }
```

3.7.7.6 Answer, Comments

Query will return data for the clearinghouse to which the user is connected.

3.7.8 JQ21 [Query risk margin scaling factor QUERY]

3.7.8.1 Fingerprint

QUERY properties	
transaction type	JQ21
calling sequence	omniapi_query_ex
struct name	query_risk_margin_scaling_factor
facility	EP5
partitioned	false
answers	JA21

VIA properties	
transaction type	JA21
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

3.7.8.2 Purpose

The purpose of this query is to retrieve the risk scale factor per market that might be imposed on own account or account within own organization.

3.7.8.3 Structure

The JQ21 QUERY has the following structure:

```
struct query_risk_margin_scaling_factor {
```


© The NASDAQ OMX Group, Inc. • 2013

```
    struct transaction_type
    struct series  // Named struct no: 50000
    struct account
    INT32_T sequence_number_n  // Sequence Number
    UINT16_T segment_number_n  // Segment Number
    UINT8_T run_type_c  // Run Type
    char[8] business_date_s  // Date, Business
    CHAR filler_1_s  // Filler
}
```

3.7.8.4 Usage and Conditions

Series

should be zero filled.

Account

should all be filled in with values in one of the following ways:

- with explicit value. All answers must match the field.
- with "*". No test is made on the value for that field.
- with a string ended by "*". All answers must in this field start with the string specified.

Date, Business

should be filled in with today's business date or a previous date. If set to blank, data for the latest available date is returned.

SequenceNumber

If not set to zero, the parameters for a margin calculation run are returned. If set to zero, the parameters for the latest available run are returned.

Run Type

should be filled with **EndOfDay, Intraday, Call, Preliminary** or **None**:

- **EndOfDay** (sequence number is N/A, answer will hold latest available data for given business date)
- **Intraday** (for given sequence number. Sequence number = 0 gives data for latest available intraday run)
- **Call** (for given sequence number. Sequence number = 0 gives data for latest available call run)
- **None** (sequence number and business date are N/A. Current setting, regardless of whether it has been used in any run or not, is returned)

3.7.8.5 Answer Structure

The JA21 VIA has the following structure:

```
struct answer_risk_margin_scaling_factor {
   struct transaction_type
   UINT16_T segment_number_n  // Segment Number
```

```
      char[8] business_date_s  // Date, Business
      UINT16_T items_n  // Items
      UINT16_T size_n  // Size
   }
   Sequence {
      struct item_hdr
      Sequence {
         struct sub_item_hdr
         Choice {
            struct risk_scale  // Named struct no: 21043
         }
      }
   }
```

3.7.8.6 Answer, Comments

Only accounts that have a *Risk marign scaling factor* set will be included in the answer.

3.7.9 JQ22 [Query Margin Aggregation Groups QUERY]

3.7.9.1 Fingerprint

QUERY properties	
transaction type	JQ22
calling sequence	omniapi_query_ex
struct name	query_margin_aggregation_groups
facility	EP5
partitioned	false
answers	JA22

VIA properties	
transaction type	JA22
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

3.7.9.2 Purpose

This query is used to query for margin aggregation groups, stored per owner participant.

3.7.9.3 Structure

The JQ22 QUERY has the following structure:

© The NASDAQ OMX Group, Inc. • 2013

```
struct query_margin_aggregation_groups {
   struct transaction_type
   struct series  // Named struct no: 50000
   UINT16_T segment_number_n  // Segment Number
   char[2] filler_2_s  // Filler
   struct margin_aggregation_group  // Of type: ACCOUNT
}
```

3.7.9.4 Usage and Conditions

Series

should be zero filled.

Margin Aggregation Group

should all be filled in with values in one of the following ways:

- with explicit value. All answers must match the field.
- with "*". No test is made on the value for that field.
- with a string ended by "*". All answers must in this field start with the string specified.

3.7.9.5 Answer Structure

The JA22 VIA has the following structure:

```
struct answer_margin_aggregation_groups {
   struct transaction_type
   UINT16_T segment_number_n  // Segment Number
   UINT16_T items_n  // Items
}
Sequence {
   struct item_hdr
   Sequence {
      struct sub_item_hdr
      Choice {
         struct margin_aggregation_group_info  // Named struct no: 21073
      }
   }
}
```

3.7.9.6 Answer, Comments

Trading code

is the user last updating or creating the item.

Date, Created and Time, Created

is the date and time when the item was created.

Date, Modified and Time, Modified

is the date and time when the item was modified.

3.7.10 JQ23 [Query Margin Aggregation Group Detail QUERY]

3.7.10.1 Fingerprint

QUERY properties	
transaction type	JQ23
calling sequence	omniapi_query_ex
struct name	query_margin_aggregation_group_detail
facility	EP5
partitioned	false
answers	JA23

VIA properties	
transaction type	JA23
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

3.7.10.2 Purpose

This query is used to query for details about one margin aggregation group.

3.7.10.3 Structure

The JQ23 QUERY has the following structure:

```
struct query_margin_aggregation_group_detail {
   struct transaction_type
   struct series  // Named struct no: 50000
   UINT16_T segment_number_n  // Segment Number
   char[2] filler_2_s  // Filler
   struct margin_aggregation_group  // Of type: ACCOUNT
}
```

3.7.10.4 Usage and Conditions

Series

© The NASDAQ OMX Group, Inc. • 2013

should be zero filled.

Margin Aggregation Group

should all be filled in with values in one of the following way:

- with explicit value. All answers must match the field.

3.7.10.5 Answer Structure

The JA23 VIA has the following structure:

```
    struct answer_margin_aggregation_group_detail {
       struct transaction_type
       UINT16_T segment_number_n  // Segment Number
       UINT16_T items_n  // Items
       struct margin_aggregation_group  // Of type: ACCOUNT
       char[3] margin_class_s  // Margin class
       char[3] risk_currency_s  // Currency, Risk
       char[40] description_s  // Description
       char[12] risk_scale_s  // Risk scale
       char[2] filler_2_s  // Filler
       INT32_T version_i  // VERSION_I
       UINT8_T risk_cur_conv_c  // Risk, Currency Conversion
       char[3] filler_3_s  // Filler
    }
    Sequence {
       struct item_hdr
       Sequence {
          struct sub_item_hdr
          Choice {
             struct answer_margin_aggregation_group_row  // Named struct no:
21071
          }
       }
    }
```

3.7.11 JQ24 [Query Var Parameter QUERY]

3.7.11.1 Fingerprint

QUERY properties	
transaction type	JQ24
calling sequence	omniapi_query_ex
struct name	query_var_parameters
facility	EP5
partitioned	false
answers	JA24

VIA properties	
transaction type	JA24
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

3.7.11.2 Purpose

This query is used to retrieve the margin parameters that have been used for calculating margin requirements for instrument series have the Value at Risk (FX margin) margin model.

3.7.11.3 Structure

The JQ24 QUERY has the following structure:

```
struct query_var_parameters {
   struct transaction_type
   struct series  // Named struct no: 50000
   INT32_T sequence_number_n  // Sequence Number
   UINT16_T segment_number_n  // Segment Number
   char[8] margin_date_s  // Margin Date
   char[3] margin_class_s  // Margin class
   char[16] fxm_id_s  // FX margin parameters, Identity
   char[12] clh_id_s  // Clearinghouse
   UINT8_T run_type_c  // Run Type
   UINT8_T margin_class_filter_c  // Margin Class Filter
   CHAR filler_1_s  // Filler
}
```

3.7.11.4 Usage and Conditions

Series

should be zero filled.

Run Type

should be filled in with one of the below alternatives:

- **End Of Day** - answer will hold latest available official data for given business date.
- **Intraday** - or given sequence number. Sequence number = 0 gives data for latest available intraday run.
- **Call** - for given sequence number. Sequence number = 0 gives data for latest available call run.

Sequence Number

If given, the parameters for a specific batch run is returned. If set to blank, the parameters for the latest available run are returned.

Margin Date

© The NASDAQ OMX Group, Inc. • 2013

should be filled in with today's business date, or a previous date. If set to blank, data for the latest available date is returned.

Clearinghouse

is not used and can be left blank.

Margin Class Filter

should be filled in with one of the below:

- **Specific** - for given **Margin Class**.
- **Relevant for me: Margin Class** - field is N/A. Data for all margin classes applicable for the Participant is returned, i.e. also margin classes used for any accounts under the participant.
- **All: Margin Class** - field is N/A. Data for all margin classes is returned.
- **Default** - is what will be used by the backwards compatible API, where it is not possible to specify any margin class parameters at all. **Margin Class** field is N/A. Data for the margin class applicable for the Participant is returned, but not margin classes specified for certain accounts under the participant.

Margin class

should be filled in if **Margin Class Filter** is set to Specific, otherwise blank.

FX margin parameters, Identity

can be filled with a wildcard name of the FX margin parameters. If set to blank, data for all FX margin parameters will be returned.

3.7.11.5 Answer Structure

The JA24 VIA has the following structure:

```
struct answer_rm_segment_hdr {
   struct transaction_type
   INT32_T sequence_number_n  // Sequence Number
   char[8] margin_date_s  // Margin Date
   char[6] margin_time_s  // Margin Time
   UINT8_T run_type_c  // Run Type
   CHAR filler_1_s  // Filler
   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 global_var_parameters  // Named struct no: 21077
         struct group_var_parameters  // Named struct no: 21078
      }
   }
}
```

3.7.11.6 Answer, Comments

The response will contain one item of the global_var_parameters struct that applies for all instruments using the Value at Risk (FX) margin model. There will in addition be one or more items of the group_var_parameters. For each instrument series using the Value at Risk (FX) margin model, the RQ3 response (and the **Margin Parameter** field) will contain a reference to the applicable FX margin parameters, Identity object.

Percentile for margin

A percentage expressed with two implicit decimals which is used to determine how many of the worst case scenarios that will be ignored when determining the scenario to use in the margin calculations. The number of price change scenarios used in the margin calculation is multiplied with this percentage and the result is truncated.

FX margin parameters, Identity

The FX margin parameter identity for the parameters returned in this item.

FX margin multiplier

A percentage expressed with two implicit decimals with which all price change scenarios will be scaled when determining the scenario price for the applicable instrument series.

3.7.12 JQ40 [Risk Cubes for Instrument QUERY]

3.7.12.1 Fingerprint

QUERY properties	
transaction type	JQ40
calling sequence	omniapi_query_ex
struct name	query_risk_cubes
facility	EP5
partitioned	false
answers	JA40

VIA properties	
transaction type	JA40
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

© The NASDAQ OMX Group, Inc. • 2013

3.7.12.2 Purpose

This query is used to retrieve the risk cubes applicable for an instrument.

3.7.12.3 Structure

The JQ40 QUERY has the following structure:

```
struct query_risk_cubes {
   struct transaction_type
   struct series  // Named struct no: 50000
   char[32] series_id_s  // Series, Identity
   char[8] business_date_s  // Date, Business
   UINT16_T segment_number_n  // Segment Number
   char[3] margin_class_s  // Margin class
   char[12] clh_id_s  // Clearinghouse
   UINT8_T run_type_c  // Run Type
   UINT8_T margin_class_filter_c  // Margin Class Filter
   CHAR filler_1_s  // Filler
}
```

3.7.12.4 Usage and Conditions

For each option series valued using Cash Flow Margin, there exists one Risk Cube per curve containing the margin requirement for 3 different volatilities of the underlying security and a variable number of calculation points. Other series than options, with the exception of repos and OTC IRS's, do also have risk cube values constructed in a similar way (for repos and ORC swaps, please use JQ41). The margin requirement for a portfolio can be calculated by using the Risk Cubes for each of the series using the same curve in the portfolio, plus the risk cubes for repo and swap trades. To summarize margin values for different curves, Curve Correlation Parameters must be used.

This query is only available when the signal BI7, Information type 8 (Evening data), type 41 (Preliminary data) has been sent.

Series

must be completed with **Country Number** and **Market Code**. The rest of the fields are independently optional, e.g. it is possible to filter for all instruments with a given underlying.

Series, Identity

can be filled in with wildcard series name.

Date, Business

should be filled in with today’s business date, or a previous date. If set to blank, data for the latest available date is returned.

Run Type

should be filled with **End Of Day** or **Preliminary**.

- **End Of Day**: answer will hold latest available official data for given business date.

© The NASDAQ OMX Group, Inc. • 2013

- **Preliminary**: answer will hold latest available preliminary data for given business date.

Margin Class Filter

should be filled in with one of the below:

- **Specific**: for given Margin Class.
- **Relevant for me**: Margin Class field is N/A. Data for all margin classes applicable for the Participant is returned, i.e. also margin classes used for any accounts under the participant.
- **All**: Margin Class field is N/A. Data for all margin classes is returned.
- **Default**: is what will be used by the backwards compatible API, where it is not possible to specify any margin class parameters at all. Margin Class field is N/A. Data for the margin class applicable for the Participant is returned, but not margin classes specified for certain accounts under the participant.

Margin Class

should be filled in if **Margin Class Filter** is set to Specific, otherwise blank.

Clearinghouse

is not used and can be left blank.

3.7.12.5 Answer Structure

The JA40 VIA has the following structure:

```
struct answer_risk_cubes {
   struct transaction_type
   UINT16_T segment_number_n  // Segment Number
   char[8] business_date_s  // Date, Business
   char[8] created_date_s  // Date, Created
   char[6] created_time_s  // Time, Created
   UINT16_T items_n  // Items
   UINT16_T size_n  // Size
}
Sequence {
   struct item_hdr
   Sequence {
      struct sub_item_hdr
      Choice {
         struct margin_class_curve  // Named struct no: 21012
         struct instrument_curve_node_values  // Named struct no: 21011
      }
   }
}
```

3.7.12.6 Answer, Comments

The answer contains a risk cube per Instrument. Each cube has a number of nodes, where each node is calculated using a yield curve (primary) stressed in a given way. Each node contains margin values for different volatilities and long or short position. There is also a discount factor to use when calculating the net present value of a forward payment cashflow. This part of the margin value cannot be included in the risk cube per instrument, since it is dependent on the trade price of a forward. The discount factor may be

© The NASDAQ OMX Group, Inc. • 2013

found using another curve (secondary) then the margin value. If forward payments are to be discounted using the same curve as was used to calculated margin values, primary and secondary curve have the same id. If no forward payment is relevant for the instrument, the discount factors are set to zero.

3.7.13 JQ41 [Risk Cubes for Trade QUERY]

3.7.13.1 Fingerprint

QUERY properties	
transaction type	JQ41
calling sequence	omniapi_query_ex
struct name	query_trade_risk_cubes
facility	EP5
partitioned	false
answers	JA41

VIA properties	
transaction type	JA41
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

3.7.13.2 Purpose

This query is used to retrieve the risk cubes applicable for trades. This is applicable for repo trades, and trades in OTC IRS's and TM FRA's.

This query is only available when the signal BI7, Information type 8 (Evening data), type 41 (Preliminary data) has been sent.

3.7.13.3 Structure

The JQ41 QUERY has the following structure:

```
struct query_trade_risk_cubes {
   struct transaction_type
   struct series  // Named struct no: 50000
   struct account
   INT64_T trade_number_q  // Trade number
   char[8] business_date_s  // Date, Business
   UINT16_T segment_number_n  // Segment Number
   UINT8_T run_type_c  // Run Type
   CHAR filler_1_s  // Filler
}
```

3.7.13.4 Usage and Conditions

This query is only available when the signal BI7, Information type 8 (Evening data), type 41 (Preliminary data) has been sent.

Series

must be completed with **Country Number** and **Market Code**.

Date, Business

should be filled in with today's business date or a previous date. If set to blank, data for the latest available date is returned.

A query can be done using three methods:

1. Searching for an explicit margin account. This can be achieved by filling in Country and Customer with explicit values and Account id with an explicit margin account id. The answer contains all trades which are included in the margin calculations for that margin account id.
2. Using Account string as wildcard search string. This can be achieved by filling in Country and Customer with explicit values and Account id = "*". The answer contains all trades for this Customer which are places on an account where Origin = client.
3. Using Trade Number as search criteria. The answer contains one specific trade.

Account

should be filled with the required customer (and possibly account id), or blank when requesting a specific trade.

Trade Number

should be filled with the trade for which results are requested, or zero for all trades at the account.

Run Type

should be filled with **EndOfDay** or **Preliminary**.

- **EndOfDay**: answer will hold latest available official data for given business date.
- **Preliminary**: answer will hold latest available preliminary data for given business date.

3.7.13.5 Answer Structure

The JA41 VIA has the following structure:

```
struct answer_trade_risk_cubes {
   struct transaction_type
   UINT16_T segment_number_n  // Segment Number
   UINT16_T items_n  // Items
   UINT16_T size_n  // Size
   char[2] filler_2_s  // Filler
}
```


© The NASDAQ OMX Group, Inc. • 2013

```
Sequence {
   struct item_hdr
   Sequence {
      struct sub_item_hdr
      Choice {
         struct margin_class_curve  // Named struct no: 21012
         struct trade_risk_values  // Named struct no: 21038
         struct trade_node_values  // Named struct no: 21010
      }
   }
}
```

3.7.13.6 Answer, Comments

The number of nodes in the risk cube is the number of scenarios that will be evaluated when stressing the yield curve, for example 5*5*5 will give 125 scenarios and 9*5*3 will give 135 scenarios.

The answer may hold the following structs, with one of the below shown sequence per trade:

- trade_risk_values (one per trade, showing trade number and account for the trade)
  - margin_class_curve (showing primary curve and curve correlation id for this trade and the following nodes). If primary and secondary curve are identical, the results are merged into one set of node values
    - trade_node_values (n node values in the risk cube for the primary curve and trade).
  - margin_class_curve (optional, showing secondary curve and its curve correlation id)
    - trade_node_values (optional, n node values in the risk cube for the secondary curve and trade).

Note:

When values for two curves are applicable their respective values must be summarized using Curve Correlation Cube parameters. To get the full margin value for a portfolio, these answers can be summarized with answers from JQ40.

3.7.14 JQ45 [Query Var Price Change Scenario QUERY]

3.7.14.1 Fingerprint

QUERY properties	
transaction type	JQ45
calling sequence	omniapi_query_ex
struct name	query_var_price_change_scenario
facility	EP5
partitioned	false
answers	JA45

VIA properties	
transaction type	JA45
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

3.7.14.2 Related Messages

RQ45, RQ46, CQ8

3.7.14.3 Purpose

This query is used to retrieve the price change scenarios that have been used for calculating margin requirements for instrument series have the Value at Risk (FX) margin model.

3.7.14.4 Structure

The JQ45 QUERY has the following structure:

```
struct query_var_price_change_scenario {
   struct transaction_type
   struct series  // Named struct no: 50000
   INT32_T sequence_number_n  // Sequence Number
   UINT16_T segment_number_n  // Segment Number
   char[8] margin_date_s  // Margin Date
   char[12] clh_id_s  // Clearinghouse
   char[3] margin_class_s  // Margin class
   UINT8_T run_type_c  // Run Type
   UINT8_T margin_class_filter_c  // Margin Class Filter
   CHAR filler_1_s  // Filler
}
```

3.7.14.5 Usage and Conditions

Series

should be zero filled.

Run Type

should be filled in with one of the below alternatives:

- **End Of Day** - answer will hold latest available official data for given business date.
- **Intraday** - or given sequence number. Sequence number = 0 gives data for latest available intraday run.
- **Call** - for given sequence number. Sequence number = 0 gives data for latest available call run.

Sequence Number

© The NASDAQ OMX Group, Inc. • 2013

If given, the parameters for a specific batch run is returned. If set to blank, the parameters for the latest available run are returned.

Margin Date

should be filled in with today's business date, or a previous date. If set to blank, data for the latest available date is returned.

Clearinghouse

is not used and can be left blank.

Margin Class Filter

should be filled in with one of the below:

- **Specific** - for given **Margin Class**.
- **Relevant for me: Margin Class** - field is N/A. Data for all margin classes applicable for the Participant is returned, i.e. also margin classes used for any accounts under the participant.
- **All: Margin Class** - field is N/A. Data for all margin classes is returned.
- **Default** - is what will be used by the backwards compatible API, where it is not possible to specify any margin class parameters at all. **Margin Class** field is N/A. Data for the margin class applicable for the Participant is returned, but not margin classes specified for certain accounts under the participant.

Margin class

should be filled in if **Margin Class Filter** is set to Specific, otherwise blank.

3.7.14.6 Answer Structure

The JA45 VIA has the following structure:

```
struct answer_rm_segment_hdr {
   struct transaction_type
   INT32_T sequence_number_n  // Sequence Number
   char[8] margin_date_s  // Margin Date
   char[6] margin_time_s  // Margin Time
   UINT8_T run_type_c  // Run Type
   CHAR filler_1_s  // Filler
   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 var_price_change_scenario  // Named struct no: 21076
        struct latest_trade_number_per_ins_type  // Named struct no: 21079
      }
   }
}
```

3.7.14.7 Answer, Comments

The response will first contain a "matrix" of items. On one axis we have combinations of base currency and price currency (i.e. currency pairs), and on the other axis we have price change scenarios. A price change scenario is thus a collection of price changes for a number of currency pairs, and the response will contain a number of such price change scenarios. A price change scenario is identified by the scenario number contained in each item.

Note that the currency pair for which price change scenarios are provided can be representing the opposite direction compared to the corresponding instrument series that is being cleared.

The query will return data for the clearinghouse to which the user is connected.

Price Change

The price change relative to the current (bid or ask) price that will be used in this scenario. The value will be expressed as a percentage with implicit decimals as specified below.

Currency, Base

The currency in which the nominal amount is expressed.

Currency, Price

The currency in which the price is expressed.

Decimals, Price

The number of implicit decimals used to express the percentage in the price change.

Scenario number

Identifies the price change scenario. I.e. all items with the same number belongs to the same price change scenario. The total number of price change scenarios will thus be the number of unique sequence numbers.

If the answer is for end of day calculations, some additional items are included, after the "matrix" is complete. These items show the last trade number included in the calculation, per instrument type.

3.7.15 JQ53 [TRADE SUM MARGIN QUERY]

3.7.15.1 Fingerprint

QUERY properties	
transaction type	JQ53
calling sequence	omniapi_query_ex
struct name	query_trade_sum_marg
facility	EP5
partitioned	false
answers	JA53

© The NASDAQ OMX Group, Inc. • 2013

VIA properties	
transaction type	JA53
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

3.7.15.2 Purpose

This query is used to retrieve summarized margin requirement per trade. The margin requirement in the instrument currency and in the risk currency is returned, and it is optional to convert the margin requirement to yet another currency.

This query is available when the signal JB1 has been sent (both intraday and end of day), or when the signal BI7, Information type 8 (Evening data) has been sent.

3.7.15.3 Structure

The JQ53 QUERY has the following structure:

```
struct query_trade_sum_marg {
   struct transaction_type
   struct series  // Named struct no: 50000
   struct account
   INT64_T trade_number_q  // Trade number
   INT32_T sequence_number_n  // Sequence Number
   UINT16_T segment_number_n  // Segment Number
   char[32] series_id_s  // Series, Identity
   char[12] clh_id_s  // Clearinghouse
   char[8] business_date_s  // Date, Business
   UINT8_T run_type_c  // Run Type
   CHAR filler_1_s  // Filler
}
```

3.7.15.4 Usage and Conditions

Series

should be filled with zeros.

Series Identity

can be filled in with wildcard series name

Date, Business

should be filled in with today’s business date or a previous date. If set to blank, data for the latest available date is returned.

Sequence Number

© The NASDAQ OMX Group, Inc. • 2013

is applicable for Run Type Intraday and Call only. If set to specific number, the data for a margin calculation run is returned. If set to zero, the data for the latest available run are returned.

Run Type

should be filled with **EndOfDay**, **Intraday** or **Call**

- **EndOfDay** (sequence number is N/A, answer will hold latest available data for given business date)
- **Intraday** (for given sequence number, sequence number = 0 gives data for latest available intraday run)
- **Call** (Sequence number is required).

Account

should all be filled in with values in one of the following ways:

- Fill in the field with explicit value. All answers must match the field.
- Fill in the field with "*". No test is made on the value for that field.
- Fill in the field with a string ended by "*". All answers must in this field start with the string specified.

Trade Number

should be filled in with values in one of the following ways:

- Fill in the field with explicit value. Answer will contain one trade only. If no values exist for the given trade, answer is empty.
- Fill in the field with "*". No test is made on the value for that field.

Clearinghouse

is not used and can be left blank.

3.7.15.5 Answer Structure

The JA53 VIA has the following structure:

```
struct answer_trade_sum_marg {
   struct transaction_type
   INT32_T sequence_number_n  // Sequence Number
   UINT16_T segment_number_n  // Segment Number
   char[8] business_date_s  // Date, Business
   UINT8_T run_type_c  // Run Type
   CHAR filler_1_s  // Filler
   UINT16_T items_n  // Items
   UINT16_T size_n  // Size
}
Sequence {
   struct item_hdr
   Sequence {
      struct sub_item_hdr
      Choice {
         struct trade_sum_marg  // Named struct no: 21041
      }
```


© The NASDAQ OMX Group, Inc. • 2013

```
        }
    }
```

3.7.15.6 Answer, Comments

All structures are returned per **Trade** and **Instrument Currency**. The first structure for a trade and instrument currency is always **TRADE_SUM_MARG**.

Power Delta hedge has no handling of OTC trades, and the query is therefore not applicable to NordPool.

3.7.16 JQ54 [Margins on Margin Requirement Account QUERY]

3.7.16.1 Fingerprint

QUERY properties	
transaction type	JQ54
calling sequence	omniapi_query_ex
struct name	query_margin_requirement_account
facility	EP5
partitioned	false
answers	JA54

VIA properties	
transaction type	JA54
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

3.7.16.2 Related Messages

JQ55

3.7.16.3 Purpose

This query is used to retrieve summarized margin requirements per Margin Requirement Account. The margin requirement can be expressed in the instrument currency or the risk currency, or both of these.

This query is available when the signal JB1 has been sent (both intraday and end of day), or when the signal BI7, Information type 8 (Evening data) has been sent.

3.7.16.4 Structure

The JQ54 QUERY has the following structure:

© The NASDAQ OMX Group, Inc. • 2013

```
struct query_margin_requirement_account {
   struct transaction_type
   struct series  // Named struct no: 50000
   char[12] clh_id_s  // Clearinghouse
   struct mra_account  // Of type: ACCOUNT
   INT32_T sequence_number_n  // Sequence Number
   UINT16_T segment_number_n  // Segment Number
   char[8] business_date_s  // Date, Business
   UINT8_T run_type_c  // Run Type
   UINT8_T instrument_or_risk_currency_c  // Instrument or risk currency.
}
```

3.7.16.5 Usage and Conditions

Series

should be zero filled.

Margin Requirement Account

should all (country, customer, account) be filled in with values in one of the following ways:

1. Fill in the field with explicit value. All answers must match the 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.

Date, Business

should be filled in with today's business date or a previous date. If set to blank, data for the latest available date is returned.

Run Type

should be filled with **EndOfDay**, **Intraday** or **Call**.

- **EndOfDay** (sequence number is N/A, answer will hold latest available data for given business date).
- **Intraday** (for given sequence number. Sequence number = 0 gives data for latest available intraday run).
- **Call** (Sequence number is required).

Sequence Number

is applicable for Run Type Intraday and Call only. If set to specific number, the data for a margin calculation run is returned. If set to zero, the data for the latest available run is returned. Sequence Numbers are retrieved by query JQ1.

Instrument Currency or Risk Currency

should be filled in with Risk Currency, Instrument Currency or both.

Clearinghouse

© The NASDAQ OMX Group, Inc. • 2013

is not used and can be left blank.

3.7.16.6 Answer Structure

The JA54 VIA has the following structure:

```
struct answer_margin_requirement_account {
   struct transaction_type
   char[8] business_date_s  // Date, Business
   char[6] margin_time_s  // Margin Time
   char[2] filler_2_s  // Filler
   INT32_T sequence_number_n  // Sequence Number
   UINT16_T segment_number_n  // Segment Number
   UINT16_T items_n  // Items
   UINT16_T size_n  // Size
   UINT8_T run_type_c  // Run Type
   CHAR filler_1_s  // Filler
}
Sequence {
   struct item_hdr
   Sequence {
      struct sub_item_hdr
      Choice {
         struct mra_account_vim  // Named struct no: 50007
         struct margin_result_components  // Named struct no: 21062
         struct margin_result_components_pdh  // Named struct no: 21065
         struct margin_result_components_cfm  // Named struct no: 21066
         struct margin_result_overdue  // Named struct no: 21063
         struct margin_result_base_api  // Named struct no: 21064
      }
   }
}
```

3.7.16.7 Answer, Comments

Answer is returned using the VIM concept. One item, with at least three sub items, per margin requirement account and currency (instrument and/or risk), is returned. The item has a number of sub items;

The sub item account itself, **MRA_ACCOUNT_VIM** (vim 50007) is always returned, first in each item.

It is then always followed by the sub item **MARGIN_RESULT_COMPONENTS** (vim 21062).

If there are margin components originating from positions using CFM model or Power Delta Hedge model, separate VIM sub items are included to show information about these figures, **MARGIN_RESULT_CFM** (vim 21066) and **MARGIN_RESULT_PDH** (vim 21065).

If there are any payment or delivery margins for the settlement date or earlier, a VIM sub item for "potentially" overdue payments and deliveries is included, **MARGIN_RESULT_OVERDUE** (vim 21063). If all payments and deliveries are met before next due time, no margin will actually be required for these items.

At the end of each item, there is a sub item showing values where margin components have been summarized into figures for Initial Margin, Variation Margin, Contingent Variation Margin and Total Margin, **MARGIN_RESULT_BASE_API** (vim 21064). This sub item is always returned for an item.

© The NASDAQ OMX Group, Inc. • 2013

3.7.17 JQ55 [Margins on Margin Requirement Account, per calculation Account QUERY]

3.7.17.1 Fingerprint

QUERY properties	
transaction type	JQ55
calling sequence	omniapi_query_ex
struct name	query_margin_requirement_account
facility	EP5
partitioned	false
answers	JA55

VIA properties	
transaction type	JA55
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

3.7.17.2 Related Messages

JQ54

3.7.17.3 Purpose

This query is used to retrieve margin requirements per Margin Requirement Account, but split up per the Margin Calculation Accounts propagated into the Margin Requirement Account. The margin requirements can be expressed in the instrument currency or the risk currency, or both of these.

This query is available when the signal JB1 has been sent (both intraday and end of day), or when the signal BI7, Information type 8 (Evening data) has been sent.

3.7.17.4 Structure

The JQ55 QUERY has the following structure:

```
struct query_margin_requirement_account {
   struct transaction_type
   struct series  // Named struct no: 50000
   char[12] clh_id_s  // Clearinghouse
   struct mra_account  // Of type: ACCOUNT
   INT32_T sequence_number_n  // Sequence Number
   UINT16_T segment_number_n  // Segment Number
   char[8] business_date_s  // Date, Business
```


© The NASDAQ OMX Group, Inc. • 2013

```
    UINT8_T run_type_c  // Run Type
    UINT8_T instrument_or_risk_currency_c  // Instrument or risk currency.
}
```

3.7.17.5 Usage and Conditions

Series

should be zero filled.

Margin Requirement Account

should all (country, customer, account) be filled in with values in one of the following ways:

1. Fill in the field with explicit value. All answers must match the 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.

Date, Business

should be filled in with today's business date or a previous date. If set to blank, data for the latest available date is returned.

Run Type

should be filled with **EndOfDay**, **Intraday** or **Call**.

- **EndOfDay** (sequence number is N/A, answer will hold latest available data for given business date).
- **Intraday** (for given sequence number. Sequence number = 0 gives data for latest available intraday run).
- **Call** (Sequence number is required).

Sequence Number

is applicable for Run Type Intraday and Call only. If set to specific number, the data for a margin calculation run is returned. If set to zero, the data for the latest available run is returned. Sequence Numbers are retrieved by query JQ1.

Instrument Currency or Risk Currency

should be filled in with Risk Currency, Instrument Currency or both.

Clearinghouse

is not used and can be left blank.

3.7.17.6 Answer Structure

The JA55 VIA has the following structure:

```
struct answer_margin_requirement_account {
   struct transaction_type
```

```
      char[8] business_date_s  // Date, Business
      char[6] margin_time_s  // Margin Time
      char[2] filler_2_s  // Filler
      INT32_T sequence_number_n  // Sequence Number
      UINT16_T segment_number_n  // Segment Number
      UINT16_T items_n  // Items
      UINT16_T size_n  // Size
      UINT8_T run_type_c  // Run Type
      CHAR filler_1_s  // Filler
   }
   Sequence {
      struct item_hdr
      Sequence {
         struct sub_item_hdr
         Choice {
            struct mra_account_vim  // Named struct no: 50007
            struct account_vim  // Named struct no: 50005
            struct margin_result_components  // Named struct no: 21062
            struct margin_result_components_pdh  // Named struct no: 21065
            struct margin_result_components_cfm  // Named struct no: 21066
            struct margin_result_overdue  // Named struct no: 21063
            struct margin_result_base_api  // Named struct no: 21064
         }
      }
   }
```

3.7.17.7 Answer, Comments

Answer is returned using the VIM concept. One item per margin requirement account, margin calculation account and currency (risk or/and instrument currency) is returned. Each item will at least include four sub items:

The sub item **MRA_ACCOUNT_VIM** (vim 50007), always comes first in each item, followed by a sub item for the margin calculation account itself, **ACCOUNT_VIM** (vim 50005). These are then always followed by the sub item MARGIN_RESULT_COMPONENTS (vim 21062).

If there are margin components originating from positions using CFM model or Power Delta Hedge model, separate VIM sub items are included to show information about these figures, **MARGIN_RESULT_CFM** (vim 21066) and **MARGIN_RESULT_PDH** (vim 21065).

If there are any payment or delivery margins for the settlement date or earlier, a VIM sub item for "potentially" overdue payments and deliveries is included, **MARGIN_RESULT_OVERDUE** (vim 21063). If all payments and deliveries are met before next due time, no margin will actually be required for these items.

At the end of each item, there is a sub item showing values where margin components have been summarized into figures for Initial Margin, Variation Margin, Contingent Variation Margin and Total Margin, **MARGIN_RESULT_BASE_API** (vim 21064). This sub item is always returned for an item.

3.7.18 JQ56 [Margins on Margin Aggregation Group QUERY]

3.7.18.1 Fingerprint

QUERY properties	
transaction type	JQ56

© The NASDAQ OMX Group, Inc. • 2013

QUERY properties	
calling sequence	omniapi_query_ex
struct name	query_margin_aggregation_group
facility	EP5
partitioned	false
answers	JA56

VIA properties	
transaction type	JA56
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

3.7.18.2 Related Messages

JQ57

3.7.18.3 Purpose

This query is used to retrieve summarized margin requirements per Margin Aggregation Group. The margin requirement can be expressed in the instrument currency or the risk currency, or both of these.

This query is available when the signal JB1 has been sent (both intraday and end of day), or when the signal BI7, Information type 8 (Evening data) has been sent.

3.7.18.4 Structure

The JQ56 QUERY has the following structure:

```
struct query_margin_aggregation_group {
   struct transaction_type
   struct series  // Named struct no: 50000
   char[12] clh_id_s  // Clearinghouse
   struct margin_aggregation_group  // Of type: ACCOUNT
   INT32_T sequence_number_n  // Sequence Number
   UINT16_T segment_number_n  // Segment Number
   char[8] business_date_s  // Date, Business
   UINT8_T run_type_c  // Run Type
   UINT8_T instrument_or_risk_currency_c  // Instrument or risk currency.
}
```

3.7.18.5 Usage and Conditions

Series

should be zero filled.

Margin Aggregation Group

should all (country, customer, account) be filled in with values in one of the following ways:

1. Fill in the field with explicit value. All answers must match the 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.

Date, Business

should be filled in with today's business date or a previous date. If set to blank, data for the latest available date is returned.

Run Type

should be filled with **EndOfDay**, **Intraday** or **Call**.

- **EndOfDay** (sequence number is N/A, answer will hold latest available data for given business date).
- **Intraday** (for given sequence number. Sequence number = 0 gives data for latest available intraday run).
- **Call** (Sequence number is required).

Sequence Number

is applicable for Run Type Intraday and Call only. If set to specific number, the data for a margin calculation run is returned. If set to zero, the data for the latest available run is returned. Sequence Numbers are retrieved by query JQ1.

Instrument or Risk Currency

should be filled in with Risk Currency, Instrument Currency or both.

Clearinghouse

is not used and can be left blank.

3.7.18.6 Answer Structure

The JA56 VIA has the following structure:

```
struct answer_margin_aggregation_group {
   struct transaction_type
   char[8] business_date_s  // Date, Business
   char[6] margin_time_s  // Margin Time
   char[2] filler_2_s  // Filler
   INT32_T sequence_number_n  // Sequence Number
   UINT16_T segment_number_n  // Segment Number
   UINT16_T items_n  // Items
   UINT16_T size_n  // Size
   UINT8_T run_type_c  // Run Type
   CHAR filler_1_s  // Filler
}
```


© The NASDAQ OMX Group, Inc. • 2013

```
Sequence {
   struct item_hdr
   Sequence {
      struct sub_item_hdr
      Choice {
         struct margin_aggregation_group_vim  // Named struct no: 50006
         struct margin_result_components  // Named struct no: 21062
         struct margin_result_components_pdh  // Named struct no: 21065
         struct margin_result_components_cfm  // Named struct no: 21066
         struct margin_result_overdue  // Named struct no: 21063
         struct margin_result_base_api  // Named struct no: 21064
      }
   }
}
```

3.7.18.7 Answer, Comments

Answer is returned using the VIM concept. One item, with at least three sub items, per margin aggregation group and currency (instrument and/or risk), is returned. The item has a number of sub items;

The VIM sub item for the margin aggregation group itself, **MARGIN_AGGREGATION_GROUP_VIM** (vim 50006) is always returned, first in each item.

It is then always followed by the sub item **MARGIN_RESULT_COMPONENTS** (vim 21062).

If there are margin components originating from positions using CFM model or Power Delta Hedge model, separate VIM sub items are included to show information about these figures, **MARGIN_RESULT_CFM** (vim 21066) and **MARGIN_RESULT_PDH** (vim 21065).

If there are any payment or delivery margins for the settlement date or earlier, a VIM sub item for "potentially" overdue payments and deliveries is included, **MARGIN_RESULT_OVERDUE** (vim 21063). If all payments and deliveries are met before next due time, no margin will actually be required for these items.

At the end of each item, there is a sub item showing values where margin components have been summarized into figures for Initial Margin, Variation Margin, Contingent Variation Margin and Total Margin, **MARGIN_RESULT_BASE_API** (vim 21064). This sub item is always returned for an item.

3.7.19 JQ57 [Margins on Margin Aggregation Group, per account QUERY]

3.7.19.1 Fingerprint

QUERY properties	
transaction type	JQ57
calling sequence	omniapi_query_ex
struct name	query_margin_aggregation_group
facility	EP5
partitioned	false
answers	JA57

VIA properties	
transaction type	JA57
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

3.7.19.2 Related Messages

JQ56

3.7.19.3 Purpose

This query is used to retrieve margin requirements per Margin Aggregation Group, split up on the accounts propagated into the Margin Aggregation Group. The margin requirements can be expressed in the instrument currency or the risk currency, or both of these

This query is available when the signal JB1 has been sent (both intraday and end of day), or when the signal BI7, Information type 8 (Evening data) has been sent.

3.7.19.4 Structure

The JQ57 QUERY has the following structure:

```
struct query_margin_aggregation_group {
   struct transaction_type
   struct series  // Named struct no: 50000
   char[12] clh_id_s  // Clearinghouse
   struct margin_aggregation_group  // Of type: ACCOUNT
   INT32_T sequence_number_n  // Sequence Number
   UINT16_T segment_number_n  // Segment Number
   char[8] business_date_s  // Date, Business
   UINT8_T run_type_c  // Run Type
   UINT8_T instrument_or_risk_currency_c  // Instrument or risk currency.
}
```

3.7.19.5 Usage and Conditions

Series

should be zero filled.

Margin Aggregation Group

should all (country, customer, account) be filled in with values in one of the following ways:

1. Fill in the field with explicit value. All answers must match the field.
2. Fill in the field with "*". No test is made on the value for that field.

© The NASDAQ OMX Group, Inc. • 2013

3. Fill in the field with a string ended by "*". All answers must in this field start with the string specified.

Date, Business

should be filled in with today's business date or a previous date. If set to blank, data for the latest available date is returned.

Run Type

should be filled with **EndOfDay**, **Intraday** or **Call**.

- **EndOfDay** (sequence number is N/A, answer will hold latest available data for given business date).
- **Intraday** (for given sequence number. Sequence number = 0 gives data for latest available intraday run).
- **Call** (Sequence number is required).

Sequence Number

is applicable for Run Type Intraday and Call only. If set to specific number, the data for a margin calculation run is returned. If set to zero, the data for the latest available run is returned. Sequence Numbers are retrieved by query JQ1.

Instrument or Risk Currency

should be filled in with Risk Currency, Instrument Currency or both.

Clearinghouse

is not used and can be left blank.

3.7.19.6 Answer Structure

The JA57 VIA has the following structure:

```
struct answer_margin_aggregation_group {
   struct transaction_type
   char[8] business_date_s  // Date, Business
   char[6] margin_time_s  // Margin Time
   char[2] filler_2_s  // Filler
   INT32_T sequence_number_n  // Sequence Number
   UINT16_T segment_number_n  // Segment Number
   UINT16_T items_n  // Items
   UINT16_T size_n  // Size
   UINT8_T run_type_c  // Run Type
   CHAR filler_1_s  // Filler
}
Sequence {
   struct item_hdr
   Sequence {
      struct sub_item_hdr
      Choice {
         struct margin_aggregation_group_vim  // Named struct no: 50006
         struct account_vim  // Named struct no: 50005
         struct margin_aggregation_info  // Named struct no: 21067
         struct margin_result_components  // Named struct no: 21062
```

```
            struct margin_result_components_pdh  // Named struct no: 21065
            struct margin_result_components_cfm  // Named struct no: 21066
            struct margin_result_overdue  // Named struct no: 21063
            struct margin_result_base_api  // Named struct no: 21064
        }
    }
}
```

3.7.19.7 Answer, Comments

When a Margin Aggregation Group has got positions propagated into it, i.e. there is a "super-position" for the Margin Aggregation Group itself, results for this positions are shown as if the group was an account.

Answer is returned using the VIM concept, where a sequence of items build up the result for one Margin Aggregation Group.

The first item (and sub item) in such a sequence holds the Margin Aggregation Group itself, returned once per sequence, **MARGIN_AGGREGATION_GROUP_VIM** (vim 50006). It is then followed by a number of items, one item per aggregated account and currency (risk or/and instrument currency). For such an item at least three sub items are included:

The sub item account itself, **ACCOUNT_VIM** (vim 50005) is always returned, first in each item.

It is then always followed by the sub item **MARGIN_RESULT_COMPONENTS** (vim 21062).

If there are margin components originating from positions using CFM model or Power Delta Hedge model, separate VIM sub items are included to show information about these figures, **MARGIN_RESULT_CFM** (vim 21066) and **MARGIN_RESULT_PDH** (vim 21065).

If there are any payment or delivery margins for the settlement date or earlier, a VIM sub item for "potentially" overdue payments and deliveries is included, **MARGIN_RESULT_OVERDUE** (vim 21063). If all payments and deliveries are met before next due time, no margin will actually be required for these items.

At the end of each item, there is a sub item showing values where margin components have been summarized into figures for Initial Margin, Variation Margin, Contingent Variation Margin and Total Margin, **MARGIN_RESULT_BASE_API** (vim 21064). This sub item is always returned for an item.

3.7.20 JQ58 [SuperPosition on Margin Aggregation Group, propagated or non-propagated QUERY]

3.7.20.1 Fingerprint

QUERY properties	
transaction type	JQ58
calling sequence	omniapi_query_ex
struct name	query_margin_aggregation_group_position
facility	EP5
partitioned	false
answers	JA58

© The NASDAQ OMX Group, Inc. • 2013

VIA properties	
transaction type	JA58
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

3.7.20.2 Related Messages

JQ56, JQ57

3.7.20.3 Purpose

This query is used to retrieve information about accounts that have been aggregated into a "super position" in a Margin Aggregation Group. Margin requirements may either be returned for the propagated super-position or for the non-propagated margin for the positions building up the super-position.

This query is available when the signal JB1 has been sent (both intraday and end of day), or when the signal BI7, Information type 8 (Evening data) has been sent.

3.7.20.4 Structure

The JQ58 QUERY has the following structure:

```
struct query_margin_aggregation_group_position {
   struct transaction_type
   struct series  // Named struct no: 50000
   char[12] clh_id_s  // Clearinghouse
   struct margin_aggregation_group  // Of type: ACCOUNT
   INT32_T sequence_number_n  // Sequence Number
   UINT16_T segment_number_n  // Segment Number
   char[8] business_date_s  // Date, Business
   UINT8_T run_type_c  // Run Type
   UINT8_T propagated_margin_position_c  // PROPAGATED_MARGIN_POSITION_C
}
```

3.7.20.5 Usage and Conditions

Series

should be zero filled.

Margin Aggregation Group

should all (country, customer, account) be filled in with values in one of the following ways:

- Fill in the field with explicit value. All answers must match the field.

Date, Business

should be filled in with business date from the answer to JQ56 or 57.

Run Type

should be filled in with **Run Type** from the answer to JQ56 or 57.

Sequence Number

should be filled in with **Sequence Number** from the answer to JQ56 or 57.

Propagated Margin Position

If set to True, the propagated super-position, and the margin requirements calculated for it is returned. If set to False, the non-propagated positions that built the super-position, and non-propagated margin figures is returned.

Clearinghouse

is not used and can be left blank.

3.7.20.6 Answer Structure

The JA58 VIA has the following structure:

```
struct answer_margin_aggregation_group_position {
   struct transaction_type
   char[8] business_date_s  // Date, Business
   char[6] margin_time_s  // Margin Time
   char[2] filler_2_s  // Filler
   INT32_T sequence_number_n  // Sequence Number
   UINT16_T segment_number_n  // Segment Number
   UINT16_T items_n  // Items
   UINT16_T size_n  // Size
   UINT8_T run_type_c  // Run Type
   UINT8_T propagated_margin_position_c  // PROPAGATED_MARGIN_POSITION_C
}
Sequence {
   struct item_hdr
   Sequence {
      struct sub_item_hdr
      Choice {
         struct margin_aggregation_group_vim  // Named struct no: 50006
         struct account_vim  // Named struct no: 50005
         struct margin_position_info  // Named struct no: 21068
         struct margin_aggregation_info  // Named struct no: 21067
         struct margin_result_base_api  // Named struct no: 21064
         struct margin_result_payment_margin  // Named struct no: 21069
      }
   }
}
```

3.7.20.7 Answer, Comments

Answer is returned using the VIM concept, where a sequence of items are returned for each Margin Aggregation Group.

© The NASDAQ OMX Group, Inc. • 2013

The first item (and sub item) in such a sequence holds the Margin Aggregation Group itself, returned once per sequence, **MARGIN_AGGREGATION_GROUP_VIM** (VIM 50006). It is then followed by a number of items, one item per aggregated account. For such an item, a number of sub items are included:

The sub item account itself, **ACCOUNT_VIM** (VIM 50005) is always returned, first in each item.

If there is a position to be margined for an aggregated account, sub item **MARGIN_POSITION_INFO** (vim 21068) is returned. If propagated positions are returned, this struct holds the Margin Aggregation Group itself.

Sub item **MARGIN_AGGREGATION_INFO** (VIM 21067) is only returned for non-propagated positions, and hold information on how the aggregated account was included in the super position.

Sub item **MARGIN_RESULT_BASE_API** (VIM 21064) shows figures for Initial Margin, Variation Margin, Contingent Variation Margin and Total Margin.

If there is payment margin for an aggregated account, or for the propagated position, sub item **MARGIN_RESULT_PAYMENT_MARGIN** (VIM 21069) is returned.

3.7.21 JQ71 [Query RM margin simulation VIQ]

3.7.21.1 Fingerprint

VIQ properties	
transaction type	JQ71
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	EP4
partitioned	false
answers	JA71

VIA properties	
transaction type	JA71
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

3.7.21.2 Related Messages

CQ41

3.7.21.3 Purpose

This query is used for simulating margin requirements.

© The NASDAQ OMX Group, Inc. • 2013

3.7.21.4 Structure

The JQ71 VIQ has the following structure:

```
struct query_rm_margin_sim {
   struct transaction_type
   struct series  // Named struct no: 50000
   UINT16_T segment_number_n  // Segment Number
   UINT16_T qry_segment_number_n  // Segment Number, Query
   UINT8_T last_qry_segment_c  // Last, Query Segment
   char[3] filler_3_s  // Filler
   UINT16_T items_n  // Items
   UINT16_T size_n  // Size
}
Sequence {
   struct item_hdr
   Sequence {
      struct sub_item_hdr
      Choice {
         struct rm_margin_simulation  // Named struct no: 21044
         struct rm_margin_sim_markets  // Named struct no: 21045
         struct rm_margin_sim_trades  // Named struct no: 21046
         struct rm_margin_sim_trades_account  // Named struct no: 21072
         struct otc_cash_flow_base  // Named struct no: 65
         struct otc_cash_flow_info  // Named struct no: 66
      }
   }
}
```

3.7.21.5 Usage and Conditions

It is possible to calculate indicative margin requirements for a specific account with current prices and positions plus a list of supplied trades. The trades can be normal trades where the contracts specifies most of the details, or OTC trades where more details about the contract must be supplied for the trade itself. It is also possible not to use any existing position, but to supply all trades used in the query.

They Query is a segmented Variable Input Query, which means that a choice of possible structs can be sent in, in one or more segments, if data cannot be fit into one segment.

Query must contain the head record QUERY_RM_MARGIN_SIM, and in the first segment one record RM_MARGIN_SIMULATION must come first. 0-n records RM_MARGIN_SIM_MARKET could follow, and there after 0-n records RM_MARGIN_SIM_TRADES. For local IRS's and TM FRA trades, 2-n number of OTC_CASH_FLOWS are included.

Segment Number In is set to zero, if only one segment is used. If more than one segment is used, Segment Number In is set to 1..n, and to 0 for the last segment.

3.7.21.6 Structure Contents

RM_MARGIN_SIMULATION

Usage of fields in this structure:

© The NASDAQ OMX Group, Inc. • 2013

Series should be zero filled.

Account may be filled with a specific account or may be left blank . This is the account for which the margin simulation will be made. Added trades are considered to be placed on this account, unless position simulation indicates that this is a margin requirement account, in which case it is possible to specify one of the margin calculation account (named struct RM_MARGIN_SIM_TRADES_ACCOUNT) to place the trades on.

Position Simulation If set to 6 or 7, i.e. result is on margin requirement account level, the specified account should be a margin requirement account. Positions included will be positions for all margin calculation accounts that propagates margin to the margin requirement account.

Date must be set to current business date.

Sub User should be set to the name of a sub user, or blank in case it is not applicable.

Margin Class should be filled in with the margin class to use in the simulation, or blank in case the configured margin class applicable for the participant or account should be used.

RM_MARGIN_SIM_MARKET

This record specifies which markets (for included positions) that should be included in the simulations. If no record of this type is included in the query, positions from all markets are included. Usage of fields in this structure:

Series should be filled with *Country Number* and *Market Code*.

RM_MARGIN_SIM_TRADES_ACCOUNT

This record specifes the account to place the trades specified in **RM_MARGIN_SIM_TRADES**. If no account specified, trades will be placed on the account specified in **RM_MARGIN_SIMULATION**.

RM_MARGIN_SIM_TRADES

Usage of fields in this structure:

Item type, Simulation Query specifies what type of input this item contains. It can take the following values:

Value	Type
2	Bought trade
3	Sold trade
4	Payment
5	Bought Delivery
6	Sold Delivery

- Items with item type 2 or 3:
  - The *Series* field should contain the series used

- The *Quantity, Simulation* field contains the desired quantity. 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 this 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 desired quantity. 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.
- 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.
- If *Positions Simulated* = 2, the only items allowed are those with Item type = 1 (i.e. 2-6 are not allowed).
- If *Prices Simulated* equals 1, the supplied values in the fields *Added Trades Simulated*, *Series Expiring Today Simulated* and *Futures Profit/loss Simulated* will be ignored.

3.7.21.7 Return Codes

The error handling in this query is as follows:

Cstatus	Txstat
Successful	RI_OMN_NORMAL – Successful Completion

© The NASDAQ OMX Group, Inc. • 2013

Cstatus	Txstat
Successful	Other than RI_OMN_NORMAL – Calculations failed

Please refer to the *System Error Messages Reference* 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.

3.7.21.8 Answer Structure

The JA71 VIA has the following structure:

```
    struct answer_rm_margin_sim {
       struct transaction_type
       UINT16_T segment_number_n  // Segment Number
       UINT16_T segment_number_out_n  // Segment Number ; Of type:
SEGMENT_NUMBER_N
       UINT16_T items_n  // Items
       UINT16_T size_n  // Size
    }
    Sequence {
       struct item_hdr
       Sequence {
          struct sub_item_hdr
          Choice {
             struct rm_margin_sim_failure_reason  // Named struct no: 21050
             struct rm_margin_sim_sum  // Named struct no: 21052
             struct rm_margin_sim_pos  // Named struct no: 21051
             struct rm_margin_sim_del  // Named struct no: 21053
             struct rm_margin_sim_sum_pos_ulg  // Named struct no: 21054
             struct rm_margin_sim_pay  // Named struct no: 21055
             struct rm_margin_sim_sum_pay_ulg  // Named struct no: 21056
             struct rm_margin_sim_prices  // Named struct no: 21047
             struct rm_margin_sim_oms2_ivl  // Named struct no: 21048
             struct rm_margin_sim_vola  // Named struct no: 21049
          }
       }
    }
```

3.7.21.9 Answer, Comments

The response received is a list of records according to the following:

First comes one:

- **Margin Requirement Simulation Summary (RM_MARGIN_SIM_SUM):**

  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.

If output level is >= 2, a sequence of the below records is also returned. The records could be included or not, depending on position:

- 1-n
  - **Margin Requirement Simulation Position (RM_MARGIN_SIM_POS):**

© The NASDAQ OMX Group, Inc. • 2013

Contains individual margin requirements for a single open position.

- **Margin Requirement Simulation Delivery records(RM_MARGIN_SIM_DEL):**

  Contains individual margin requirements for a single delivery position.

- 1

  - **Margin Requirement Simulation Sum Position per Underlying (RM_MARGIN_SIM_SUM_POS_ULG):**

    Contains the summary of margin requirements for open and delivery positions for an underlying. Note that in the record, the series contains only underlying data.

- 1-n

  - **Margin Requirement Simulation Payment (RM_MARGIN_SIM_PAY):**

    Contains individual margin requirement for a single payment position.

- 1

  - **Margin Requirement Simulation Sum Payment per Underlying (RM_MARGIN_SIM_SUM_PAY_ULG):**

    Contains summary margin requirement of payment positions for an underlying.

If output level is = 3, a sequence of the below records is also returned. The records could be included or not, depending on position:

- 1-n

  - **Margin Requirement Simulation Prices (RM_MARGIN_SIM_PRICES):**

    Contains prices used in the calculations.

  - **Margin Requirement Simulation OMS2_Intervals (RM_MARGIN_SIM_OMS2_IVL):**

    Contains valuation intervals used in OMS2 calculations. Only included if OMS2 was used.

  - **Margin Requirement Volatilities (RM_MARGIN_SIM_VOL):**

    Contains volatilities used in option calculations. Only included for options.

3.7.22 RC60 [Private price list TRANSACTION]

3.7.22.1 Fingerprint

TRANSACTION properties	
transaction type	RC60
calling sequence	omniapi_tx_ex
struct name	modify_private_price_list
facility	EP4
partitioned	false

© The NASDAQ OMX Group, Inc. • 2013

3.7.22.2 Related Messages

RQ60

3.7.22.3 Purpose

This transaction is used for initializing data in the private price list for the user who is sending the transaction. It is used for margin simulations in Genium INET Clearing.

3.7.22.4 Structure

The RC60 TRANSACTION has the following structure:

```
struct modify_private_price_list {
   struct transaction_type
   struct series  // Named struct no: 50000
   UINT16_T is_apply_spread_rule_n  // Apply spread rule
   char[32] sub_user_s  // Sub User
   UINT8_T private_price_list_cmd_c  // Private price list command
   CHAR filler_1_s  // Filler
}
```

3.7.22.5 Usage and conditions

Series

should be zeroed.

Sub User

should be set to blank, except when used from Genium INET Clearing Back Office Server.

Private price list command

specifies how the price list should be initialized.

3.7.23 RC65 [Private margin underlying prices TRANSACTION]

3.7.23.1 Fingerprint

TRANSACTION properties	
transaction type	RC65
calling sequence	omniapi_tx_ex
struct name	modify_margin_ulg_price_private
facility	EP4
partitioned	false

© The NASDAQ OMX Group, Inc. • 2013

3.7.23.2 Related Messages

RC60, RQ60, RQ65, RC66, RQ66, RQ71

3.7.23.3 Purpose

This transaction is used for setting underlying prices contained in a private price list. It is used for margin simulations in Genium INET Clearing.

3.7.23.4 Structure

The RC65 TRANSACTION has the following structure:

```
struct modify_margin_ulg_price_private {
   struct transaction_type
   struct series  // Named struct no: 50000
   UINT16_T items_n  // Items
   char[2] filler_2_s  // Filler
   char[32] sub_user_s  // Sub User
   Array ITEM [max no: 500] {
      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
      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
      UINT8_T marg_theo_c  // Margin, Settlement Price Theoretical Mark
      char[2] filler_2_s  // Filler
   }
}
```

3.7.23.5 Usage and conditions

Series

should be zeroed.

Sub User

should be set to blank, except when used from Genium INET Clearing Back Office Server.

3.7.24 RC66 [Private margin prices and volatilities TRANSACTION]

3.7.24.1 Fingerprint

TRANSACTION properties	
transaction type	RC66

© The NASDAQ OMX Group, Inc. • 2013

TRANSACTION properties	
calling sequence	omniapi_tx_ex
struct name	modify_margin_series_price_private
facility	EP4
partitioned	false

3.7.24.2 Related Messages

RC60, RQ60, RC65, RQ65, RQ66, RQ71

3.7.24.3 Purpose

This transaction is used for setting series prices and volatilities contained in a private price list. It is used for margin simulations in Genium INET Clearing.

3.7.24.4 Structure

The RC66 TRANSACTION has the following structure:

```
struct modify_margin_series_price_private {
   struct transaction_type
   struct series  // Named struct no: 50000
   UINT16_T items_n  // Items
   char[32] sub_user_s  // Sub User
   char[2] filler_2_s  // Filler
   Array ITEM [max no: 1000] {
      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 bid_marg_vol_i  // Margin, Volatility Bid
      INT32_T ask_marg_vol_i  // Margin, Volatility Ask
      INT32_T mid_marg_vol_i  // Margin, Volatility Mid
      UINT8_T bid_theo_c  // Bid, Theoretical Mark
      UINT8_T ask_theo_c  // Ask, Theoretical Mark
      UINT8_T marg_theo_c  // Margin, Settlement Price Theoretical Mark
      UINT8_T fix_theo_c  // Fixing value, Origin
   }
}
```

3.7.24.5 Usage and conditions

Series

should be zeroed.

Sub user

should be set to blank, except when used from Genium INET Clearing Back Office Server.

3.7.25 RQ3 [Extended Margin Parameters for series QUERY]

3.7.25.1 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

3.7.25.2 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.

3.7.25.3 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
}
```

3.7.25.4 Usage and conditions

Series

© The NASDAQ OMX Group, Inc. • 2013

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.

3.7.25.5 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
   }
}
```

3.7.25.6 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.

3.7.26 RQ6 [Extended Margin Information QUERY]

3.7.26.1 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

3.7.26.2 Purpose

This query contains margin requirements at a detailed level per account and series.

3.7.26.3 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
}
```

3.7.26.4 Usage and conditions

This query is only available when the signal BI7, Information type 8 has been sent.

Series

© The NASDAQ OMX Group, Inc. • 2013

must be completed with **Country Number** and **Market Code**.

3.7.26.5 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
   }
}
```

3.7.27 RQ7 [Margin Detail QUERY]

3.7.27.1 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

3.7.27.2 Purpose

The purpose of this transaction is to retrieve margin results on a detailed level, that is, per account and series.

3.7.27.3 Structure

The RQ7 QUERY has the following structure:

© The NASDAQ OMX Group, Inc. • 2013

```
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
}
```

3.7.27.4 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 (N/A for NASDAQ OMX Nordic)
2	Intra day margin call, propagated (N/A for NASDAQ OMX Nordic)
10	Evening data, non-propagated
11	Intra day calculation, non-propagated (N/A for NASDAQ OMX Nordic)

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.

3.7.27.5 Answer Structure

The RA7 ANSWER has the following structure:

```
struct answer_margin_detail {
```


© The NASDAQ OMX Group, Inc. • 2013

```
    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: 430] {
       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
    }
}
```

3.7.27.6 Answer, comments

Time Created

equals calculation time in the intra day case. The field is blank in the evening case.

Quantity Cover

Always zero in NASDAQ OMX Nordic 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

© The NASDAQ OMX Group, Inc. • 2013

- If net margining is applied, Held marginable and Written Marginable are netted down so that one of the sides equals zero (0).

3.7.28 RQ12 [Extended Margin Vector QUERY]

3.7.28.1 Fingerprint

QUERY properties	
transaction type	RQ12
calling sequence	omniapi_query_ex
struct name	query_margin_vector_ext
facility	EP4
partitioned	false
answers	RA12

ANSWER properties	
transaction type	RA12
struct name	answer_margin_vector_ext
segmented	true

3.7.28.2 Purpose

This query returns a list of margin vector values for margin calculations using the window method as margining method.

3.7.28.3 Structure

The RQ12 QUERY has the following structure:

```
struct query_margin_vector_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_day3_c  // Intra Day3
   CHAR filler_1_s  // Filler
}
```

3.7.28.4 Usage and conditions

For each option series that has an open interest, there exists one Margin Vector containing the margin requirement for 3 different volatilities of the underlying security and a variable number of calculation points. Other series than options do also have margin vector values constructed in a similar way. The margin requirement for a position can be calculated by using the Margin Vectors for each of the series in the position.

© The NASDAQ OMX Group, Inc. • 2013

This query is only available when the signal BI7, Information type 9 (Evening data), type 41 (Preliminary data) or type 42 (Intra day data) has been sent.

Series

must be completed with Country Number and Market Code.

Intra Day 3

may have one of the following values:

0	Evening data
1	Latest intra day calculation
3	Preliminary data

3.7.28.5 Answer Structure

The RA12 ANSWER has the following structure:

```
struct answer_margin_vector_ext {
   struct transaction_type
   UINT16_T segment_number_n  // Segment Number
   UINT16_T items_n  // Items
   char[8] created_date_s  // Date, Created
   char[6] created_time_s  // Time, Created
   char[2] filler_2_s  // Filler
   Array ITEM [max no: 1000] {
      struct series  // Named struct no: 50000
      UINT32_T point_i  // Point number
      UINT32_T spot_i  // Spot
      UINT32_T held_low_i  // Held, Low
      UINT32_T written_low_i  // Written, Low
      UINT32_T held_middle_i  // Held, Middle
      UINT32_T written_middle_i  // Written, Middle
      UINT32_T held_high_i  // Held, High
      UINT32_T written_high_i  // Written, High
      char[8] created_date_s  // Date, Created
   }
}
```

3.7.28.6 Answer, comments

For the intra day case, Date created and Time created contain intra day calculation date and time. For evening and preliminary data, these fields are blank.

For intra day calculations, data will not be returned for new TM series that have been added during the day.

© The NASDAQ OMX Group, Inc. • 2013

3.7.29 RQ20 [Account Product Area Margin QUERY]

3.7.29.1 Fingerprint

QUERY properties	
transaction type	RQ20
calling sequence	omniapi_query_ex
struct name	query_margin_pa_acc
facility	EP4
partitioned	false
answers	RA20

ANSWER properties	
transaction type	RA20
struct name	answer_margin_pa_acc
segmented	true

3.7.29.2 Purpose

This query contains sum margin requirement per account, product area and instrument currency.

3.7.29.3 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
}
```

3.7.29.4 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 NASDAQ OMX Group, Inc. • 2013

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.

3.7.29.5 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
   }
}
```

3.7.29.6 Answer, comments

The key to the answer items consists of the following fields:

- **Customer**
- **Account**
- **Product Area**
- **Currency, Market**

© The NASDAQ OMX Group, Inc. • 2013

3.7.30 RQ21 [Account Sum Margin QUERY]

3.7.30.1 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

3.7.30.2 Purpose

This query contains sum margin requirement per account, currency and custodian bank together with currency conversions made.

3.7.30.3 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
}
```

3.7.30.4 Usage and conditions

This query is only available when the signal BI7, Information type 11 has been sent.

Series

© The NASDAQ OMX Group, Inc. • 2013

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

3.7.30.5 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
   }
}
```

3.7.30.6 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**

© The NASDAQ OMX Group, Inc. • 2013

3.7.31 RQ23 [Member Sum Margin QUERY]

3.7.31.1 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
struct name	answer_margin_mem
segmented	true

3.7.31.2 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.

3.7.31.3 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
}
```

3.7.31.4 Usage and conditions

This query is only available when the signal BI7, Information type 11 has been sent.

Series

© The NASDAQ OMX Group, Inc. • 2013

could be completed with any **Country Number** and **Market Code**.

3.7.31.5 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
   }
}
```

3.7.31.6 Answer, comments

The key to the answer items consists of the following fields:

- **Customer**
- **Currency, Risk**
- **Custodian Bank**

3.7.32 RQ31 [Margin Exchange Rate QUERY]

3.7.32.1 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

3.7.32.2 Purpose

This query contains exchange rates used in margin calculations.

3.7.32.3 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
}
```

3.7.32.4 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**.

3.7.32.5 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
   }
}
```

3.7.32.6 Answer, comments

Currency, Price

© The NASDAQ OMX Group, Inc. • 2013

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.

3.7.33 RQ35 [Data Used for Margin Calculation QUERY]

3.7.33.1 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

3.7.33.2 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.

3.7.33.3 Structure

The RQ35 QUERY has the following structure:

© The NASDAQ OMX Group, Inc. • 2013

```
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
}
```

3.7.33.4 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.

3.7.33.5 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
```


© The NASDAQ OMX Group, Inc. • 2013

```
        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
    }
}
```

3.7.33.6 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.

3.7.34 RQ36 [Greeks QUERY]

3.7.34.1 Fingerprint

QUERY properties	
transaction type	RQ36
calling sequence	omniapi_query_ex
struct name	query_greeks
facility	EP4

© The NASDAQ OMX Group, Inc. • 2013

QUERY properties	
partitioned	false
answers	RA36

ANSWER properties	
transaction type	RA36
struct name	answer_greeks
segmented	true

3.7.34.2 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.

3.7.34.3 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
}
```

3.7.34.4 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.

3.7.34.5 Answer Structure

The RA36 ANSWER has the following structure:

© The NASDAQ OMX Group, Inc. • 2013

```
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
   }
}
```

3.7.34.6 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.

3.7.35 RQ41 [Margin Underlying Price QUERY]

3.7.35.1 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
struct name	answer_margin_ulg_price
segmented	true

3.7.35.2 Purpose

This query contains underlying prices used in margin calculations.

> **Note:** RQ41 will be replaced by RQ45.

3.7.35.3 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
}
```

3.7.35.4 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.

3.7.35.5 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
   }
}
```

3.7.35.6 Answer, comments

The response is a list of underlyings together with prices used in margin calculations.

© The NASDAQ OMX Group, Inc. • 2013

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.

3.7.36 RQ42 [Margin Series Price QUERY]

3.7.36.1 Fingerprint

QUERY properties	
transaction type	RQ42
calling sequence	omniapi_query_ex
struct name	query_margin_series_price
facility	EP4
partitioned	false
answers	RA42

ANSWER properties	
transaction type	RA42
struct name	answer_margin_series_price
segmented	true

3.7.36.2 Purpose

This query contains series prices used in margin calculations.

Note: RQ42 will be replaced by RQ46.

3.7.36.3 Structure

The RQ42 QUERY has the following structure:

```
struct query_margin_series_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
}
```

3.7.36.4 Usage and conditions

Series

must be completed by **Country Number** and **Market Code**.

3.7.36.5 Answer Structure

The RA42 ANSWER has the following structure:

```
struct answer_margin_series_price {
   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
      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
   }
}
```

3.7.36.6 Answer, comments

The response is a list of series together with prices used in margin calculations.

The answer is available at the same time as the margin information is available, as indicated by the broadcast BI7, information type 8.

3.7.37 RQ44 [Margin Underlying Real Time Price QUERY]

3.7.37.1 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

© The NASDAQ OMX Group, Inc. • 2013

3.7.37.2 Purpose

This query contains real time underlying prices.

3.7.37.3 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
}
```

3.7.37.4 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.

3.7.37.5 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
   }
}
```

3.7.38 RQ45 [Margin Underlying Price Extended QUERY]

3.7.38.1 Fingerprint

QUERY properties	
transaction type	RQ45
calling sequence	omniapi_query_ex
struct name	query_margin_ulg_price_ext
facility	EP4
partitioned	false
answers	RA45

ANSWER properties	
transaction type	RA45
struct name	answer_margin_ulg_price_ext
segmented	true

3.7.38.2 Purpose

The purpose of this transaction is to retrieve underlying prices used in margin calculations.

3.7.38.3 Structure

The RQ45 QUERY has the following structure:

```
struct query_margin_ulg_price_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
}
```

3.7.38.4 Usage and conditions

Series

must be complete up to **Country Number** and **Market Code**. Data will be returned for underlyings having series in the specified market.

3.7.38.5 Answer Structure

The RA45 ANSWER has the following structure:

© The NASDAQ OMX Group, Inc. • 2013

```
struct answer_margin_ulg_price_ext {
   struct transaction_type
   UINT16_T segment_number_n  // Segment Number
   UINT16_T items_n  // Items
   UINT8_T is_preliminary_c  // Is Preliminary
   char[3] filler_3_s  // Filler
   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
   }
}
```

3.7.38.6 Answer, comments

The answer received contains 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 first available as preliminary prices when preliminary vector files are ready, as indicated by the broadcast BI7, information type 41.

The answer is later available as definitive, as indicated by the broadcast BI7, information type 8.

Each response is prefaced with the transaction type (RA45) and an item field specifying the number of records contained in the response.

Is preliminary

specifies if the received prices are preliminary or definitive.

3.7.39 RQ46 [Margin Series Price Extended QUERY]

3.7.39.1 Fingerprint

QUERY properties	
transaction type	RQ46
calling sequence	omniapi_query_ex
struct name	query_margin_series_price_ext
facility	EP4
partitioned	false
answers	RA46

ANSWER properties	
transaction type	RA46
struct name	answer_margin_series_price_ext
segmented	true

3.7.39.2 Purpose

The purpose of this transaction is to retrieve prices and volatility used for series in margin calculations.

3.7.39.3 Structure

The RQ46 QUERY has the following structure:

```
struct query_margin_series_price_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
}
```

3.7.39.4 Usage and conditions

Series

must be complete up to **Country Number** and **Market Code**.

3.7.39.5 Answer Structure

The RA46 ANSWER has the following structure:

```
struct answer_margin_series_price_ext {
   struct transaction_type
   UINT16_T segment_number_n  // Segment Number
   UINT16_T items_n  // Items
   UINT8_T is_preliminary_c  // Is Preliminary
   char[3] filler_3_s  // Filler
   Array ITEM [max no: 500] {
      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 last_paid_i  // Last, Paid
      INT32_T bid_marg_vol_i  // Margin, Volatility Bid
      INT32_T ask_marg_vol_i  // Margin, Volatility Ask
      INT32_T mid_marg_vol_i  // Margin, Volatility Mid
      INT32_T calc_bid_price_i  // Calculation Price, Bid
      INT32_T calc_ask_price_i  // Calculation Price, Ask
      INT32_T calc_marg_price_i  // Calculation Price, Margin
      INT32_T calc_bid_marg_vol_i  // Calculation Margin Volatility, Bid
```


© The NASDAQ OMX Group, Inc. • 2013

```
        INT32_T calc_ask_marg_vol_i  // Calculation Margin Volatility, Ask
        INT32_T calc_mid_marg_vol_i  // Calculation Margin Volatility, Mid
        INT32_T high_price_i  // Price, High
        INT32_T low_price_i  // Price, Low
        INT64_T turnover_u  // Turnover
        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
        UINT8_T calc_bid_theo_c  // Calculation Bid Price, Theoretical Mark
        UINT8_T calc_ask_theo_c  // Calculation Ask Price, Theoretical Mark
       UINT8_T calc_marg_theo_c  // Calculation Margin Settlement Price, Origin
        CHAR filler_1_s  // Filler
    }
}
```

3.7.39.6 Answer, comments

The answer received contains a list of series together with prices used in margin calculations.

The answer is first available as preliminary prices when preliminary vector files are ready, as indicated by the broadcast BI7, information type 41.

The answer is later available as definitive, as indicated by the broadcast BI7, information type 8.

Each response is prefaced with the transaction type (RA46) and an item field specifying the number of records contained in the response.

Is preliminary

specifies if the received prices are preliminary or definitive.

Bid price, Ask price, Margin Settlement Price, Margin volatility bid, Margin volatility ask, Margin volatility mid

do always contain data that is calculated from data of the individual series itself. For options, the prices may be theoretically calculated out from the volatility used.

Calculation Bid price, Calculation Ask price, Calculation Margin Settlement Price, Calculation Margin volatility bid, Calculation Margin volatility ask, Calculation Margin volatility mid

do contain the data that is actually used in the margin calculation.

They differ from the previous fields for options using a “three most at the money rule” for margin volatility. In this case, the calculation volatility is the same for all options with the same underlying, expiration and type (call/put). The calculation price fields contain theoretical prices based on this calculation volatility.

High Price, Low Price and Turnover

© The NASDAQ OMX Group, Inc. • 2013

High Price, Low Price and Turnover are never used in margin calculations; they are only present as informational fields in this query.

3.7.40 RQ60 [Private price list QUERY]

3.7.40.1 Fingerprint

QUERY properties	
transaction type	RQ60
calling sequence	omniapi_query_ex
struct name	query_private_price_list
facility	EP4
partitioned	false
answers	RA60

ANSWER properties	
transaction type	RA60
struct name	answer_private_price_list
segmented	true

3.7.40.2 Related Messages

RC60, RC65, RC66, RQ65, RQ66, RQ71

3.7.40.3 Purpose

This query is used for retrieving data about a private price list. It is used for margin simulations in Genium INET Clearing.

3.7.40.4 Structure

The RQ60 QUERY has the following structure:

```
struct query_private_price_list {
   struct transaction type
   struct series  // Named struct no: 50000
   char[32] sub user s  // Sub User
}
```

3.7.40.5 Usage and conditions

Series

© The NASDAQ OMX Group, Inc. • 2013

should be zeroed.

Sub user

should be set to blank, except when used from Genium INET Clearing Back Office Server.

3.7.40.6 Answer Structure

The RA60 ANSWER has the following structure:

```
struct answer_private_price_list {
   struct transaction type
   char[8] full collect date s  // Full collect date
   char[8] part collect date s  // Partial collect date
   char[6] full collect time s  // Full collect time
   char[6] part collect time s  // Partial collect time
   UINT8_T private price list src c  // Private price list source
   char[3] filler 3 s  // Filler
}
```

3.7.40.7 Answer, comments

The answer contains data about the private price list used for the user sending the query.

3.7.41 RQ65 [Private margin underlying prices QUERY]

3.7.41.1 Fingerprint

QUERY properties	
transaction type	RQ65
calling sequence	omniapi_query_ex
struct name	query_margin_ulg_price_private
facility	EP4
partitioned	false
answers	RA65

ANSWER properties	
transaction type	RA65
struct name	answer_margin_ulg_price_private
segmented	true

3.7.41.2 Related Messages

RC60, RQ60, RC65, RC66, RQ66, RQ71

3.7.41.3 Purpose

This query is used for retrieving underlying prices contained in a private price list. It is used for margin simulations in Genium INET Clearing.

3.7.41.4 Structure

The RQ65 QUERY has the following structure:

```
struct query_margin_ulg_price_private {
   struct transaction type
   struct series  // Named struct no: 50000
   UINT16_T segment_number_n  // Segment Number
   char[32] sub_user_s  // Sub User
   char[6] com_id_s  // Underlying Identity
}
```

3.7.41.5 Usage and conditions

Series

should be zeroed.

Sub user

should be set to blank, except when used from Genium INET Clearing Back Office Server.

Underlying Identity

The field must be filled in one of the following ways:

- Fill in the field with explicit underlying name.

- Fill in the field with “*”. No test is done on underlying name.

- Fill in the field with a string ended by “*”. All underlying names must start by the string given.

3.7.41.6 Answer Structure

The RA65 ANSWER has the following structure:

```
struct answer_margin_ulg_price_private {
   struct transaction type
   UINT16_T segment_number_n  // Segment Number
   UINT16_T items_n  // Items
   Array ITEM [max no: 500] {
      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
      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
```


© The NASDAQ OMX Group, Inc. • 2013

```
        UINT8_T marg_theo_c  // Margin, Settlement Price Theoretical Mark
        char[2] filler_2_s  // Filler
    }
}
```

3.7.41.7 Answer, comments

The answer contains underlying prices from the private price list used for the user sending the query.

3.7.42 RQ66 [Private margin prices and volatilities QUERY]

3.7.42.1 Fingerprint

QUERY properties	
transaction type	RQ66
calling sequence	omniapi_query_ex
struct name	query_margin_series_price_private
facility	EP4
partitioned	false
answers	RA66

ANSWER properties	
transaction type	RA66
struct name	answer_margin_series_price_private
segmented	true

3.7.42.2 Related Messages

RC60, RQ60, RC65, RQ65, RC66, RQ71

3.7.42.3 Purpose

The answer contains data about the private price list used for the user sending the query.

3.7.42.4 Structure

The RQ66 QUERY has the following structure:

```
struct query_margin_series_price_private {
   struct transaction_type
   struct series  // Named struct no: 50000
   UINT16_T segment_number_n  // Segment Number
   char[32] sub_user_s  // Sub User
   char[32] ins_id_s  // Series, Identity
   UINT8_T only_traded_c  // Traded series only
```

```
   CHAR filler_1_s  // Filler
}
```

3.7.42.5 Usage and conditions

Series

should either be zeroed or completed with **Country Number** and **Market code**.

Sub User

should be set to blank, except when used from Genium INET Clearing Back Office Server.

Series Identity

The field must be filled in one of the following ways:

- Fill in the field with explicit series name.

- Fill in the field with “*”. No test is done on series name.

- Fill in the field with a string ended by “*”. All series names must start by the string given.

3.7.42.6 Answer Structure

The RA66 ANSWER has the following structure:

```
struct answer_margin_series_price_private {
   struct transaction_type
   UINT16_T segment_number_n  // Segment Number
   UINT16_T items_n  // Items
   Array ITEM [max no: 600] {
      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 last_paid_i  // Last, Paid
      INT32_T bid_marg_vol_i  // Margin, Volatility Bid
      INT32_T ask_marg_vol_i  // Margin, Volatility Ask
      INT32_T mid_marg_vol_i  // Margin, Volatility Mid
      INT32_T calc_bid_price_i  // Calculation Price, Bid
      INT32_T calc_ask_price_i  // Calculation Price, Ask
      INT32_T calc_marg_price_i  // Calculation Price, Margin
      INT32_T calc_fixing_value_i  // Calculation Price, Fixing
      INT32_T calc_bid_marg_vol_i  // Calculation Margin Volatility, Bid
      INT32_T calc_ask_marg_vol_i  // Calculation Margin Volatility, Ask
      INT32_T calc_mid_marg_vol_i  // Calculation Margin Volatility, Mid
      INT32_T high_price_i  // Price, High
      INT32_T low_price_i  // Price, Low
      INT64_T turnover_u  // Turnover
      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
```


© The NASDAQ OMX Group, Inc. • 2013

```
        UINT8_T fix_theo_c  // Fixing value, Origin
        UINT8_T calc_bid_theo_c  // Calculation Bid Price, Theoretical Mark
        UINT8_T calc_ask_theo_c  // Calculation Ask Price, Theoretical Mark
       UINT8_T calc_marg_theo_c  // Calculation Margin Settlement Price, Origin
        UINT8_T calc_fix_theo_c  // Calculation price, Fixing Origin
        char[3] filler_3_s  // Filler
    }
}
```

3.7.42.7 Answer, comments

The answer contains prices and volatilities from the private price list used for the user sending the query.

3.7.43 RQ71 [Margin Simulation QUERY]

3.7.43.1 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

3.7.43.2 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.

3.7.43.3 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
    }
 }
```

3.7.43.4 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.

Sub User

should be set to blank, except when used from Genium INET Clearing Back Office Server.

Margin Class

For future use. Not applicable.

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:

© The NASDAQ OMX Group, Inc. • 2013

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).

© The NASDAQ OMX Group, Inc. • 2013

Note: If the field Prices Simulated equals 1, the supplied values in the fields **Addedtrades Simulated, Series Expiring today simulated and Futures profit/loss** simulated will be ignored.

3.7.43.5 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.

3.7.43.6 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
   }
}
```


© The NASDAQ OMX Group, Inc. • 2013

3.7.43.7 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

© The NASDAQ OMX Group, Inc. • 2013

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.

© The NASDAQ OMX Group, Inc. • 2013

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.

3.7.44 RQ72 [Added trades in margin simulation QUERY]

3.7.44.1 Fingerprint

QUERY properties	
transaction type	RQ72
calling sequence	omniapi_query_ex
struct name	query_marg_sim_add_trade
facility	EP4
partitioned	false
answers	RA72

ANSWER properties	
transaction type	RA72
struct name	answer_marg_sim_add_trade
segmented	true

© The NASDAQ OMX Group, Inc. • 2013

3.7.44.2 Related Messages

RQ71

3.7.44.3 Purpose

This query is used for retrieving series additional trades for margin simulation that a user has frozen in Genium INET Clearing.

3.7.44.4 Structure

The RQ72 QUERY has the following structure:

```
struct query_marg_sim_add_trade {
   struct transaction_type
   struct series  // Named struct no: 50000
   UINT16_T segment_number_n  // Segment Number
   char[32] sub_user_s  // Sub User
   char[2] filler_2_s  // Filler
}
```

3.7.44.5 Usage and conditions

Series

should be zeroed.

Sub user

should be set to blank, except when used from Genium INET Clearing Back Office Server.

3.7.44.6 Answer Structure

The RA72 ANSWER has the following structure:

```
struct answer_marg_sim_add_trade {
   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
      INT64_T sim_qty_q  // Quantity, Simulation
      INT32_T trade_price_sim_i  // Trade Price, Simulated
      char[8] closing_date_s  // Date, Closing
      char[8] date_settlement_s  // Date, Settlement
      UINT8_T item_type_c  // Item Type
      char[3] filler_3_s  // Filler
      INT32_T reserved_i  // Reserved
      char[8] reserved_8_s  // Reserved
   }
}
```


© The NASDAQ OMX Group, Inc. • 2013

3.7.44.7 Answer, comments

The answer contains additional trades frozen by the querying user. The contents of the answer is the same as when sending in the additional trades via RQ71.

3.8 Collateral management

3.8.1 FB1 [Directed Collateral VIB]

3.8.1.1 Fingerprint

VIB properties	
transaction type	FB1
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

3.8.1.2 Purpose

This broadcast notifies about changes of two kinds. The first kind is when a collateral balance or holding (amount_q) has been created or changed. The other kind is when a fixed margin requirement (a.k.a Member Deposit, used for default fund requirements and base collaterals) has been changed.

3.8.1.3 Structure

The FB1 VIB has the following structure:

```
struct directed_collateral {
   struct broadcast_type
   UINT16_T items_n  // Items
   UINT16_T size_n  // Size
}
Sequence {
   struct sub_item_hdr
   Choice {
      struct collateral_info  // Named struct no: 18000
      struct guarantee  // Named struct no: 18001
      struct member_deposit  // Named struct no: 18002
      struct cash_collateral  // Named struct no: 18003
      struct security  // Named struct no: 18009
   }
}
```

3.8.2 FB6 [Collateral Transaction broadcast (VIM) VIB]

3.8.2.1 Fingerprint

VIB properties	
transaction type	FB6
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

3.8.2.2 Related Messages

FQ22

3.8.2.3 Purpose

This broadcast notifies that a collateral transaction has been created or changed state.

3.8.2.4 Structure

The FB6 VIB has the following structure:

```
struct broadcast_hdr
Sequence {
   struct sub_item_hdr
   Choice {
      struct sequence_number_info  // Named struct no: 18023
      struct deposit_withdraw_collateral  // Named struct no: 18022
      struct collateral_transaction_info  // Named struct no: 18024
      struct corporate_action_info  // Named struct no: 18038
   }
}
```

3.8.2.5 Usage and Conditions

The broadcast contains one VIM item per transaction. Each Vim Item consists of at least two sub_items, a third sub_item is included if collateral position has been adjusted for ongoing corporate action. One sub_item, **DEPOSIT_WITHDRAW_COLLATERAL** (vim 18022) holds information on the collateral transaction data, and the other, **COLLATERAL_TRANSACTION_INFO** (vim 18024) holds information about the status and results for the transaction. **CORPORATE_ACTION_INFO** holds reference to ongoing corporate action. For internal transfers two FB6 are sent based on one internal transfer transaction. One FB6 for the withdrawal and one for the deposit linked via instr_ref_s (SEME).

© The NASDAQ OMX Group, Inc. • 2013

3.8.3 FB17 [Collateral Evaluation Run Broadcast (VIM) VIB]

3.8.3.1 Fingerprint

VIB properties	
transaction type	FB17
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

3.8.3.2 Related Messages

FQ17

3.8.3.3 Purpose

This broadcast notifies that a general collateral evaluation has been successfully completed.

3.8.3.4 Structure

The FB17 VIB has the following structure:

```
struct broadcast_hdr
Sequence {
   struct sub_item_hdr
   Choice {
      struct collateral_evaluation_run_info  // Named struct no: 18033
   }
}
```

3.8.3.5 Usage and Conditions

Account

is always wildcard.

Margin Sequence Number and Margin Date

points out the margin calculation run that this evaluation is based on.

Collateral State

is always completed (=4).

3.8.4 FB18 [Collateral Evaluation Run Broadcast, dedicated (VIM) VIB]

3.8.4.1 Fingerprint

VIB properties	
transaction type	FB18
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

3.8.4.2 Related Messages

FQ18, FB17

3.8.4.3 Purpose

This broadcast notifies that a collateral evaluation for one single account or for a subset of accounts within a participant has been successfully completed. General collateral evaluations is distributed via FB17.

3.8.4.4 Structure

The FB18 VIB has the following structure:

```
struct_broadcast_hdr
Sequence {
   struct_sub_item_hdr
   Choice {
      struct_collateral_evaluation_run_info  // Named struct no: 18033
   }
}
```

3.8.4.5 Usage and Conditions

Account

if the evaluation is for a single account this is specified. If the evaluation is for several accounts, this field only specifies participant.

Margin Sequence Number and Margin Date

points out the margin calculation run that this evaluation is based on.

Collateral State

© The NASDAQ OMX Group, Inc. • 2013

is always completed (=4).

3.8.5 FQ1 [Collateral QUERY]

3.8.5.1 Fingerprint

QUERY properties	
transaction type	FQ1
calling sequence	omniapi_query_ex
struct name	query_collateral
facility	EP3
partitioned	false
segmented	true
answers	FA1

VIA properties	
transaction type	FA1
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

3.8.5.2 Purpose

This query is used to get information of two kinds. The first kind is collateral balances or holdings for a collateral account (below referred to as a collateral position). The other kind is fixed margin requirements (a.k.a Member Deposit, used for default fund requirements and base collaterals) for a margin requirement account.

3.8.5.3 Structure

The FQ1 QUERY has the following structure:

```
struct query_collateral {
   struct transaction_type
   struct series  // Named struct no: 50000
   struct account
   char[32] series_id_s  // Series, Identity
   UINT16_T segment_number_n  // Segment Number
   UINT8_T collateral_type_c  // Collateral types
   UINT8_T state_c  // State
}
```

3.8.5.4 Usage and Conditions

Account

is a collateral account when querying for cash collateral, securities and guarantees, and a margin account when querying for member deposits. If an account is both a collateral account and a margin requirement account, and Collateral Type is set to wildcard, both collaterals and member deposits will be returned. The field is mandatory and wildcards (*) are allowed.

Series ID

must be specified. Wildcards are allowed (*).

Collateral Type

is mandatory. Wildcard (=0) is allowed, and will return all collateral types.

State

is mandatory. Supported values are: 4 (=Active), 10 (=Deleted) and 12 (=Expired). Wildcard (=0) is allowed, and will return all states.

3.8.5.5 Answer Structure

The FA1 VIA has the following structure:

```
struct answer_collateral {
   struct transaction_type
   UINT16_T segment_number_n  // Segment Number
   UINT16_T items_n  // Items
}
Sequence {
   struct item_hdr
   Sequence {
      struct sub_item_hdr
      Choice {
         struct collateral_info  // Named struct no: 18000
         struct guarantee  // Named struct no: 18001
         struct member_deposit  // Named struct no: 18002
         struct cash_collateral  // Named struct no: 18003
         struct security  // Named struct no: 18009
      }
   }
}
```

3.8.5.6 Answer, comments

This query returns the highest version of each active collateral position. It returns data applicable for the querying participant.

Each Vim Item consists of two sub_items. One sub_item holds a structure of common information, collateral_info_t structure (vim 18000), the other holds one of the remaining namned structs.

name_s, user_code_s will always be blank.

© The NASDAQ OMX Group, Inc. • 2013

preliminary_amount_ca_adjusted gives the position that is used in evaluation.

3.8.6 FQ2 [Collateral Version QUERY]

3.8.6.1 Fingerprint

QUERY properties	
transaction type	FQ2
calling sequence	omniapi_query_ex
struct name	query_collateral_version
facility	EP3
partitioned	false
segmented	true
answers	FA2

VIA properties	
transaction type	FA2
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

3.8.6.2 Purpose

This query returns all versions of a specific collateral position, i.e. collateral history position.

3.8.6.3 Structure

The FQ2 QUERY has the following structure:

```
struct query_collateral_version {
   struct transaction type
   struct series  // Named struct no: 50000
   UINT16_T segment_number_n  // Segment Number
   char[2] filler_2_s  // Filler
   UINT64_T collateral_nbr_q  // Collateral Number
}
```

3.8.6.4 Usage and Conditions

Returned collateral history positions are not sorted.

Collateral number

is received via FQ1.

3.8.6.5 Answer Structure

The FA2 VIA has the following structure:

```
struct answer_collateral {
   struct transaction_type
   UINT16_T segment_number_n  // Segment Number
   UINT16_T items_n  // Items
}
Sequence {
   struct item_hdr
   Sequence {
      struct sub_item_hdr
      Choice {
         struct collateral_info  // Named struct no: 18000
         struct guarantee  // Named struct no: 18001
         struct member_deposit  // Named struct no: 18002
         struct cash_collateral  // Named struct no: 18003
         struct security  // Named struct no: 18009
      }
   }
}
```

3.8.6.6 Answer, comments

Each Vim Item consists of two sub_items. One sub_item holds a structure of common information, collateral_info_t structure (vim 18000), the other holds one of the remaining namned structs.

name_s, user_code_s will always be blank.

3.8.7 FQ14 [Collateral Value per Inst Series Query QUERY]

3.8.7.1 Fingerprint

QUERY properties	
transaction type	FQ14
calling sequence	omniapi_query_ex
struct name	query_coll_val_per_series
facility	EP5
partitioned	false
segmented	true
answers	FA14

VIA properties	
transaction type	FA14

© The NASDAQ OMX Group, Inc. • 2013

VIA 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.
segmented	true

3.8.7.2 Related Messages

FQ17

3.8.7.3 Purpose

This query returns the value of collaterals account and instrument series (below referred to as a collateral position) after a CMS evaluation.

3.8.7.4 Structure

The FQ14 QUERY has the following structure:

```
struct query_coll_val_per_series {
   struct transaction_type
   struct series  // Named struct no: 50000
   struct collateral_account  // Of type: ACCOUNT
   struct margin_account
   UINT32_T request_nbr_u  // Request number
   char[12] clh_id_s  // Clearinghouse
   char[32] series_id_s  // Series, Identity
   char[8] valuation_date_s  // Valuation Date
   UINT16_T segment_number_n  // Segment Number
   char[2] filler_2_s  // Filler
}
```

3.8.7.5 Usage and Conditions

Series

should be zero filled.

Valuation date

must be filled with date. Is retrieved from FQ17.

Request number

should either be filled with a request number that refers to an external CMS evaluation retrieved from FQ17, or zero filled, in which case results from the latest available CMS evaluation are returned.

Collateral account

refers to a collateral account. This account could be the same account as the margin requirement account or there could be multiple collateral accounts covering one margin requirement account. Should all (country, customer, account) be filled in with values in one of the following ways:

1. Fill in the field with explicit value. All answers must match the 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.

Margin account

refers to margin requirement account, i.e. the level on which margin requirements should be met with deposited collaterals. Should all be filled in with values in one of the following ways:

1. Fill in the field with explicit value. All answers must match the 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.

Series id

should be filled in with values in one of the following ways:

1. Fill in the field with explicit value. All answers must match the 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.

Clearinghouse Id

should be left blank.

3.8.7.6 Answer Structure

The FA14 VIA has the following structure:

```
struct answer_collateral {
   struct transaction_type
   UINT16_T segment_number_n  // Segment Number
   UINT16_T items_n  // Items
}
Sequence {
   struct item_hdr
   Sequence {
      struct sub_item_hdr
      Choice {
         struct run_info  // Named struct no: 18037
         struct coll_val_per_series  // Named struct no: 18036
         struct coll_val_per_series_risk_cur  // Named struct no: 18026
         struct coll_val_per_series_base_cur  // Named struct no: 18025
      }
   }
}
```


© The NASDAQ OMX Group, Inc. • 2013

3.8.7.7 Answer, comments

First, one VIM item (and sub item) which hold information about the collateral evaluation run that produced the results for all coming collateral position, is sent, **RUN_INFO** (VIM 18037). This item is only included once, and is always the first item.

Thereafter, one VIM (and sub item) item per collateral position is sent. Each such item consists of two sub_items. A sub item **COLL_VAL_PER_SERIES** (VIM 18036) is always returned. It is then followed by either a sub item **COLL_VAL_PER_SERIES_RISK_CUR** (VIM 18026) or a sub item **COLL_VAL_PER_SERIES_BASE_CUR** (VIM 18025).

If no base currency conversion is applied, sub item **COLL_VAL_PER_SERIES_RISK_CUR** is returned. Otherwise, sub item **COLL_VAL_PER_SERIES_BASE_CUR** is returned. A collateral value in sub item **COLL_VAL_PER_SERIES_BASE_CUR** has been converted to the base currency for the account, in order to make it possible to apply and show the result of valuation group limits.

FQ14 returns data for accounts applicable for the querying participant.

3.8.8 FQ15 [Collateral Value per Val Group Query QUERY]

3.8.8.1 Fingerprint

QUERY properties	
transaction type	FQ15
calling sequence	omniapi_query_ex
struct name	query_coll_val_per_val_group
facility	EP5
partitioned	false
segmented	true
answers	FA15

VIA properties	
transaction type	FA15
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

3.8.8.2 Related Messages

FQ17

3.8.8.3 Purpose

This query returns the value of collaterals per margin requirement account and valuation group after a CMS evaluation.

© The NASDAQ OMX Group, Inc. • 2013

3.8.8.4 Structure

The FQ15 QUERY has the following structure:

```
struct query_coll_val_per_val_group {
   struct transaction_type
   struct series  // Named struct no: 50000
   struct margin_account
   UINT32_T request_nbr_u  // Request number
   char[12] clh_id_s  // Clearinghouse
   char[12] vag_id_s  // Valuation Group Identity
   char[8] valuation_date_s  // Valuation Date
   UINT16_T segment_number_n  // Segment Number
   char[2] filler_2_s  // Filler
}
```

3.8.8.5 Usage and Conditions

Series

should be zero filled.

Valuation date

must be filled with date. Is retrieved from FQ17.

Request number

should either be filled with a request number that refers to an external CMS evaluation retrieved from FQ17, or zero filled, in which case results from the latest available CMS evaluation are returned.

Margin account

refers to margin requirement account, i.e. the level on which margin requirements should be met with deposited collaterals. Should all (country, customer, account) be filled in with values in one of the following ways:

1. Fill in the field with explicit value. All answers must match the 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.

Valuation Group Id

should be filled in with values in one of the following ways:

1. Fill in the field with explicit value. All answers must match the 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.

Clearinghouse Id

© The NASDAQ OMX Group, Inc. • 2013

should be left blank.

3.8.8.6 Answer Structure

The FA15 VIA has the following structure:

```
struct answer_collateral {
   struct transaction_type
   UINT16_T segment_number_n  // Segment Number
   UINT16_T items_n  // Items
}
Sequence {
   struct item_hdr
   Sequence {
      struct sub_item_hdr
      Choice {
         struct run_info  // Named struct no: 18037
         struct coll_val_per_val_group_tsn  // Named struct no: 18027
      }
   }
}
```

3.8.8.7 Answer Comments

First, one VIM item (and sub item) which hold information about the collateral evaluation run that produced the results for all coming collateral values per margin requirement account, is sent, **RUN_INFO** (VIM 18037). This item is only included once, and is always the first item.

Thereafter, one VIM item (and sub item) per margin requirement account and valuation group is returned, **COLL_VAL_PER_VAL_GROUP_TSN** (VIM 18027). This item shows how much, in percent, of the total collateral value for the margin requirement account that was stemming from each valuation group, and the allowed percent for each group. Item also shows the collateral value for each group, before and after the limit has been applied.

FQ15 returns data for accounts applicable for the querying participant.

3.8.9 FQ16 [Collateral information (VIM) QUERY]

3.8.9.1 Fingerprint

QUERY properties	
transaction type	FQ16
calling sequence	omniapi_query_ex
struct name	query_collateral_information
facility	EP5
partitioned	false
segmented	true
answers	FA16

© The NASDAQ OMX Group, Inc. • 2013

VIA properties	
transaction type	FA16
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

3.8.9.2 Related Messages

FQ17, FQ18

3.8.9.3 Purpose

The purpose of this query is to retrieve the result of a CMS evaluation. It is a summary per margin requirement account and currency.

3.8.9.4 Structure

The FQ16 QUERY has the following structure:

```
struct query_collateral_information {
   struct transaction_type
   struct series  // Named struct no: 50000
   char[12] clh_id_s  // Clearinghouse
   struct margin_account
   UINT16_T segment_number_n  // Segment Number
   char[2] filler_2_s  // Filler
   char[8] valuation_date_s  // Valuation Date
   UINT32_T request_nbr_u  // Request number
}
```

3.8.9.5 Usage and Conditions

The query returns the results of a collateral evaluation, i.e. a comparison between margin requirements and collateral values, per margin requirement account and currency. A collateral evaluation is uniquely identified by the valuation date and the request number.

> **Note:**
> At each collateral evaluation, there is also a calculation of collateral values, i.e. how much each collateral position is worth.

Series

should be zero filled.

Request number

© The NASDAQ OMX Group, Inc. • 2013

must be filled with a request number that refers to an external CMS evaluation. Is retrieved from FQ17.

Margin account

refers to margin requirement account, i.e. the level on which margin requirements should be met with deposited collaterals. Should all (country, customer, account) be filled in with values in one of the following ways:

1. Fill in the field with explicit value. All answers must match the 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.

Valuation Date

is mandatory and retrieved from FQ17.

Clearinghouse

should be left blank.

3.8.9.6 Answer Structure

The FA16 VIA has the following structure:

```
    struct answer_collateral {
       struct transaction_type
       UINT16_T segment_number_n  // Segment Number
       UINT16_T items_n  // Items
    }
    Sequence {
       struct item_hdr
       Sequence {
          struct sub_item_hdr
          Choice {
             struct run_info  // Named struct no: 18037
             struct collateral_information_base  // Named struct no: 18028
             struct collateral_information_npc  // Named struct no: 18031
             struct collateral_information_payment_delivery  // Named struct
no: 18030
             struct collateral_information_default_fund  // Named struct no:
18029
             struct base_call  // Named struct no: 18043
             struct deficit_to_cover  // Named struct no: 18049
          }
       }
    }
```

3.8.9.7 Answer Comments

The answer contains the result of a collateral evaluation.

First, one VIM item (and sub item) which hold information about the collateral evaluation run that produced the results for all coming margin requirement accounts, is sent, **RUN_INFO** (vim 18037). This item is only included once, and is always the first item.

© The NASDAQ OMX Group, Inc. • 2013

Thereafter, one VIM item per margin requirement account and currency is sent. Each such item consists of at least one sub item, **COLLATERAL_INFORMATION_BASE** (vim 18028). This sub item may then be followed by one of the other sub items, if applicable. **COLLATERAL_INFORMATION_NPC** (vim 18031) is sent in case of margin collateral for NPC. **COLLATERAL_INFORMATION_PAYMENT_DELIVERY** (vim 18030) is sent in case of margin collateral for TSN which have to cover for payment of overdue margin requirements. **COLLATERAL_INFORMATION_DEFAULT_FUND** (vim 18029) is sent in case of default fund collateral. **BASE_CALL** (vim 18043) is sent in case base collateral requirement is calculated.

The number of decimals in the amount fields are decided by the number of decimals defined for each currency respectively.

For evaluations where base currency conversion applies, the total surplus or deficit is returned in query FQ18.

FQ16 returns data for accounts applicable for the querying participant.

An account which has been subject to an intraday margin call - preliminary or final - performed before, but on the same valuation date, as an EOD evaluation, will be excluded from the calculated results for the EOD evaluation. The account is however returned in the result, with figures set to zero, and an indicator showing that the account was excluded due to IDMC.

3.8.10 FQ17 [Collateral evaluation run (VIM) QUERY]

3.8.10.1 Fingerprint

QUERY properties	
transaction type	FQ17
calling sequence	omniapi_query_ex
struct name	query_collateral_evaluation_run
facility	EP5
partitioned	false
segmented	true
answers	FA17

VIA properties	
transaction type	FA17
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

3.8.10.2 Related Messages

FQ16, FQ18, FQ14, FQ15

© The NASDAQ OMX Group, Inc. • 2013

3.8.10.3 Purpose

This query is used to get information about collateral evaluations made by the system.

3.8.10.4 Structure

The FQ17 QUERY has the following structure:

```
struct query_collateral_evaluation_run {
   struct transaction_type
   struct series  // Named struct no: 50000
   struct margin_account
   char[12] clh_id_s  // Clearinghouse
   char[8] valuation_date_s  // Valuation Date
   char[8] created_date_s  // Date, Created
   char[6] from_time_s  // Time, From
   char[6] to_time_s  // Time, To
   UINT16_T segment_number_n  // Segment Number
   UINT8_T collateral_evaluation_type_c  // Collateral evaluation type
   UINT8_T is_final_c  // Final, Is
   UINT8_T is_intraday_c  // Intraday, Is
   char[3] filler_3_s  // Filler
}
```

3.8.10.5 Usage and Conditions

Collateral evaluations (a calculation where the value for all deposited collaterals is compared to the margin requirement) per margin requirement account are made at several occasions during a day.

- An official evaluation made that is the base for how much collaterals that must be deposited each day.
- A preliminary official evaluation is made to give participants a preview of the final evaluation. This occurs normally in the evening on the day before collaterals are due.
- A final official evaluation is made, to make sure that the deposited collaterals actually do cover the margin requirements at the specified due time.

There are also evaluations taking place when a deposit of a new collateral is made to reflect the current value of deposited collaterals, and when a call back of a deposited collateral is requested to ensure that the deposited collateral amount is not decreased below the required amount. These evaluations are made for the affected account only.

If configured, a collateral evaluation may also be performed when an over-the-counter trade is sent in for clearing, before the clearinghouse accepts to "novate" the trade (i.e. to assume the counterparty risk). Here as well, the evaluation is only made for the affected account.

Note that at each evaluation (comparison between margin requirements and collateral values), a calculation of collateral values is also performed (i.e. how much each collateral position is worth).

In order to query for collateral information or values for a specific evaluation, you must know which evaluations that have been run and their respective request numbers for a specific valuation date. This query is used to get information about performed evaluations. It takes a date as input and returns information about

© The NASDAQ OMX Group, Inc. • 2013

evaluations run either for that day or on that day. It is also possible to filter for evaluations of a specific type, and/or evaluations made for a specific account.

Series

should be zero filled.

Valuation Date, Created Date

One of these dates must be filled in. If Created date is used, it is also possible to specify a specific time span.

Margin account

refers to margin requirement account. Should all (country, customer, account) be filled in with values in one of the following ways:

1. Fill in the field with explicit value. All answers must match the 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.

Collateral Evaluation Type

is optional. If given, should be a valid evaluation type. If not given, all valuation types are returned.

Clearinghouse Id

should be left blank.

Is Intraday, Is Final

is optional. If specified, only the matching evaluations will be returned. If not given, no filtering on the field is made.

3.8.10.6 Answer Structure

The FA17 VIA has the following structure:

```
struct answer_collateral {
   struct transaction_type
   UINT16_T segment_number_n  // Segment Number
   UINT16_T items_n  // Items
}
Sequence {
   struct item_hdr
   Sequence {
      struct sub_item_hdr
      Choice {
         struct collateral_evaluation_run_info  // Named struct no: 18033
      }
   }
}
```


© The NASDAQ OMX Group, Inc. • 2013

3.8.10.7 Answer Comments

The answer to the query will only show information about evaluations applicable for the querying user.

Please note that FQ17 will return collateral evaluations made as a result of a Deposit/Withdraw/Internal Transfer request (not distributed in FB17 or FB18).

If an Intraday Margin Call has been made for a specific account, the account is filled in explicitly. If an Intraday Margin Call has been made for a collection of accounts, the account struct will only hold information about the member, and the account field is set to '*'.

3.8.11 FQ18 [Base Currency Conversion (VIM) QUERY]

3.8.11.1 Fingerprint

QUERY properties	
transaction type	FQ18
calling sequence	omniapi_query_ex
struct name	query_base_currency_conversion
facility	EP5
partitioned	false
segmented	true
answers	FA18

VIA properties	
transaction type	FA18
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

3.8.11.2 Related Messages

FQ16, FQ17

3.8.11.3 Purpose

This query is used to get information about the last step in a collateral evaluation, where one surplus or deficit figure per margin requirement account is found, by converting deficits or surplus in different currencies to one figure, expressed in the base currency chosen for the margin requirement account.

3.8.11.4 Structure

The FQ18 QUERY has the following structure:

© The NASDAQ OMX Group, Inc. • 2013

```
struct query_base_currency_conversion {
   struct transaction_type
   struct series  // Named struct no: 50000
   struct margin_account
   UINT32_T request_nbr_u  // Request number
   char[12] clh_id_s  // Clearinghouse
   char[8] valuation_date_s  // Valuation Date
   UINT16_T segment_number_n  // Segment Number
   char[2] filler_2_s  // Filler
}
```

3.8.11.5 Usage and Conditions

Series

should be zero filled.

Margin account

refers to margin requirement account, i.e. the level on which margin requirements should be met with deposited collaterals. . Should all (country, customer, account) be filled in with values in one of the following ways:

1. Fill in the field with explicit value. All answers must match the 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.

Request number

must be filled with a request number that refers to an external CMS evaluation. Is retrieved from FQ17.

Clearinghouse Id

should be left blank.

Valuation Date

is mandatory and retrieved from FQ17.

3.8.11.6 Answer Structure

The FA18 VIA has the following structure:

```
struct answer_collateral {
   struct transaction_type
   UINT16_T segment_number_n  // Segment Number
   UINT16_T items_n  // Items
}
Sequence {
   struct item_hdr
   Sequence {
      struct sub_item_hdr
      Choice {
```


© The NASDAQ OMX Group, Inc. • 2013

```
            struct run_info  // Named struct no: 18037
            struct base_currency_conversion  // Named struct no: 18032
            struct base_currency_conversion_grand_total  // Named struct no:
18035
        }
    }
}
```

3.8.11.7 Answer Comments

First, one VIM item (and sub item) which hold information about the collateral evaluation run that produced the results for all margin requirement accounts is sent, **RUN_INFO** (vim 18037). This item is only included once, and is always the first item.

Thereafter, one VIM item per account and converted currency is returned, **BASE_CURRENCY_CONVERSION** (vim 18032) . At the end of each account, one item containing a grand total for the account is returned, **BASE_CURRENCY_CONVERSION_GRAND_TOTAL** (vim 18035).

FQ18 returns data for accounts applicable for the querying participant.

An account which has been subject to an intraday margin call - preliminary or final - performed before, but on the same valuation date, as an EOD evaluation, will be excluded from the calculated results for the EOD evaluation. The account is however returned in the result, with figures set to zero, and an indicator showing that the account was excluded due to IDMC.

3.8.12 FQ20 [Collateral Transaction QUERY]

3.8.12.1 Fingerprint

QUERY properties	
transaction type	FQ20
calling sequence	omniapi_query_ex
struct name	query_collateral_transaction
facility	EP5
partitioned	false
segmented	true
answers	FA20

VIA properties	
transaction type	FA20
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

3.8.12.2 Purpose

This query is used to retrieve all collateral transactions for which the highest version's:

- state is in any of the specified states.
- created or modified match the supplied time span.
- other attributes match the supplied other search criterias.

3.8.12.3 Structure

The FQ20 QUERY has the following structure:

```
struct query_collateral_transaction {
   struct transaction_type
   struct series  // Named struct no: 50000
   struct collateral_account  // Of type: 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
   char[16] instr_ref_s  // SWIFT reference.
   char[16] cancel_ref_s  // SWIFT reference.
   char[12] ext_acc_registrar_s  // External Account Registrar
   char[15] ext_acc_controller_s  // External Account Controller
   char[34] ext_acc_id_s  // External Account ID
   char[32] series_id_s  // Series, Identity
   UINT8_T collateral_transaction_type_c  // Collateral transaction type
   UINT16_T segment_number_n  // Segment Number
   UINT16_T items_n  // Items
   char[2] filler_2_s  // Filler
   Array ITEM [max no: 20] {
      UINT8_T collateral_transaction_state_c  // Collateral transaction state
   }
}
```

3.8.12.4 Usage and Conditions

If the from/to dates are left blank, all transactions for which the highest version is in any of the specified states are returned.

If the array list of states contains zero items, all collateral transaction states are returned.

Series

should be zero filled.

Collateral Account

should all (country, customer, account) be filled in with values in one of the following ways:

1. Fill in the field with explicit value. All answers must match the field.

© The NASDAQ OMX Group, Inc. • 2013

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.

From Date, From Time, To Date, To Time

specify a time interval when the retrieved collateral transactions were created or the highest version was modified. Can be left blank meaning no time span is defined.

Instrument reference

refers to SWIFT reference and should be filled in with values in one of the following ways:

1. Fill in the field with explicit value. All answers must match the 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.

Cancel reference

should be filled in with values in one of the following ways:

1. Fill in the field with explicit value. All answers must match the 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.

External Account Registrar and External Account Controller

are not used and should be left blank.

External Account ID

refers to depot number in Custody system and should be filled in with values in one of the following ways:

1. Fill in the field with explicit value. All answers must match the 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.

Series id

refers to the name of the collateral instrument. Should be filled in with values in one of the following ways:

1. Fill in the field with explicit value. All answers must match the 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.

Collateral transaction type

should be filled with a specific collateral transaction type or set to zero meaning no filtering on collateral transaction type.

Array list of Collateral transaction state

© The NASDAQ OMX Group, Inc. • 2013

should be filled with an array of collateral transaction states. If array list contains zero items this means no filtering on collateral transaction type.

3.8.12.5 Answer Structure

The FA20 VIA has the following structure:

```
struct answer_segment_hdr
Sequence {
   struct item_hdr
   Sequence {
      struct sub_item_hdr
      Choice {
         struct deposit_withdraw_collateral  // Named struct no: 18022
         struct collateral_transaction_info  // Named struct no: 18024
         struct corporate_action_info  // Named struct no: 18038
      }
   }
}
```

3.8.12.6 Answer Comments

Answer contains one VIM item per transaction. Each Vim Item consists of at least two sub_items, a third sub_item is included if collateral position has been adjusted for ongoing corporate action. One sub_item, **DEPOSIT_WITHDRAW_COLLATERAL** (vim 18022) holds information on the collateral transaction data, and the other, **COLLATERAL_TRANSACTION_INFO** (vim 18024) holds information about the status and results for the transaction. **CORPORATE_ACTION_INFO** holds reference to ongoing corporate action.

reason_s will hold reason for returned status.

FQ20 returns data for accounts applicable for the querying participant.

3.8.13 FQ21 [Collateral Transaction Version QUERY]

3.8.13.1 Fingerprint

QUERY properties	
transaction type	FQ21
calling sequence	omniapi_query_ex
struct name	query_collateral_transaction_version
facility	EP5
partitioned	false
segmented	true
answers	FA21

VIA properties	
transaction type	FA21

© The NASDAQ OMX Group, Inc. • 2013

VIA 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.
segmented	true

3.8.13.2 Related Messages

FQ20

3.8.13.3 Purpose

This query is used to retrieve all collateral transaction versions for a specific collateral transaction number.

3.8.13.4 Structure

The FQ21 QUERY has the following structure:

```
struct query_collateral_transaction_version {
   struct transaction_type
   struct series  // Named struct no: 50000
   UINT64_T collateral_transaction_nbr_q  // Collateral Transaction Number
   UINT16_T segment_number_n  // Segment Number
   char[2] filler_2_s  // Filler
}
```

3.8.13.5 Usage and Conditions

Series

should be zero filled.

Collateral transaction number

should all be filled with a specific collateral transaction number. This number can for example be retrieved using FQ20.

3.8.13.6 Answer Structure

The FA21 VIA has the following structure:

```
   struct answer_segment_hdr
   Sequence {
      struct item_hdr
      Sequence {
         struct sub_item_hdr
         Choice {
            struct deposit_withdraw_collateral  // Named struct no: 18022
            struct collateral_transaction_info  // Named struct no: 18024
```

```
            struct corporate_action_info  // Named struct no: 18038
        }
    }
}
```

3.8.13.7 Answer Comments

Answer contains one VIM item for each version of a collateral transaction. Each Vim Item consists of at least two sub_items, a third sub_item is included if collateral position has been adjusted for ongoing corporate action. One sub_item, **DEPOSIT_WITHDRAW_COLLATERAL** (vim 18022) holds information on the collateral transaction data, and the other, **COLLATERAL_TRANSACTION_INFO** (vim 18024) holds information about the status and results for the transaction. **CORPORATE_ACTION_INFO** holds reference to ongoing corporate action.

reason_s will hold reason for returned status.

Returns all version of the specified collateral transaction number.

3.8.14 FQ22 [Missing Collateral Transaction QUERY]

3.8.14.1 Fingerprint

QUERY properties	
transaction type	FQ22
calling sequence	omniapi_query_ex
struct name	query_missing_collateral_transaction
facility	EP5
partitioned	false
segmented	true
answers	FA22

VIA properties	
transaction type	FA22
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

3.8.14.2 Related Messages

FB6

© The NASDAQ OMX Group, Inc. • 2013

3.8.14.3 Purpose

This query is used to retrieve missing FB6 Collateral transaction Broadcasts for a specific clearing date. For example, if a missing sequence number is detected for the FB6 broadcasts, this query is used to get in synch with the broadcast flow again.

3.8.14.4 Structure

The FQ22 QUERY has the following structure:

```
struct query_missing_collateral_transaction {
   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] clearing_date_s  // Clearing Date
}
```

3.8.14.5 Usage and Conditions

Series

should be zero filled.

Sequence First, Sequence Last

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.

Clearing Date

should contain current business date.

3.8.14.6 Answer Structure

The FA22 VIA has the following structure:

```
   struct answer_segment_hdr
   Sequence {
      struct item_hdr
      Sequence {
         struct sub_item_hdr
         Choice {
            struct sequence_number_info  // Named struct no: 18023
            struct deposit_withdraw_collateral  // Named struct no: 18022
            struct collateral_transaction_info  // Named struct no: 18024
            struct corporate_action_info  // Named struct no: 18038
```

© The NASDAQ OMX Group, Inc. • 2013

```
        }
    }
}
```

3.8.14.7 Answer Comments

Returns the specified sequence of collateral transactions.

Answer contains one VIM item for each missing collateral transaction. Each Vim Item consists of at least three sub_items, an additional sub_item is included if collateral position has been adjusted for ongoing corporate action. The first sub item, **SEQUENCE_NUMBER_INFO** (vim 18023) holds the sequence number for this transaction. Next sub_item, **DEPOSIT_WITHDRAW_COLLATERAL** (vim 18022) holds information on the collateral transaction data, and the third, **COLLATERAL_TRANSACTION_INFO** (vim 18024), holds information about the status and results for the transaction.

reason_s will hold reason for returned status.

3.9 Settlement

3.9.1 SB1 [DvP Instruction BROADCAST]

3.9.1.1 Fingerprint

BROADCAST properties	
transaction type	SB1
calling sequence	omniapi_read_event_ext_ex or omniapi_read_event_block
struct name	dvp_instruction_bdx
info type	dedicated

3.9.1.2 Purpose

This broadcast is sent out for every new DvP instruction created, or when a DvP instruction is changed by either an SC2 or SC3 transaction.

3.9.1.3 Structure

The SB1 BROADCAST has the following structure:

```
struct dvp_instruction_bdx {
   struct broadcast_type
   struct dvp_instruction_api
}
```


© The NASDAQ OMX Group, Inc. • 2013

3.9.1.4 Usage and Conditions

Series

refers to the matched series (the series in the contract).

CSD

is the CSD defined as a participant/member of the exchange. Both parties must have the same CSD.

Sequence no

holds the number within a daily sequence of SB1:s, exclusive for each member.

DvP Sequence no

is the number within a sequence exclusive for each CSD.

Original DvP Sequence no

holds the **DvP Sequence no** for a previously submitted DvP instruction that is subject to change of some kind, and those changes are contained within this new broadcast.

Delivery Unit

is a number containing a number that springs from the the DvP's source, that is, the deal no if the DvP arises from a deal, or the flow no be the DvP created from a flow.

Items

holds the number of DvP Items in the broadcast (at least two).

Length

is the length in bytes of the whole DvP Instruction.

CSD Status

is a blankpadded alphanumeric status code returned from the CSD.

Reason

is a text field typically holding the reason for a potential rejection.

Message Type

is essentially a variant of the string XvY, for example DvP, thereby defining the nature of the instruction.

Operation Type

is a further specification to the Message Type, and states the indended use of the message.

Settlement Status

reflects the status of the DvP towards the CSD.

Chain Info

© The NASDAQ OMX Group, Inc. • 2013

reflects the broadcast's relative position in a potential chain of broadcasts.

Version

holds the version number of this DvP instruction. This number increases for every alteration of the instruction's data.

The broadcast will contain at least 2 DvP items, where the roles of the party/counterparty are opposed. Each party will receive its own copy of the broadcast, so will the CSD involved.

3.9.2 SQ1 [Pay Note QUERY]

3.9.2.1 Fingerprint

QUERY properties	
transaction type	SQ1
calling sequence	omniapi_query_ex
struct name	query_paynote_info
facility	EP4
partitioned	false
answers	SA1

ANSWER properties	
transaction type	SA1
struct name	answer_paynote_info
segmented	true

3.9.2.2 Purpose

The purpose of this query is to obtain the amount of money to be paid or received on a certain settlement date.

3.9.2.3 Structure

The SQ1 QUERY has the following structure:

```
struct query_paynote_info {
   struct transaction type
   struct series  // Named struct no: 50000
   INT32_T pay_note_number_i  // Pay note number
   UINT16_T segment_number_n  // Segment Number
   char[2] filler_2_s  // Filler
   char[8] date_s  // Date
}
```


© The NASDAQ OMX Group, Inc. • 2013

3.9.2.4 Usage and conditions

Series

must be completed with **Country Number** and **Market Code**.

> **Note:**
> The Account Identity (account_id_s) field is now used with an id value.

3.9.2.5 Answer Structure

The SA1 ANSWER has the following structure:

```
struct answer_paynote_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: 250] {
      struct series  // Named struct no: 50000
      struct account
      char[8] clearing_date_s  // Clearing Date
      INT32_T pay_note_number_i  // Pay note number
      UINT8_T event_type_c  // Event Type
      UINT8_T settle_class_c  // Class Number
      char[2] filler_2_s  // Filler
      INT64_T amount_u  // Amount
      char[3] currency_s  // Currency
      UINT8_T pay_or_receive_c  // Deliver/Pay or Receive
     char[8] settlement_instr_date_s  // Date, Settlement Instruction (defined
 for this struct) ; Of type: DATE_S
   }
}
```

3.9.2.6 Answer, comments

Series

must be completed with **Country Number** and **Market Code**.

Account

If the Account field contains a relevant value or not depends on the Clearinghouse's policy.

Pay or Receive

is defined from the member's pesrpective.

Amount

© The NASDAQ OMX Group, Inc. • 2013

The sign is from the Clearinghouse perspective. Negative amount = Clearinghouse pays, which means that the member receive. Positive amount = Clearinghouse receive, which means that the member delivers.

The answer is aggregated per country, market, currency, account, event type and class.

3.9.3 SQ2 [Manual Payment QUERY]

3.9.3.1 Fingerprint

QUERY properties	
transaction type	SQ2
calling sequence	omniapi_query_ex
struct name	query_manual_payments
facility	EP4
partitioned	false
answers	SA2

ANSWER properties	
transaction type	SA2
struct name	answer_manual_payments
segmented	true

3.9.3.2 Purpose

The purpose of this query is to obtain payment manually entered by the clearinghouse (usually fees) via the Settlement Client Application.

The data is available as soon as the manual payment is entered by the clearinghouse.

3.9.3.3 Structure

The SQ2 QUERY has the following structure:

```
struct query_manual_payments {
   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
}
```

3.9.3.4 Usage and conditions

Series

© The NASDAQ OMX Group, Inc. • 2013

must be completed with **Country Number** and **Market Code**.

3.9.3.5 Answer Structure

The SA2 ANSWER has the following structure:

```
struct answer_manual_payments {
   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[8] settlement_date_s  // Date, Settlement
      INT64_T amount_u  // Amount
      char[60] invc_text_s  // Invoice Text
      char[3] currency_s  // Currency
      UINT8_T pay_or_receive_c  // Deliver/Pay or Receive
     char[8] settlement_instr_date_s  // Date, Settlement Instruction (defined
 for this struct) ; Of type: DATE_S
   }
}
```

3.9.3.6 Answer, comments

Series

must be completed with **Country Number** and **Market Code**.

Account

If the Account field contains a relevant value or not depends on the Clearinghouse policy.

3.9.4 SQ4 [Delivery instructions one Settlement Day QUERY]

3.9.4.1 Fingerprint

QUERY properties	
transaction type	SQ4
calling sequence	omniapi_query_ex
struct name	query_delivery_instruction
facility	EP4
partitioned	false
answers	SA4

© The NASDAQ OMX Group, Inc. • 2013

ANSWER properties	
transaction type	SA4
struct name	answer_delivery_instruction
segmented	true

3.9.4.2 Purpose

The purpose of this query is to obtain information regarding the Delivery Instructions.

3.9.4.3 Structure

The SQ4 QUERY has the following structure:

```
struct query_delivery_instruction {
   struct transaction_type
   struct series  // Named struct no: 50000
   UINT16_T segment_number_n  // Segment Number
   char[8] settlement_date_s  // Date, Settlement
   char[2] filler_2_s  // Filler
}
```

3.9.4.4 Usage and Conditions

Series

must be completed with Country Number.

Country Number

can be a wildcard, 0.

Instead of having the deliveries specified with the clearing house as a counterpart, delivery instructions are deliveries that have been calculated for bi-lateral deliveries. Genium INET Clearing has, based on an algorithm, paired deliveries between the members.

3.9.4.5 Answer Structure

The SA4 ANSWER has the following structure:

```
struct answer_delivery_instruction {
   struct transaction_type
   UINT16_T segment_number_n  // Segment Number
   UINT16_T items_n  // Items
   Array ITEM [max no: 100] {
      UINT32_T sequence_number_u  // Sequence Number
      struct participant {
         char[2] country_id_s  // Name, Country
         char[5] ex_customer_s  // Customer, Identity
         CHAR filler_1_s  // Filler
      }
```


© The NASDAQ OMX Group, Inc. • 2013

```
      char[8] clearing_date_s  // Clearing date of Exercise/Closing (defined
for this struct)
      char[8] settlement_date_s  // Settlement date in CSD (defined for this
struct)
      char[12] isin_code_s  // ISIN Code of delivered underlying (defined for
 this struct)
      char[6] com_id_s  // Underlying Identity
      char[2] filler_2_s  // Filler
      INT64_T deliv_isin_quantity_q  // Nbr of underlying to be
delivered(-)/Recieved(+) (defined for this struct)
      INT64_T delivery_quantity_q  // Settlement Amount to Pay(-)/Receive(+)
(defined for this struct)
      struct party
   }
}
```

3.9.5 SQ5 [DvP Instruction, Missing QUERY]

3.9.5.1 Fingerprint

QUERY properties	
transaction type	SQ5
calling sequence	omniapi_query_ex
struct name	query_missing_dvp_instruction
facility	EP3
partitioned	false
answers	SA5

ANSWER properties	
transaction type	SA5
struct name	answer_missing_dvp_instruction
segmented	true

3.9.5.2 Purpose

This query is used by clients to recover missing SB1 broadcasts, or if the client has a more batch like approach to execute. It handles recovery of today's broadcasts.

3.9.5.3 Structure

The SQ5 QUERY has the following structure:

```
struct query_missing_dvp_instruction {
   struct transaction_type
   struct series  // Named struct no: 50000
   char[8] date_s  // Date
```

```
    INT32_T sequence_first_i  // Number, First Sequential
    INT32_T sequence_last_i  // Number, Last Sequential
}
```

3.9.5.4 Answer Structure

The SA5 ANSWER has the following structure:

```
struct answer_missing_dvp_instruction {
    struct transaction_type
    UINT16_T items_n  // Items
    char[2] filler_2_s  // Filler
    Array ITEM [max no: 40] {
        struct dvp_instruction_api
    }
}
```

3.9.5.5 Answer, comments

The answer is a list of DVP Instruction records where the DVP Instruction items are included. The number of items in one DVP instruction is variable, which means that the size of one DVP can vary and that the number of DVP records in one answer varies.

3.9.6 SQ6 [DvP Instruction, Historic QUERY]

3.9.6.1 Fingerprint

QUERY properties	
transaction type	SQ6
calling sequence	omniapi_query_ex
struct name	query_historic_dvp_instruction
facility	EP5
partitioned	false
answers	SA6

ANSWER properties	
transaction type	SA6
struct name	answer_historic_dvp_instruction
segmented	true

3.9.6.2 Purpose

This query is used by clients to recover missing SB1 broadcasts, or if the client has a more batch like approach to execute. It handles recovery of previous date’s broadcasts.

© The NASDAQ OMX Group, Inc. • 2013

3.9.6.3 Structure

The SQ6 QUERY has the following structure:

```
struct query_historic_dvp_instruction {
   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
}
```

3.9.6.4 Answer Structure

The SA6 ANSWER has the following structure:

```
struct answer_historic_dvp_instruction {
   struct transaction_type
   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: 40] {
      struct dvp_instruction_api
   }
}
```

3.9.6.5 Answer, comments

The answer is a list of DVP Instruction records where the DVP Instruction items are included. The number of items in one DVP instruction is variable, which means that the size of one DVP can vary and that the number of DVP records in one answer varies.

3.9.7 SQ14 [Paynote details QUERY]

3.9.7.1 Fingerprint

QUERY properties	
transaction type	SQ14
calling sequence	omniapi_query_ex
struct name	query_paynote_info_detail
facility	EP4
partitioned	false
segmented	true
answers	SA14

© The NASDAQ OMX Group, Inc. • 2013

VIA properties	
transaction type	SA14
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

3.9.7.2 Purpose

This query retrieves details for paynotes, manual payments or pending settlement.

3.9.7.3 Structure

The SQ14 QUERY has the following structure:

```
struct query_paynote_info_detail {
   struct transaction_type
   struct series  // Named struct no: 50000
   INT32_T pay_note_number_i  // Pay note number
   struct account
   UINT16_T segment_number_n  // Segment Number
   char[8] settlement_date_from_s  // Date, Settlement ; Of type:
SETTLEMENT_DATE_S
   char[8] settlement_date_to_s  // Date, Settlement ; Of type:
SETTLEMENT_DATE_S
   UINT8_T incl_manual_registrations_c  // Include manual, not invoiced,
registrations
   UINT8_T incl_paynotes_c  // Include invoiced or payed paynotes
   UINT8_T incl_pending_settlements_c  // Include pending, not invoiced
settlements
   char[3] filler_3_s  // Filler
}
```

3.9.7.4 Usage and Conditions

The query is multi-purpose. It returns overview or detailed information about paynotes and detailed information about manual registrations and pending settlements.

Detailed information from paynotes only returns data for one specific paynote number per query.

The following fields must be set in the query struct query_paynote_info_detail:

- incl_ manual_registrations_c; value 1 will include manual payments.
- incl_paynotes_c; value 1 will include generated paynotes.
- incl_pending_settlements_c; value 1 will include future settlement.

Settlement Date From/To

is filled with the date range requested.

Account

© The NASDAQ OMX Group, Inc. • 2013

can be filled in as a filter for manual registrations and future settlements account.

pay_note_number_i

is used to read all details for a specific paynote number.

> **Note:**
> The Account Identity (account_id_s) field is now used with an id value.

3.9.7.5 Answer Structure

The SA14 VIA has the following structure:

```
struct answer_partition_hdr {
   struct transaction_type
   struct partition_low
   struct partition_high
   UINT16_T items_n  // Items
   UINT16_T size_n  // Size
   UINT16_T segment_number_n  // Segment Number
   char[2] filler_2_s  // Filler
}
Sequence {
   struct item_hdr
   Sequence {
      struct sub_item_hdr
      Choice {
         struct paynote_info_detail  // Named struct no: 19001
         struct paynote_info_detail_item  // Named struct no: 19002
      }
   }
}
```

3.9.8 SQ16 [All Pay Notes Created one Settlement Instruction Day QUERY]

3.9.8.1 Fingerprint

QUERY properties	
transaction type	SQ16
calling sequence	omniapi_query_ex
struct name	query_created_paynote_info
facility	EP4
partitioned	false
answers	SA16

ANSWER properties	
transaction type	SA16
struct name	answer_created_paynote_info
segmented	true

3.9.8.2 Related Messages

SQ1

3.9.8.3 Purpose

The purpose of this query is for participants to obtain their paynotes created on a certain settlement instruction date.

3.9.8.4 Structure

The SQ16 QUERY has the following structure:

```
struct query_created_paynote_info {
   struct transaction_type
   struct series  // Named struct no: 50000
   UINT16_T segment_number_n  // Segment Number
   char[2] exchange_code_s  // Exchange Code
   char[8] date_s  // Date
}
```

3.9.8.5 Usage and Conditions

The transaction is similar to SQ1 (which queries with respect to settlement date and per member) and SQ2058 (which queries with respect to clearing date and per exchange). The returned information will contain all payment information for the participant, created by the settlement system that date.

Series

must be complete with Country Number, Market Code.

Exchange extra short name

is the exchange to which the member belongs.

Date

the settlement instruction date when payments were created.

> **Note:**
> The Account Identity (account_id_s) field is now used with an id value.

© The NASDAQ OMX Group, Inc. • 2013

3.9.8.6 Answer Structure

The SA16 ANSWER has the following structure:

```
struct answer_created_paynote_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: 250] {
      struct series  // Named struct no: 50000
      struct account
      char[8] settlement_date_s  // Date, Settlement
      UINT8_T event_type_c  // Event Type
      UINT8_T settle_class_c  // Class Number
      char[2] filler_2_s  // Filler
      INT64_T amount_u  // Amount
      char[3] currency_s  // Currency
      UINT8_T pay_or_receive_c  // Deliver/Pay or Receive
   }
}
```

3.10 Reports

3.10.1 LQ1 [List QUERY]

3.10.1.1 Fingerprint

QUERY properties	
transaction type	LQ1
calling sequence	omniapi_query_ex
struct name	query_list
facility	EP4
partitioned	false
answers	LA1

ANSWER properties	
transaction type	LA1
struct name	answer_list
segmented	true

3.10.1.2 Purpose

The purpose of this transaction is to transfer text files.

Reports are sent as ASCII ISO Latin 1 text files.

> **Note:** A translation service to the local computer's character set is provided by the OMnet API (omniapi_cvt_string).

3.10.1.3 Structure

The LQ1 QUERY has the following structure:

```
struct query_list {
   struct transaction_type
   struct series  // Named struct no: 50000
   UINT16_T segment_number_n  // Segment Number
   char[6] yymmdd_s  // Trading Date
   INT32_T info_type_i  // Information Type
}
```

3.10.1.4 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.

3.10.1.5 Answer Structure

The LA1 ANSWER has the following structure:

```
struct answer_list {
   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
   UINT16_T items_n  // Items
   char[50000] text_buffer_s  // Text, Buffer
}
```

3.10.1.6 Answer, comments

Item Number

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

The reports can be downloaded like this:

© The NASDAQ OMX Group, Inc. • 2013

Information type	Available reports
256	List of available reports which can be downloaded using this transaction.
<nnn>	<specific report>

3.10.2 LQ2 [Available Reports QUERY]

3.10.2.1 Fingerprint

QUERY properties	
transaction type	LQ2
calling sequence	omniapi_query_ex
struct name	query_report
facility	EP4
partitioned	false
answers	LA2

ANSWER properties	
transaction type	LA2
struct name	answer_report
segmented	true

3.10.2.2 Purpose

This query is used for asking for available Reports.

3.10.2.3 Structure

The LQ2 QUERY has the following structure:

```
struct query_report {
   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
}
```

3.10.2.4 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 LQ1 and 'possible' = description about reports that can be created in the system.

A query about 'available' reports will return LAST 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.

3.10.2.5 Answer Structure

The LA2 ANSWER has the following structure:

```
struct answer_report {
   struct transaction_type
   UINT16_T segment_number_n  // Segment Number
   UINT16_T items_n  // Items
   Array ITEM [max no: 351] {
      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[5] ex_customer_s  // Customer, Identity
      char[32] name_s  // Name
      char[40] description_s  // Description
      UINT8_T ascii_bin_c  // ASCII or Binary
   }
}
```

3.10.3 LQ3 [List with Version QUERY]

3.10.3.1 Fingerprint

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

© The NASDAQ OMX Group, Inc. • 2013

ANSWER properties	
transaction type	LA3
struct name	answer_list_ver
segmented	true

3.10.3.2 Purpose

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

3.10.3.3 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
}
```

3.10.3.4 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.

3.10.3.5 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
}
```

© The NASDAQ OMX Group, Inc. • 2013

3.10.3.6 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.

3.10.4 LQ4 [Available Reports with Version QUERY]

3.10.4.1 Fingerprint

QUERY properties	
transaction type	LQ4
calling sequence	omniapi_query_ex
struct name	query_report_ver
facility	EP4
partitioned	false
answers	LA4

ANSWER properties	
transaction type	LA4
struct name	answer_report_ver
segmented	true

3.10.4.2 Purpose

This transaction is used for querying for available report versions.

3.10.4.3 Structure

The LQ4 QUERY has the following structure:

```
struct query_report_ver {
   struct transaction_type
```


© The NASDAQ OMX Group, Inc. • 2013

```
    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
}
```

3.10.4.4 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.

3.10.4.5 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
   }
}
```

3.10.4.6 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.

3.10.5 LR5 [NRS List with Version QUERY]

3.10.5.1 Fingerprint

QUERY properties	
transaction type	LR5
calling sequence	omniapi_query_ex
struct name	query_list_ver_nrs
facility	EP1
partitioned	false
answers	LA5

ANSWER properties	
transaction type	LA5
struct name	answer_list_ver_nrs
segmented	true

3.10.5.2 Purpose

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

3.10.5.3 Structure

The LR5 QUERY has the following structure:

```
struct query_list_ver_nrs {
   struct transaction_type
   UINT16_T segment_number_n  // Segment Number
   char[80] file_name_s  // File Name
}
```


© The NASDAQ OMX Group, Inc. • 2013

3.10.5.4 Usage and conditions

To find out which file to query for, first make a LR6 query.

NRS = New Report Server

File Name

is the specific list you want to query for.

3.10.5.5 Answer Structure

The LA5 ANSWER has the following structure:

```
struct answer_list_ver_nrs {
   struct transaction_type
   UINT16_T buffer_length_n  // Buffer Length
   UINT16_T segment_number_n  // Segment Number
   char[80] file_name_s  // File Name
   INT32_T report_no_i  // Report Number
   char[2] country_id_s  // Name, Country
   char[5] ex_customer_s  // Customer, Identity
   char[5] report_spec_s  // Report Specification
   char[12] clh_id_s  // Clearinghouse
   char[8] business_date_s  // Date, Business
   char[3] report_version_s  // Report Version
   UINT8_T ascii_bin_c  // ASCII or Binary
   UINT8_T[61440] data_buffer_s  // Data, Buffer
}
```

3.10.5.6 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.

ASCII or Binary

is binary for all files, but CSV files can be handled as ASCII.

The response is received as a list of text lines.

3.10.6 LR6 [NRS Available Reports with Version QUERY]

3.10.6.1 Fingerprint

QUERY properties	
transaction type	LR6

© The NASDAQ OMX Group, Inc. • 2013

QUERY properties	
calling sequence	omniapi_query_ex
struct name	query_report_nrs
facility	EP1
partitioned	false
answers	LA6

ANSWER properties	
transaction type	LA6
struct name	answer_report_nrs
segmented	true

3.10.6.2 Purpose

This transaction is used for querying for available report versions.

3.10.6.3 Structure

The LR6 QUERY has the following structure:

```
struct query_report_nrs {
   struct transaction_type
   INT32_T report_no_i  // Report Number
   UINT16_T segment_number_n  // Segment Number
   char[2] country_id_s  // Name, Country
   char[5] ex_customer_s  // Customer, Identity
   char[5] report_spec_s  // Report Specification
   char[12] clh_id_s  // Clearinghouse
   char[8] business_date_s  // Date, Business
   UINT8_T only_account_reports_c  // Only Account Reports
   CHAR filler_1_s  // Filler
}
```

3.10.6.4 Usage and conditions

Report Number

is used to identify the report. Each report template is assigned a unique number.

Report Specification

specifies the products the report is created for.

Business date

© The NASDAQ OMX Group, Inc. • 2013

must be specified. Wildcard is not allowed.

3.10.6.5 Answer Structure

The LA6 ANSWER has the following structure:

```
struct answer_report_nrs {
   struct transaction_type
   UINT16_T segment_number_n  // Segment Number
   UINT16_T items_n  // Items
   Array ITEM [max no: 351] {
      INT32_T report_no_i  // Report Number
      char[2] country_id_s  // Name, Country
      char[5] ex_customer_s  // Customer, Identity
      char[5] report_spec_s  // Report Specification
      char[12] clh_id_s  // Clearinghouse
      char[8] business_date_s  // Date, Business
      char[3] report_version_s  // Report Version
      UINT8_T ascii_bin_c  // ASCII or Binary
      char[80] file_name_s  // File Name
      char[8] file_type_s  // File Type
      char[8] created_date_s  // Date, Created
      char[6] created_time_s  // Time, Created
      char[2] filler_2_s  // Filler
   }
}
```

3.10.6.6 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.

File Type

contains the suffix of the report file.

The response is received as a list of text lines.

3.11 Miscellaneous

3.11.1 BI7 [Signal Information Ready BROADCAST]

3.11.1.1 Fingerprint

BROADCAST properties	
transaction type	BI7

© The NASDAQ OMX Group, Inc. • 2013

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

3.11.1.2 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.

3.11.1.3 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
}
```

3.11.1.4 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 NASDAQ OMX Group, Inc. • 2013

Information type	Interpretation	Comment
		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.
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.

© The NASDAQ OMX Group, Inc. • 2013

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

3.11.2 BI26 [Pay note information ready BROADCAST]

3.11.2.1 Fingerprint

BROADCAST properties	
transaction type	BI26
calling sequence	omniapi_read_event_ext_ex or omniapi_read_event_block
struct name	pay_note_info_ready
info type	general

3.11.2.2 Purpose

This is a transaction that signals when Pay Note Information is available.

© The NASDAQ OMX Group, Inc. • 2013

3.11.2.3 Structure

The BI26 BROADCAST has the following structure:

```
struct pay_note_info_ready {
   struct broadcast_type
   INT32_T info_type_i  // Information Type
   char[8] settlement_date_s  // Date, Settlement
   char[12] clh_id_s  // Clearinghouse
   char[8] sent_date_s  // Date, Sent
   char[6] sent_time_s  // Time, Sent
   char[8] clearing_date_s  // Date ; Of type: DATE_S
   char[2] filler_2_s  // Filler
}
```

3.11.3 BI73 [Undo Signal Ready Info BROADCAST]

3.11.3.1 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

3.11.3.2 Purpose

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

3.11.3.3 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
}
```

© The NASDAQ OMX Group, Inc. • 2013

3.11.4 BI74 [Dedicated Broker to Broker Message Info BROADCAST]

3.11.4.1 Fingerprint

BROADCAST properties	
transaction type	BI74
calling sequence	omniapi_read_event_ext_ex or omniapi_read_event_block
struct name	dedicated_message_info
info type	dedicated

3.11.4.2 Related Messages

BI75, BI76

3.11.4.3 Purpose

The Dedicated Message Information Broadcast is used for sending dedicated messages to recipients.

3.11.4.4 Structure

The BI74 BROADCAST has the following structure:

```
struct dedicated_message_info {
   struct broadcast_type
   struct sender_user_code
   UINT64_T message_id_q  // Message, Identity
   char[50] sender_alias_s  // Sender Alias
   char[8] yyyymmdd_s  // Date
   char[6] hhmmss_s  // Time, External
   struct message_text
   UINT8_T urgent_c  // Urgent
   char[3] filler_3_s  // Filler
}
```

3.11.4.5 Usage and Conditions

The Dedicated Message Information Broadcast is triggered by either UI5 or UI6 (or their respective internal version), and is always followed by a BI75. If the Dedicated Message Information Broadcast is not sent to one or more intended recipients, a BI76 will be sent to the original sender of the message.

Applications using the OMnet subscriptions mechanism (see OMnet Application Programmer's Interface Manual) that want to receive the BI74 broadcast must set the value of the member_info_n member variable of the infobj_t struct for BI74 to 1 when setting up subscriptions using the OMnet API function omniapi_set_event_ex.

Sender

© The NASDAQ OMX Group, Inc. • 2013

Sender specifies the full signature (exchange code, customer code, and user signature) of the user who is sending the message. It is set by the central system and this field will be left blank if the sender has claimed to be anonymous.

Date

Date specifies the date in local time. Set by the central system.

Time

Time specifies the time in local time. Set by the central system.

Message ID

Message ID is an identification value that uniquely identifies the message. Set by the central system.

Message Text

Message text is the actual message to be distributed.

Urgent

Urgent can obtain the value True or False and will indicate whether this message should be treated as urgent. Using this information the client can take special actions for important messages, for example display the message in different colours, use beeps, etc.

3.11.5 BI75 [General Broker to Broker Message Info BROADCAST]

3.11.5.1 Fingerprint

BROADCAST properties	
transaction type	BI75
calling sequence	omniapi_read_event_ext_ex or omniapi_read_event_block
struct name	general_message_info
info type	general

3.11.5.2 Related Messages

BI76

3.11.5.3 Purpose

The General Message Information Broadcast is used for sending general messages to recipients that are allowed to listen to them.

3.11.5.4 Structure

The BI75 BROADCAST has the following structure:

```
struct general_message_info {
```

© The NASDAQ OMX Group, Inc. • 2013

```
    struct broadcast_type
    struct sender_user_code
    UINT64_T message_id_q  // Message, Identity
    char[50] sender_alias_s  // Sender Alias
    char[8] yyyymmdd_s  // Date
    char[6] hhmmss_s  // Time, External
    struct message_text
    UINT16_T items_n  // Items
    UINT8_T urgent_c  // Urgent
    CHAR filler_1_s  // Filler
    Array ITEM [max no: 40] {
       struct user_code
    }
}
```

3.11.5.5 Usage and conditions

Sender

Sender specifies the full signature (exchange code, customer code, and user signature) of the user who is sending the message. It is set by the central system and this field will be left blank if the sender has claimed to be anonymous.

Sender Alias

is a field that contains a more user-friendly name of the sender. May be blank.

Date

Date specifies the date in local time. Set by the central system.

Time

Time specifies the time in local time. Set by the central system.

Items

Items specifies the number of recipients that the original broadcast (that triggered this broadcast) was sent to.

Message ID

Message ID is an identification value that uniquely identifies the message. Set by the central system.

Message Text

Message text is the actual message to be distributed.

Urgent

Urgent may have the value True or False and will indicate whether this message should be treated as urgent. Using this information, the client can take special actions for important messages, for example, display the message in different colours, and use beeps.

Recipient

© The NASDAQ OMX Group, Inc. • 2013

Recipient specifies a recipient of the original broadcast (that triggered this broadcast). A recipient can either be a customer firm (exchange code and customer code) or an individual user (exchange code, customer code, and user signature).

3.11.6 BI76 [Broker to Broker Message Status BROADCAST]

3.11.6.1 Fingerprint

BROADCAST properties	
transaction type	BI76
calling sequence	omniapi_read_event_ext_ex or omniapi_read_event_block
struct name	message_status
info type	dedicated

3.11.6.2 Purpose

The Message Status Broadcast is used for sending the message status to the sender of the message.

3.11.6.3 Structure

The BI76 BROADCAST has the following structure:

```
struct message_status {
   struct broadcast_type
   UINT64_T message_id_q  // Message, Identity
   UINT16_T items_n  // Items
   char[2] filler_2_s  // Filler
   Array ITEM [max no: 80] {
      UINT8_T reason_u  // Reason
      char[3] filler_3_s  // Filler
      struct user_code
   }
}
```

3.11.6.4 Usage and conditions

The Message Status Broadcast is triggered if there is a problem with sending the original broadcast. When triggered, it is sent to the original sender of the message that generated the broadcast.

Applications using the OMnet subscriptions mechanism that want to receive the BI76 broadcast must set the value of the member_info_n member variable of the infobj_t struct for BI76 to 1 when setting up subscriptions using the OMnet API function omniapi_set_event_ex.

Items

© The NASDAQ OMX Group, Inc. • 2013

Items specifies the number of recipients that the original message was not sent to.

Message ID

Message ID is an identification value that uniquely identifies the message. Set by the central system.

Reason

The reason for why the original message was not sent (and consequently, why this Message Status Broadcast message was sent) is given per recipient.

Note that if a recipient of a message is a firm and not all users on that firm are logged on, a Message Status Broadcast will not be triggered.

Recipient

Recipient specifies a recipient that the original message was not sent to. A recipient can either be a customer firm (exchange code and customer code) or an individual user (exchange code, customer code, and user signature).

3.11.7 BI81 [Market Announcement Information VIB]

3.11.7.1 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

3.11.7.2 Purpose

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

3.11.7.3 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
   }
```


© The NASDAQ OMX Group, Inc. • 2013

```
}
```

3.11.7.4 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.

3.11.7.5 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.

© The NASDAQ OMX Group, Inc. • 2013

Link, URL holds the URL link pointing to a document where e.g. a full announcement can be found.

3.11.8 BI93 [Report ready BROADCAST]

3.11.8.1 Fingerprint

BROADCAST properties	
transaction type	BI93
calling sequence	omniapi_read_event_ext_ex or omniapi_read_event_block
struct name	report_ready
info type	general

3.11.8.2 Related Messages

DQ58, LR5, LR6

3.11.8.3 Purpose

This broadcast is disseminated every time a new RS report is created.

3.11.8.4 Structure

The BI93 BROADCAST has the following structure:

```
struct report_ready {
   struct broadcast_type
   INT32_T report_no_i  // Report Number
   char[2] country_id_s  // Name, Country
   char[12] clh_id_s  // Clearinghouse
   char[5] report_spec_s  // Report Specification
   char[8] business_date_s  // Date, Business
   char[8] as_of_date_s  // Date, As Of
   char[8] sent_date_s  // Date, Sent
   char[6] sent_time_s  // Time, Sent
   char[3] filler_3_s  // Filler
}
```

3.11.8.5 Usage and Conditions

Bi93 will be disseminated every time a new RS report is created. The receiving application should check in the authorized reports list for the participant, as retrieved by DQ58, if the participant is allowed to access the report specified in the received Bi93 broadcast. If so, LR5 should be used to transfer the report.

© The NASDAQ OMX Group, Inc. • 2013

Note that report number and report specification is unique per report. Note also that even if the participant is authorized for the report, no report may exist to transfer.

Report Number

will be filled in with the Report Template number for the created report.

Report Specification

will be filled in with the specification for which products the report is created for.

3.11.9 II2148 [Set Supervision Reference Price by Issuer TRANSACTION]

3.11.9.1 Fingerprint

TRANSACTION properties	
transaction type	II2148
calling sequence	omniapi_tx_ex
struct name	set_ref_price_trans
facility	EP1
partitioned	false

3.11.9.2 Purpose

This transaction is used to set reference prices for series.

3.11.9.3 Structure

The II2148 TRANSACTION has the following structure:

```
struct set_ref_price_trans {
   struct transaction_type
   struct series  // Named struct no: 50000
   UINT16_T items_n  // Items
   char[2] filler_2_s  // Filler
   Array ITEM [max no: 3500] {
      struct series  // Named struct no: 50000
      INT32_T reference_price_i  // REFERENCE_PRICE_I
   }
}
```

3.11.9.4 Usage and conditions

Series

is not used and should be zero filled.

Price, Reference

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.

The series and price in the item list can be repeated up to 3500 times.

3.11.10 UI1 [Application Status TRANSACTION]

3.11.10.1 Fingerprint

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

3.11.10.2 Purpose

The Application Status Transaction is used to inform the central Marketplace that the application is fully initialized and ready for normal processing. An application is ready for normal processing when it has logged on and all necessary initializations are executed.

3.11.10.3 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
}
```

3.11.10.4 Usage and Conditions

Series

is a reserved field and is not in use.

Status

must be equal to one.

After a successful UI1, the marketplace is aware of the fact that the client is initialized. There are no return codes.

© The NASDAQ OMX Group, Inc. • 2013

3.11.11 UI5 [External Dedicated Message TRANSACTION]

3.11.11.1 Fingerprint

TRANSACTION properties	
transaction type	UI5
calling sequence	omniapi_tx_ex
struct name	dedicated_message
facility	EP0
partitioned	false

3.11.11.2 Related Messages

UI261 is the internal variant, UI6 is the anonymous sender variant

3.11.11.3 Purpose

The Dedicated Message Transaction is used for sending dedicated a message to one or several recipients.

3.11.11.4 Structure

The UI5 TRANSACTION has the following structure:

```
struct dedicated_message {
   struct transaction_type
   struct series  // Named struct no: 50000
   char[50] sender_alias_s  // Sender Alias
   char[2] filler_2_s  // Filler
   struct message_text
   UINT16_T items_n  // Items
   UINT8_T urgent_c  // Urgent
   CHAR filler_1_s  // Filler
   Array ITEM [max no: 500] {
      struct user_code
   }
}
```

3.11.11.5 Usage and conditions

UI5 is the external variant of the Dedicated Message Transaction.

The Dedicated Message Transaction will generate a BI74 and a BI75. If the BI74 broadcast is not sent to one or more intended recipients, a BI76 will be sent to the sender of this message.

Message Text

Message text is the actual message to be distributed.

Items

Items specifies the number of recipients of the message.

Urgent

Urgent can obtain the value True or False and will indicate whether this message should be treated as urgent. Using this information the client can take special actions for important messages, for example display the message in different colours, use beeps, etc.

Only users configured as internal are allowed to send an Urgent message. The API does not allow external users to do this. If the central system receives a message transaction from an external user, the urgent field is set to false in the subsequent broadcasts regardless of what the received value was set to.

Recipient

Recipient specifies a recipient of the message. An item in the list can either be a customer firm (exchange code and customer code) or an individual user (exchange code, customer code, and user signature).

3.11.11.6 Return Codes

After a successful UI5 transaction, a transaction ID will be returned to the sender. The Message ID included in the generated broadcast equals this transaction ID.

Cstatus	Txstat	ordidt
Successful	transaction ID	-
Transaction aborted	Please refer to the **Error Messages Reference Manual** for details about why transactions are aborted.	-

3.11.12 UI6 [External Anonymous Dedicated Message TRANSACTION]

3.11.12.1 Fingerprint

TRANSACTION properties	
transaction type	UI6
calling sequence	omniapi_tx_ex
struct name	anonymous_dedicated_message
facility	EP0
partitioned	false

3.11.12.2 Related Messages

UI262 is the internal variant, UI5 is the non-anonymous sender variant

© The NASDAQ OMX Group, Inc. • 2013

3.11.12.3 Purpose

The Anonymous Dedicated Message Transaction is used for anonymously sending a dedicated message to one or several recipients.

3.11.12.4 Structure

The UI6 TRANSACTION has the following structure:

```
struct anonymous_dedicated_message {
   struct transaction_type
   struct series  // Named struct no: 50000
   char[50] sender_alias_s  // Sender Alias
   char[2] filler_2_s  // Filler
   struct message_text
   UINT16_T items_n  // Items
   UINT8_T urgent_c  // Urgent
   CHAR filler_1_s  // Filler
   Array ITEM [max no: 500] {
      struct user_code
   }
}
```

3.11.12.5 Usage and conditions

UI6 is the external variant of the Anonymous Dedicated Message Transaction.

The Anonymous Dedicated Message Transaction will generate a BI74 and a BI75. If the BI74 broadcast is not sent to one or more intended recipients, a BI76 will be sent to the sender of this message.

Sender is for the broadcasts (that this transaction will generate) set automatically by the central system and will be left blank since this is an anonymous message.

Sender Alias

Sender Alias will be left blank in the broadcasts that this transaction will generate since this is an anonymous message.

Message Text

Message text is the actual message to be distributed.

Items

Items specifies the number of recipients of the message.

Urgent

Urgent can obtain the value True or False and will indicate whether this message should be treated as urgent. Using this information the client can take special actions for important messages, for example display the message in different colours, use beeps, etc.

© The NASDAQ OMX Group, Inc. • 2013

Only users configured as internal are allowed to send an Urgent message. The API does not allow external users to do this. If the central system receives a message transaction from an external user, the urgent field is set to false in the subsequent broadcasts regardless of what the received value was set to.

Recipient

Recipient specifies a recipient of the message. An item in the list can either be a customer firm (exchange code and customer code) or an individual user (exchange code, customer code, and user signature).

3.11.12.6 Return Codes

After a successful UI6 transaction, a transaction ID will be returned to the sender. The Message ID included in the generated broadcast equals this transaction ID.

Cstatus	Txstat	ordidt
Successful	transaction ID	-
Transaction aborted	Please refer to the **Error Messages Reference Manual** for details about why transactions are aborted.	-

3.11.13 UQ1 [Partition QUERY]

3.11.13.1 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

3.11.13.2 Purpose

This query will return all partition information.

3.11.13.3 Structure

The UQ1 QUERY has the following structure:

© The NASDAQ OMX Group, Inc. • 2013

```
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
}
```

3.11.13.4 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
      }
   }
}
```

3.11.13.5 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

© The NASDAQ OMX Group, Inc. • 2013

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.

3.11.14 UQ9 [BI7 Signals Sent QUERY]

3.11.14.1 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

3.11.14.2 Purpose

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

3.11.14.3 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
}
```


© The NASDAQ OMX Group, Inc. • 2013

3.11.14.4 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
   }
}
```

3.11.15 UQ10 [BI26 Signal Sent QUERY]

3.11.15.1 Fingerprint

QUERY properties	
transaction type	UQ10
calling sequence	omniapi_query_ex
struct name	query_bi26_signals_sent
facility	EP1
partitioned	false
answers	UA10

ANSWER properties	
transaction type	UA10
struct name	answer_bi26_signals_sent
segmented	true

3.11.15.2 Purpose

The purpose of this query is to retrieve all Signal Pay Note Information (BI26) signals that have been sent for the date given in the query.

3.11.15.3 Structure

The UQ10 QUERY has the following structure:

© The NASDAQ OMX Group, Inc. • 2013

```
struct query_bi26_signals_sent {
   struct transaction_type
   UINT16_T segment_number_n  // Segment Number
   char[8] settlement_date_s  // Date, Settlement
   char[2] filler_2_s  // Filler
}
```

3.11.15.4 Answer Structure

The UA10 ANSWER has the following structure:

```
struct answer_bi26_signals_sent {
   struct transaction_type
   UINT16_T segment_number_n  // Segment Number
   UINT16_T items_n  // Items
   Array ITEM [max no: 1000] {
      INT32_T info_type_i  // Information Type
      char[8] settlement_date_s  // Date, Settlement
      char[8] clearing_date_s  // Clearing Date
      char[12] clh_id_s  // Clearinghouse
      char[8] sent_date_s  // Date, Sent
      char[6] sent_time_s  // Time, Sent
      char[2] filler_2_s  // Filler
   }
}
```

3.11.16 UQ12 [Business Date QUERY]

3.11.16.1 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

3.11.16.2 Purpose

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

© The NASDAQ OMX Group, Inc. • 2013

3.11.16.3 Structure

The UQ12 QUERY has the following structure:

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

3.11.16.4 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.

3.11.16.5 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
}
```

3.11.16.6 Answer, comments

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

3.11.17 UQ13 [BI27 Broadcasts Sent QUERY]

3.11.17.1 Fingerprint

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

© The NASDAQ OMX Group, Inc. • 2013

ANSWER properties	
transaction type	UA13
struct name	answer_bi27_broadcasts_sent
segmented	true

3.11.17.2 Purpose

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

3.11.17.3 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
}
```

3.11.17.4 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
      }
   }
}
```

3.11.17.5 Answer, comments

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

© The NASDAQ OMX Group, Inc. • 2013

3.11.18 UQ14 [BI81 Broadcasts Sent QUERY]

3.11.18.1 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

3.11.18.2 Purpose

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

3.11.18.3 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
}
```

3.11.18.4 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.

3.11.18.5 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
      }
   }
}
```

3.11.18.6 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.

© The NASDAQ OMX Group, Inc. • 2013

Date Time, External	Time stamps in UTC.

3.11.19 UQ20 [BI73 Signals Sent QUERY]

3.11.19.1 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

3.11.19.2 Purpose

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

3.11.19.3 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
}
```

3.11.19.4 Usage and Conditions

The query is sent to the Supervision subsystem.

© The NASDAQ OMX Group, Inc. • 2013

3.11.19.5 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
   }
}
```

3.11.19.6 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.

© The NASDAQ OMX Group, Inc. • 2013

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
}
```

© The NASDAQ OMX Group, Inc. • 2013

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 AUTH_BY_WHOM

```
struct auth_by_whom {
   struct trading_code
   char[32] name_s  // Name
}
```

4.6 BASE_TRADE_REPORT

```
struct base_trade_report {
   struct party
   struct account
   struct series
   char[32] passthrough_s  // Passthrough Information
   char[8] settlement_date_s  // Date, Settlement
   char[8] asof_date_s  // Date, As Of
   char[24] cash_account_s  // Account, Cash
   char[24] security_account_s  // Account, Security
   char[80] participant_info_s  // Participant Info
   char[32] name_s  // Name
   char[120] sell_ssi_s  // Sell SSI
   UINT8_T bought_or_sold_c  // Bought or Sold
   UINT8_T use_ssi_c  // Use SSI
   UINT8_T trade_report_category_c  // Trade Report Category
   UINT8_T novation_c  // Novation
   UINT8_T payment_settlement_c  // Payment settled by CSD Yes/ No
   char[3] filler_3_s  // Filler
}
```


© The NASDAQ OMX Group, Inc. • 2013

4.7 BROADCAST_HDR

```
struct broadcast_hdr {
   struct broadcast_type
   UINT16_T items_n  // Items
   UINT16_T size_n  // Size
}
```

4.8 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
}
```

4.9 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
}
```

4.10 CHANGES

```
struct changes {
   UINT32_T sequence_number_u  // Sequence Number
   UINT32_T ob_position_u  // Order Book Position
   INT64_T quantity_difference_i  // Quantity, Difference
   UINT8_T ob_command_c  // Order-Book Command
   UINT8_T change_reason_c  // Change Reason
   UINT8_T combo_mark_c  // Combination Order Mark
   CHAR filler_1_s  // Filler
}
```

4.11 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
```

© The NASDAQ OMX Group, Inc. • 2013

```
      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
   struct clearing_account  // Of type: ACCOUNT
   char[4] filler_4_s  // Filler
   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
}
```

4.12 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
```


© The NASDAQ OMX Group, Inc. • 2013

```
   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
}
```

4.13 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
}
```

4.14 COUNTERSIGN

```
struct countersign {
   char[2] country_id_s  // Name, Country
   char[5] ex_customer_s  // Customer, Identity
   CHAR filler_1_s  // Filler
}
```

4.15 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
}
```

4.16 CSD

```
struct csd {
   char[2] country_id_s  // Name, Country
   char[5] ex_customer_s  // Customer, Identity
   CHAR filler_1_s  // Filler
}
```

© The NASDAQ OMX Group, Inc. • 2013

4.17 CURRENCY

```
struct currency {
   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
}
```

4.18 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
}
```

4.19 DVP_INSTRUCTION_API

```
struct dvp_instruction_api {
   struct series
   struct csd
   struct modified_by
   UINT32_T dvp_sequence_number_u  // DVP_SEQUENCE_NUMBER_U
   UINT32_T dvp_properties_u  // Delivery Properties
   UINT32_T sequence_number_u  // Sequence Number
   UINT32_T orig_dvp_sequence_number_u  // ORIG_DVP_SEQUENCE_NUMBER_U
   UINT32_T delivery_unit_u  // Delivery Unit
   INT32_T version_i  // VERSION_I
   UINT16_T items_n  // Items
   UINT16_T dvp_length_n  // DVP_LENGTH_N
   char[8] created_date_s  // Date, Created
   char[6] created_time_s  // Time, Created
   char[8] timestamp_date_s  // Timestamp, Date
   char[6] timestamp_time_s  // Timestamp, Time
   char[16] csd_status_s  // CSD Status
   char[80] reason_s  // Reason
   char[4] operation_type_s  // Operation Type
   char[8] settlement_date_s  // Date, Settlement
   char[8] clearing_date_s  // Clearing Date
   char[3] message_type_s  // Message Type
   UINT8_T settle_status_c  // Settlement Status
```


© The NASDAQ OMX Group, Inc. • 2013

```
   UINT8_T chain_info_c  // Chain Info
   UINT8_T sub_settle_status_c  // Settlement Sub-status
   char[8] settlement_instr_date_s  // Date ; Of type: DATE_S
   char[2] filler_2_s  // Filler
   Array ITEM [max no: 2] {
      struct account
      struct party_account  // Of type: ACCOUNT
      struct confirmed_by  // Of type: TRADING_CODE
      INT64_T first_quantity_q  // Quantity, First
      INT64_T second_quantity_q  // Quantity, Second
      UINT32_T dvp_item_number_u  // DVP_ITEM_NUMBER_U
      UINT32_T dvp_item_properties_u  // DVP_ITEM_PROPERTIES_U
      UINT16_T dec_in_first_quantity_n  // DEC_IN_FIRST_QUANTITY_N
      UINT16_T dec_in_second_quantity_n  // DEC_IN_SECOND_QUANTITY_N
      char[34] csd_code_s  // Code, CSD
      char[34] party_csd_code_s  // CSD code, Counterpart
      char[24] first_dvp_account_s  // FIRST_DVP_ACCOUNT_S
      char[12] first_isin_code_s  // FIRST_ISIN_CODE_S
      char[24] second_dvp_account_s  // SECOND_DVP_ACCOUNT_S
      char[12] second_isin_code_s  // SECOND_ISIN_CODE_S
      char[32] passthrough_s  // Passthrough Information
      char[16] external_ref_s  // External reference
      char[16] instr_ref_s  // SWIFT reference.
      UINT8_T state_item_c  // STATE_ITEM_C
      UINT8_T bought_or_sold_c  // Bought or Sold
      char[2] filler_2_s  // Filler
   }
}
```

4.20 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
}
```

4.21 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
}
```

4.22 IR_SWAP

```
struct ir_swap {
   struct base_trade_report
```

© The NASDAQ OMX Group, Inc. • 2013

```
   struct upfront  // Of type: PAYMENT
   char[8] date_termination_s  // Date, Maturity
   INT64_T notional_amount_q  // Notional amount
   char[256] swap_condition_s  // Swap condition
   char[5] first_holiday_id_s  // First State Holiday ID
   char[3] filler_3_s  // Filler
   char[5] second_holiday_id_s  // Second State Holiday ID
   UINT8_T apply_holiday_c  // State holiday applied, Yes/No
   UINT8_T business_day_conv_c  // BUSINESS_DAY_CONV_C
   UINT8_T rate_reset_c  // Rate Reset
   UINT8_T reset_days_c  // Reset Days
   UINT8_T payment_set_c  // Payment Set
   char[2] filler_2_s  // Filler
   struct member_pay  // Of type: IR_SWAP_LEG
   struct counterparty_pay  // Of type: IR_SWAP_LEG
}
```

4.23 IR_SWAP_LEG

```
struct ir_swap_leg {
   INT32_T fixed_interest_rate_i  // Fixed Interest Rate
   struct float_rate_index  // Of type: SERIES
   INT32_T spread_i  // Spread
   INT32_T init_interest_rate_i  // Init Interest Rate
   char[8] first_rollover_date_s  // First Rollover Date
   UINT8_T day_count_conv_c  // Day Count Convention
   UINT8_T rollover_period_c  // Rollover Period
   UINT8_T rollover_day_c  // Rollover Day
   UINT8_T fixed_or_float_c  // Fixed or Float
   struct party_pay  // Of type: PARTY
}
```

4.24 ITEM_HDR

```
struct item_hdr {
   UINT16_T items_n  // Items
   UINT16_T size_n  // Size
}
```

4.25 MARGIN_ACCOUNT

```
struct margin_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
}
```


© The NASDAQ OMX Group, Inc. • 2013

4.26 MARGIN_ACCOUNT

```
struct margin_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
}
```

4.27 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
}
```

4.28 MESSAGE_TEXT

```
struct message_text {
   Array ITEM [max no: 10] {
      char[80] text_line_s  // Text, Line
   }
}
```

4.29 MODIFIED_BY

```
struct modified_by {
   char[2] country_id_s  // Name, Country
   char[5] ex_customer_s  // Customer, Identity
   char[5] user_id_s  // User
}
```

4.30 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
}
```

© The NASDAQ OMX Group, Inc. • 2013

4.31 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
}
```

4.32 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
}
```

4.33 OM_EXCHANGE_INFO

```
struct om_exchange_info {
   INT32_T clean_price_i  // Clean price
   UINT8_T order_capacity_c  // Order Capacity
   CHAR filler_1_s  // Filler
   CHAR[4] clearing_firm_s  // Clearing Firm
   CHAR[12] clearing_account_s  // Clearing Account
   char[10] order_reference_s  // Order Reference
}
```

4.34 ORDER

```
struct order {
   struct series
   struct trading_code
   struct order_var
   struct ex_user_code
   struct give_up_member
  struct exchange_info_s  // Internally overlayed structure: OM_EXCHANGE_INFO
   UINT32_T order_index_u  // Order Index
   UINT16_T transaction_number_n  // Transaction Type Number
   UINT8_T change_reason_c  // Change Reason
```


© The NASDAQ OMX Group, Inc. • 2013

```
   CHAR filler_1_s  // Filler
}
```

4.35 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
}
```

4.36 ORDER_TRANS_HDR

```
struct order_trans_hdr {
   struct transaction_type
   struct series
   UINT16_T items_n  // Items
   char[2] filler_2_s  // Filler
}
```

4.37 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
}
```

4.38 ORIGINATOR_TRADING_CODE

```
struct originator_trading_code {
   char[2] country_id_s  // Name, Country
   char[5] ex_customer_s  // Customer, Identity
```

© The NASDAQ OMX Group, Inc. • 2013

```
   char[5] user_id_s  // User
}
```

4.39 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
}
```

4.40 OTC_TRADE_REPORT

```
struct otc_trade_report {
   struct trading_code
   struct user_code
   struct auth_by_whom
   UINT32_T delivery_unit_u  // Delivery Unit
   UINT32_T trade_report_type_i  // Trade Report Type
   UINT64_T trade_report_nbr_q  // Trade report number
   UINT64_T party_trade_report_nbr_q  // Party trade report number
   INT32_T sequence_number_i  // Sequence Number
   UINT32_T netting_req_nbr_u  // Netting request number
   UINT32_T pay_calc_req_nbr_u  // Pay calc request number
   UINT32_T novation_sequence_nbr_u  // Novation sequence number
   INT32_T deal_number_i  // Deal Number
   UINT16_T trade_report_version_n  // Trade report version
   char[8] timestamp_date_s  // Timestamp, Date
   char[6] timestamp_time_s  // Timestamp, Time
   char[12] isin_code_s  // ISIN Code
   char[24] agreement_type_s  // Agreement, Type
   UINT8_T trade_report_state_c  // Trade Report State
   UINT8_T trade_report_sub_state_c  // Trade Report Substate
   UINT8_T authorization_state_c  // Authorization State
   UINT8_T confirm_letter_c  // Trade report reason ; Of type:
TRADE_REPORT_REASON_C
   UINT8_T use_agreement_c  // Use agreement
   char[3] filler_3_s  // Filler
}
```

4.41 PARTITION_HIGH

```
struct partition_high {
   UINT8_T country_c  // Country Number
   UINT8_T market_c  // Market Code
```


© The NASDAQ OMX Group, Inc. • 2013

```
   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 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
}
```

4.43 PARTY

```
struct party {
   char[2] country_id_s  // Name, Country
   char[5] ex_customer_s  // Customer, Identity
   CHAR filler_1_s  // Filler
}
```

4.44 PAYMENT

```
struct payment {
   struct paying_member  // Of type: PARTY
   char[8] settlement_date_s  // Date, Settlement
   INT64_T amount_q  // Amount
   struct currency  // Of type: SERIES
}
```

4.45 PHYSICAL_SERIES

```
struct physical_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
}
```

© The NASDAQ OMX Group, Inc. • 2013

4.46 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
}
```

4.47 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
}
```

4.48 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
}
```

4.49 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
}
```

4.50 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
}
```


© The NASDAQ OMX Group, Inc. • 2013

4.51 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
}
```

4.52 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
}
```

4.53 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
}
```

4.54 QUERY_HDR

```
struct query_hdr {
   struct transaction_type
   struct series
   UINT16_T items_n  // Items
   UINT16_T size_n  // Size
}
```

4.55 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
```

© The NASDAQ OMX Group, Inc. • 2013

```
   INT32_T strike_price_i  // Strike Price
}
```

4.56 SENDER_USER_CODE

```
struct sender_user_code {
   char[2] country_id_s  // Name, Country
   char[5] ex_customer_s  // Customer, Identity
   char[5] user_id_s  // User
}
```

4.57 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
}
```

4.58 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
}
```

4.59 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
}
```


© The NASDAQ OMX Group, Inc. • 2013

4.60 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
}
```

4.61 SUB_ITEM_HDR

```
struct sub_item_hdr {
   UINT16_T named_struct_n  // Named Struct, Number
   UINT16_T size_n  // Size
}
```

4.62 SWAP_FLOW

```
struct swap_flow {
   UINT64_T trade_report_nbr_q  // Trade report number
   UINT32_T flow_number_u  // FLOW_NUMBER_U
   struct party
   char[8] start_date_s  // Date, Start
   char[8] end_date_s  // Date, End
   char[8] fixing_date_s  // Fixing Date
   INT32_T fixing_value_i  // Fixing Value
   INT64_T notional_amount_q  // Notional amount
   char[8] settlement_date_s  // Date, Settlement
   INT64_T consideration_q  // Consideration
   struct currency  // Of type: SERIES
   UINT16_T days_in_period_n  // Days in Period
   UINT8_T fixed_or_float_c  // Fixed or Float
   UINT8_T leg_number_c  // Leg Number
   INT64_T accumulated_consideration_q  // Consideration, Accumulated
}
```

4.63 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
}
```

© The NASDAQ OMX Group, Inc. • 2013

4.64 TIME_SPEC

```
struct time_spec {
   UINT32_T tv_sec  // Time in seconds
   INT32_T tv_nsec  // Time in nanoseconds
}
```

4.65 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
}
```

4.66 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
}
```

4.67 TRD_RPT_CUST

```
struct trd_rpt_cust {
   struct party
   char[10] ex_client_s  // Client
   char[15] customer_info_s  // Customer, Information
   struct exchange_info_s  // Internally overlayed structure: OM_EXCHANGE_INFO
   UINT8_T open_close_req_c  // Open  Close Request
   UINT16_T exch_order_type_n  // Order Type, Exchange
   struct give_up_member
}
```

4.68 UL_SERIES

```
struct ul_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
```


© The NASDAQ OMX Group, Inc. • 2013

```
}
```

4.69 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
}
```

4.70 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
}
```

4.71 WHOSE

```
struct whose {
   struct trading_code
   char[10] ex_client_s  // Client
   char[2] filler_2_s  // Filler
}
```

© The NASDAQ OMX Group, Inc. • 2013

© The NASDAQ OMX Group, Inc. • 2013

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_API (1)

```
struct cl_trade_api {
   struct trading_code
   struct series  // Named struct no: 50000
   struct account
   struct user_code
   struct countersign_code
   struct new_series
   struct party
   struct pos_account
   struct orig_series
   struct combo_series
   struct match_id
   INT32_T sequence_number_i  // Sequence Number
   INT32_T trade_number_i  // Trade Number
   INT32_T orig_trade_number_i  // Trade Number, Original
   INT32_T deal_price_i  // Price, Deal
   INT64_T trade_quantity_i  // Quantity, Trade
   INT32_T deal_number_i  // Deal Number
   UINT32_T global_deal_no_u  // Global Deal Number
   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
   QUAD_WORD order_number_u  // Order Number
   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 combo_deal_price_i  // Combo deal price
   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
   char[15] customer_info_s  // Customer, Information
   char[8] clearing_date_s  // Clearing Date
   char[32] passthrough_s  // Passthrough Information
   char[10] ex_client_s  // Client
   char[2] filler_2_s  // Filler
   char[2] reserved_2_s  // Reserved
   UINT8_T orig_trade_type_c  // Trade Type, Original
   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 open_close_c  // Open or Closed
```

```
    UINT8_T trade_type_c  // Type, Trade
    CHAR reserved_1_c  // Reserved
    UINT8_T trade_state_c  // Trade, State
    UINT8_T attention_c  // Attention
    UINT8_T account_type_c  // Account Type
    UINT8_T instigant_c  // Instigant
    UINT8_T cab_price_ind_c  // Cabinet Price Indicator
    CHAR ext_trade_fee_type_c  // External Trade, Fee Type
    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 commission_i  // Commission
    struct give_up_member  // Named struct no: 50002
    INT32_T give_up_number_i  // Give Up, Number
    UINT8_T give_up_state_c  // Give Up, State
    UINT8_T le_state_c  // Type, Legal Event
    UINT8_T instance_c  // Instance, Number
    UINT8_T trade_venue_c  // Trade venue
    UINT32_T big_attention_u  // Big Attention
}
```

5.2 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
   struct exchange_info_s  // Internally overlayed structure: OM_EXCHANGE_INFO
```


© The NASDAQ OMX Group, Inc. • 2013

```
   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
}
```

5.3 FX_TRADE_REPORT (7)

```
struct fx_trade_report {
   struct otc_trade_report
   struct fx {
      struct base_trade_report
      struct buy_currency  // Of type: SERIES ; Named struct no: 50000
      struct sell_currency  // Of type: SERIES ; Named struct no: 50000
      INT64_T buy_amount_q  // Buy Amount
      INT64_T sell_amount_q  // Sell Amount
      INT64_T exchange_rate_q  // Exchange rate
      char[120] buy_si_s  // Buy Settlement Instruction
      char[120] sell_si_s  // Sell Settlement Instruction
      char[16] method_dealt_s  // Method
      UINT8_T buy_use_ssi_c  // Special settlement instruction
      UINT8_T sell_use_ssi_c  // Sell use ssi
      UINT8_T settle_domestic_currency_c  // Settlement Domestic Currency
      UINT8_T settle_foreign_currency_c  // Settlement Foreign Currency
   }
}
```

5.4 CASH_TRADE_REPORT (8)

```
struct cash_trade_report {
   struct otc_trade_report
   struct cash {
      struct base_trade_report
      INT64_T amount_q  // Amount
      INT32_T interest_rate_i  // Interest Rate
      char[4] cash_type_s  // Cash Type
   }
}
```

5.5 AGREEMENT_TRADE_REPORT (9)

```
struct agreement_trade_report {
   struct otc_trade_report
```

© The NASDAQ OMX Group, Inc. • 2013

```
   struct agreement {
      struct base_trade_report
      char[24] agreement_type_s  // Agreement, Type
      char[8] agreement_date_s  // Date, Agreement
      char[24] agreement_version_s  // Agreement, Version
   }
}
```

5.6 SSI_TRADE_REPORT (10)

```
struct ssi_trade_report {
   struct otc_trade_report
   struct ssi {
      struct base_trade_report
      struct currency  // Of type: SERIES ; Named struct no: 50000
      char[120] settlement_instruction_s  // Settlement instruction
      UINT8_T instrument_level_c  // INSTRUMENT_LEVEL_C
      char[3] filler_3_s  // Filler
   }
}
```

5.7 FRA_TRADE_REPORT (11)

```
struct fra_trade_report {
   struct otc_trade_report
   struct fra  // Named struct no: 85
   struct float_rate_series  // Of type: SERIES ; Named struct no: 50000
   INT64_T fixed_consideration_q  // Fixed Consideration
   INT64_T float_consideration_q  // Float Consideration
   INT64_T pay_amount_q  // Pay Amount
   INT32_T float_interest_rate_i  // Float Interest Rate
}
```

5.8 EQUITY_TRADE_REPORT (12)

```
struct equity_trade_report {
   struct otc_trade_report
   struct equity {
      struct base_trade_report
      INT64_T quantity_q  // Quantity
      INT64_T consideration_q  // Consideration
      INT32_T deal_price_i  // Price, Deal
   }
}
```


© The NASDAQ OMX Group, Inc. • 2013

5.9 FI_TRADE_REPORT (13)

```
struct fi_trade_report {
   struct otc_trade_report
   struct fixed_income {
      struct base_trade_report
      INT64_T face_value_q  // Face Value
      INT32_T yield_i  // YIELD_I
      INT64_T consideration_q  // Consideration
   }
}
```

5.10 FI_REPO_TRADE_REPORT (14)

```
struct fi_repo_trade_report {
   struct otc_trade_report
   struct fi_repo {
      struct base_trade_report
      INT64_T face_value_q  // Face Value
      INT64_T consideration_q  // Consideration
      INT64_T unwind_consideration_q  // UNWIND_CONSIDERATION_Q
      INT32_T cash_rate_i  // CASH_RATE_I
      INT32_T margin_ratio_i  // Margin Ratio
      INT32_T yield_i  // YIELD_I
      char[8] unwind_settlement_date_s  // Unwind Settlement Date
      UINT8_T state_c  // State
      char[3] filler_3_s  // Filler
   }
   INT64_T init_consideration_q  // Initial consideration
   INT64_T init_face_value_q  // Initial face value
   char[8] effective_date_s  // Date, Effective
   char[8] delivery_unit_date_s  // DELIVERY_UNIT_DATE_S
   UINT8_T repo_category_c  // REPO_CATEGORY_C
   char[3] filler_3_s  // Filler
}
```

5.11 IR_SWAP_TRADE_REPORT (15)

```
struct ir_swap_trade_report {
   struct otc_trade_report
   struct ir_swap
   char[256] party_swap_condition_s  // Party swap condition
   UINT16_T flow_version_n  // Trade report version ; Of type:
TRADE_REPORT_VERSION_N
   char[8] delivery_unit_date_s  // DELIVERY_UNIT_DATE_S
   UINT8_T condition_confirmed_c  // CONDITION_CONFIRMED_C
   UINT8_T party_condition_confirmed_c  // Party Condition Confirmed
   UINT8_T termination_state_c  // Termination State
   char[3] filler_3_s  // Filler
```

© The NASDAQ OMX Group, Inc. • 2013

```
}
```

5.12 XCUR_SWAP_TRADE_REPORT (16)

```
struct xcur_swap_trade_report {
   struct otc_trade_report
   struct xcur_swap {
      struct base_trade_report
      struct upfront  // Of type: PAYMENT
      char[8] date_termination_s  // Date, Maturity
      char[8] principal_exchange_date_s  // Principal Exchange Date
      INT64_T exchange_rate_q  // Exchange rate
      char[256] swap_condition_s  // Swap condition
      char[5] first_holiday_id_s  // First State Holiday ID
      char[3] filler_3_s  // Filler
      char[5] second_holiday_id_s  // Second State Holiday ID
      UINT8_T apply_holiday_c  // State holiday applied, Yes/No
      UINT8_T principal_exchange_c  // Principal Exchange
      CHAR filler_1_s  // Filler
      struct member_pay {  // Of type: XCUR_SWAP_LEG
         struct currency  // Of type: SERIES ; Named struct no: 50000
         INT64_T notional_amount_q  // Notional amount
         INT32_T fixed_interest_rate_i  // Fixed Interest Rate
         struct float_rate_index  // Of type: SERIES ; Named struct no: 50000
         INT32_T spread_i  // Spread
         INT32_T init_interest_rate_i  // Init Interest Rate
         char[8] first_rollover_date_s  // First Rollover Date
         char[120] settlement_instruction_s  // Settlement instruction
         char[24] cash_account_s  // Account, Cash
         UINT8_T use_ssi_c  // Use SSI
         UINT8_T payment_settlement_c  // Payment settled by CSD Yes/ No
         UINT8_T day_count_conv_c  // Day Count Convention
         UINT8_T rollover_period_c  // Rollover Period
         UINT8_T rollover_day_c  // Rollover Day
         UINT8_T business_day_conv_c  // BUSINESS_DAY_CONV_C
         UINT8_T rate_reset_c  // Rate Reset
         UINT8_T reset_days_c  // Reset Days
         UINT8_T payment_set_c  // Payment Set
         UINT8_T fixed_or_float_c  // Fixed or Float
         char[2] filler_2_s  // Filler
         struct party_pay  // Of type: PARTY
      }
      struct counterparty_pay {  // Of type: XCUR_SWAP_LEG
         struct currency  // Of type: SERIES ; Named struct no: 50000
         INT64_T notional_amount_q  // Notional amount
         INT32_T fixed_interest_rate_i  // Fixed Interest Rate
         struct float_rate_index  // Of type: SERIES ; Named struct no: 50000
         INT32_T spread_i  // Spread
         INT32_T init_interest_rate_i  // Init Interest Rate
         char[8] first_rollover_date_s  // First Rollover Date
         char[120] settlement_instruction_s  // Settlement instruction
         char[24] cash_account_s  // Account, Cash
         UINT8_T use_ssi_c  // Use SSI
         UINT8_T payment_settlement_c  // Payment settled by CSD Yes/ No
```


© The NASDAQ OMX Group, Inc. • 2013

```
            UINT8_T day_count_conv_c  // Day Count Convention
            UINT8_T rollover_period_c  // Rollover Period
            UINT8_T rollover_day_c  // Rollover Day
            UINT8_T business_day_conv_c  // BUSINESS_DAY_CONV_C
            UINT8_T rate_reset_c  // Rate Reset
            UINT8_T reset_days_c  // Reset Days
            UINT8_T payment_set_c  // Payment Set
            UINT8_T fixed_or_float_c  // Fixed or Float
            char[2] filler_2_s  // Filler
            struct party_pay  // Of type: PARTY
         }
      }
      char[256] party_swap_condition_s  // Party swap condition
      UINT16_T flow_version_n  // Trade report version ; Of type:
   TRADE_REPORT_VERSION_N
      char[8] delivery_unit_date_s  // DELIVERY_UNIT_DATE_S
      UINT8_T condition_confirmed_c  // CONDITION_CONFIRMED_C
      UINT8_T party_condition_confirmed_c  // Party Condition Confirmed
      UINT8_T termination_state_c  // Termination State
      char[3] filler_3_s  // Filler
   }
```

5.13 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
```

© The NASDAQ OMX Group, Inc. • 2013

```
   UINT8_T instigant_c  // Instigant
   UINT8_T cab_price_ind_c  // Cabinet Price Indicator
}
```

5.14 CASH_TRANSFER_GROUP_OTC (22)

```
struct cash_transfer_group_otc {
   struct otc_trade_report
   struct cash_transfer_group {
      struct base_trade_report
      char[12] cash_transfer_group_s  // Cash transfer group
      char[12] cash_transfer_code_s  // Cash transfer code
      char[10] party_account_id_s  // Cash transfer group
      struct proxy_account  // Of type: ACCOUNT
      char[2] filler_2_s  // Filler
   }
}
```

5.15 CASH_TRANSFER_TRADE_REPORT (23)

```
struct cash_transfer_trade_report {
   struct otc_trade_report
   struct cash_transfer {
      struct base_trade_report
      INT64_T amount_q  // Amount
      char[12] cash_transfer_code_s  // Cash transfer code
      char[10] party_account_id_s  // Cash transfer group
      char[24] transfer_cash_account_s  // Transfer Account, Cash
      struct proxy_account  // Of type: ACCOUNT
      char[2] filler_2_s  // Filler
   }
}
```

5.16 NETTING_SWAP (45)

```
struct netting_swap {
   struct account
   struct party_account  // Of type: ACCOUNT
   struct series  // Named struct no: 50000
   struct currency  // Of type: SERIES ; Named struct no: 50000
   UINT64_T trade_report_nbr_q  // Trade report number
   INT64_T notional_amount_q  // Notional amount
   INT64_T payment_notional_amount_q  // Payment notional amount
   INT64_T payment_q  // Payment
   INT32_T deal_number_i  // Deal Number
   INT32_T interest_rate_i  // Interest Rate
   UINT32_T delivery_unit_u  // Delivery Unit
   UINT32_T netting_req_nbr_u  // Netting request number
   char[32] passthrough_s  // Passthrough Information
```


© The NASDAQ OMX Group, Inc. • 2013

```
   char[8] termination_date_s  // Date ; Of type: YYYYMMDD_S
   char[8] payment_date_s  // Date ; Of type: YYYYMMDD_S
   char[8] settlement_date_s  // Date, Settlement
   UINT8_T fixed_or_float_c  // Fixed or Float
   char[3] filler_3_s  // Filler
}
```

5.17 NETTING_FRA (46)

```
struct netting_fra {
   struct account
   struct party_account  // Of type: ACCOUNT
   struct series  // Named struct no: 50000
   struct currency  // Of type: SERIES ; Named struct no: 50000
   UINT64_T trade_report_nbr_q  // Trade report number
   INT64_T consideration_q  // Consideration
   INT64_T pay_amount_q  // Pay Amount
   INT64_T notional_amount_q  // Notional amount
   INT32_T deal_number_i  // Deal Number
   INT32_T interest_rate_i  // Interest Rate
   UINT32_T delivery_unit_u  // Delivery Unit
   UINT32_T netting_req_nbr_u  // Netting request number
   char[32] passthrough_s  // Passthrough Information
   char[8] termination_date_s  // Date ; Of type: YYYYMMDD_S
   char[8] effective_date_s  // Date, Effective
   char[8] settlement_date_s  // Date, Settlement
   CHAR buy_or_sell_c  // Buy or Sell
   UINT8_T fixed_or_float_c  // Fixed or Float
   char[2] filler_2_s  // Filler
}
```

5.18 NETTING_FX (47)

```
struct netting_fx {
   struct account
   struct party_account  // Of type: ACCOUNT
   struct series  // Named struct no: 50000
   struct currency  // Of type: SERIES ; Named struct no: 50000
   UINT64_T trade_report_nbr_q  // Trade report number
   INT64_T pay_amount_q  // Pay Amount
   INT64_T amount_q  // Amount
   INT32_T deal_number_i  // Deal Number
   INT64_T exchange_rate_q  // Exchange rate
   UINT32_T delivery_unit_u  // Delivery Unit
   UINT32_T netting_req_nbr_u  // Netting request number
   char[32] passthrough_s  // Passthrough Information
   char[8] payment_date_s  // Date ; Of type: YYYYMMDD_S
   char[8] settlement_date_s  // Date, Settlement
   CHAR buy_or_sell_c  // Buy or Sell
   char[3] filler_3_s  // Filler
}
```

© The NASDAQ OMX Group, Inc. • 2013

5.19 DC_HOLD_DEAL_EXTERNAL (63)

```
struct dc_hold_deal_external {
   struct series  // Named struct no: 50000
   char[8] created_date_s  // Date, Created
   UINT8_T dc_deal_state_c  // State, Deal
   UINT8_T init_dc_deal_state_c  // State, Deal ; Of type: DC_DEAL_STATE_C
   char[2] filler_2_s  // Filler
}
```

5.20 DC_HOLD_TRADE_EXTERNAL (64)

```
struct dc_hold_trade_external {
   QUAD_WORD order_number_u  // Order Number
   UINT8_T bought_or_sold_c  // Bought or Sold
   char[3] filler_3_s  // Filler
}
```

5.21 OTC_CASH_FLOW_BASE (65)

```
struct otc_cash_flow_base {
   struct account
   struct series  // Named struct no: 50000
   char[40] description_s  // Description
   INT32_T sequence_number_i  // Sequence Number
}
```

5.22 OTC_CASH_FLOW_INFO (66)

```
struct otc_cash_flow_info {
   UINT64_T trade_report_nbr_q  // Trade report number
   INT64_T notional_amount_q  // Notional amount
   UINT64_T consideration_u  // Consideration
   INT32_T interest_rate_i  // Interest Rate
   INT32_T spread_i  // Spread
   UINT16_T dec_in_nominal_n  // Decimals, Nominal
   UINT16_T dec_in_consideration_n  // DEC_IN_CONSIDERATION_N
   UINT16_T dec_in_rate_n  // Decimals, Rate
   UINT16_T dec_in_spread_n  // Decimals, Rate ; Of type: DEC_IN_RATE_N
   UINT16_T days_in_period_n  // Days in Period
   UINT16_T days_per_year_n  // DAYS_PER_YEAR_N
   char[32] passthrough_s  // Passthrough Information
   char[8] start_date_s  // Date, Start
   char[8] end_date_s  // Date, End
   char[8] payment_date_s  // Date, Payment
   char[3] currency_s  // Currency
   UINT8_T fixed_or_float_c  // Fixed or Float
```


© The NASDAQ OMX Group, Inc. • 2013

```
   UINT8_T pay_or_receive_c  // Deliver/Pay or Receive
   UINT8_T otc_cash_flow_type_c  // OTC cash flow type
   UINT8_T business_day_conv_c  // BUSINESS_DAY_CONV_C
   UINT8_T basis_swap_relation_c  // The relation of cash flows
   char[8] reset_date_s  // Date, Reset
   INT32_T fixing_value_i  // Fixing Value
   char[8] trade_clearing_date  // Clearing Date ; Of type: CLEARING_DATE_S
   INT32_T sequence_number_i  // Sequence Number
   INT64_T accumulated_consideration_q  // Consideration, Accumulated
   INT64_T estimated_accumulated_consideration_q  // Estimated Consideration,
 Accumulated
   char[8] estimated_consideration_date_s  // Estimated Consideration Date
   CHAR is_flow_reset_c  // BOOLEAN ; Of type: BOOLEAN
   char[3] filler_3_s  // Filler
}
```

5.23 CL_TRADE_TRADE_REPORT_API (67)

```
struct cl_trade_trade_report_api {
   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[2] filler_2_s  // Filler
}
```

5.24 CL_TRADE_FIXED_INCOME_API (68)

```
struct cl_trade_fixed_income_api {
   INT32_T corresponding_yield_price_i  // Corresponding Yield/Price
   INT64_T consideration_q  // Consideration
   UINT16_T deferred_time_n  // Deferred Publication time
   char[8] settlement_date_s  // Date, Settlement
   char[2] filler_2_s  // Filler
}
```

5.25 CL_TRADE_CANCEL_TRADE_API (70)

```
struct cl_trade_cancel_trade_api {
   UINT32_T trade_reject_sec_u  // Trade Reject, Seconds
}
```

5.26 IR_SWAP_FLOW_FOR_SIM (75)

```
struct ir_swap_flow_for_sim {
   struct series  // Named struct no: 50000
   char[8] effective_date_s  // Date, Effective
   char[8] date_termination_s  // Date, Maturity
   INT64_T notional_amount_q  // Notional amount
```

© The NASDAQ OMX Group, Inc. • 2013

```
   char[5] first_holiday_id_s  // First State Holiday ID
   UINT8_T rate_reset_c  // Rate Reset
   UINT8_T reset_days_c  // Reset Days
   UINT8_T payment_set_c  // Payment Set
   char[5] second_holiday_id_s  // Second State Holiday ID
   UINT8_T business_day_conv_c  // BUSINESS_DAY_CONV_C
   char[2] filler_2_s  // Filler
   struct member_pay  // Of type: IR_SWAP_LEG
   struct counterparty_pay  // Of type: IR_SWAP_LEG
}
```

5.27 CL_ACCOUNT_BASE_API (81)

```
struct cl_account_base_api {
   struct account
   struct countersign
   struct prop_trade_account
   struct prop_settlement_account {
      char[2] country_id_s  // Name, Country
      char[5] ex_customer_s  // Customer, Identity
      char[10] account_id_s  // Account, Identity
   }
   struct prop_pos_account
   struct prop_margin_account
   struct sink_account
   struct prop_origin_account
   struct prop_call_account
   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[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 acc_allow_nov_c  // Novation Allowed
   struct prop_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
}
```


© The NASDAQ OMX Group, Inc. • 2013

5.28 CL_ACCOUNT_RISK_ATTRIBUTE_API (82)

```
struct cl_account_risk_attribute_api {
   INT32_T rank_class_i  // Risk Ranking Class
   char[3] risk_currency_s  // Currency, Risk
   UINT8_T risk_cur_conv_c  // Risk, Currency Conversion
   UINT8_T risk_margin_net_c  // Risk, Margin Net
   char[3] margin_class_s  // Margin class
   char[12] risk_scale_s  // Risk scale
}
```

5.29 OTC_CLEARING_INFO (83)

```
struct otc_clearing_info {
   struct position_account  // Of type: ACCOUNT
   char[8] clearing_date_s  // Clearing Date
   char[8] orig_clearing_date_s  // Clearing Date, Original
}
```

5.30 FRA (85)

```
struct fra {
   struct base_trade_report
   struct float_rate_index  // Of type: SERIES ; Named struct no: 50000
   INT64_T notional_amount_q  // Notional amount
   INT32_T fixed_interest_rate_i  // Fixed Interest Rate
   char[8] float_rate_fixing_date_s  // Float Rate Fixing Date
   char[8] date_termination_s  // Date, Maturity
   UINT8_T day_count_conv_c  // Day Count Convention
   char[3] filler_3_s  // Filler
}
```

5.31 CL_ACCOUNT_COLLATERAL_ATTRIBUTE_API (86)

```
struct cl_account_collateral_attribute_api {
   struct dd_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
   }
   char[3] base_cur_id_s  // Currency, Base
   UINT8_T account_collateral_handling_c  // Account Collateral Handling
}
```

© The NASDAQ OMX Group, Inc. • 2013

5.32 CL_ACCOUNT_BASE_COLLATERAL_API (94)

```
struct cl_account_base_collateral_api {
   struct base_collateral_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
   }
   INT32_T bc_adjustment_factor_i  // Base collateral requirement adjustment
factor.
}
```

5.33 CL_OTC_OPERATION_INFO (95)

```
struct cl_otc_operation_info {
   UINT8_T cl_otc_trade_operation_c  // CL_OTC_Trade_Operation
   UINT8_T le_state_c  // Type, Legal Event
   char[2] filler_2_s  // Filler
   INT32_T orig_deal_number_i  // Deal Number, Original
   struct series  // Named struct no: 50000
   INT32_T sequence_number_i  // 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
   INT32_T tx_status_i  // TX_STATUS_I
   struct trading_code
   char[8] business_date_s  // Date, Business
}
```

5.34 CL_OTC_TRADE_OPERATION (96)

```
struct cl_otc_trade_operation {
   struct account
   struct pos_account
   INT32_T orig_trade_number_i  // Trade Number, Original
   UINT8_T trade_type_c  // Type, Trade
   UINT8_T trade_report_reason_c  // Trade report reason
   UINT8_T buy_sell_c  // BUY_SELL_C
   CHAR filler_1_s  // Filler
   INT64_T trade_quantity_i  // Quantity, Trade
   INT64_T total_surplus_deficit_q  // Total surplus deficit
}
```


© The NASDAQ OMX Group, Inc. • 2013

5.35 CL_ACCOUNT_INTRADAY_FUNDING_API (97)

```
struct cl_account_intraday_funding_api {
   struct intraday_funding_account  // Of type: ACCOUNT
}
```

5.36 COLLATERAL_INFO (18000)

```
struct collateral_info {
   struct trading_code
   struct user_code
   UINT64_T collateral_nbr_q  // Collateral Number
   UINT16_T version_n  // Collateral position version (defined for this struct)
   char[8] timestamp_date_s  // Timestamp, Date
   char[6] timestamp_time_s  // Timestamp, Time
   char[8] asof_date_s  // Date, As Of
   char[6] asof_time_s  // Time, As Of
   char[32] name_s  // NT User name (defined for this struct)
   UINT8_T collateral_type_c  // Collateral types
   UINT8_T state_c  // State
   INT64_T preliminary_amount_q  // Preliminary Collateral Balance or Holding
 adjusted for not yet settled collateral withdraw requests.
   INT64_T preliminary_amount_ca_adjusted_q  // Preliminary Collateral Balance
 or Holding after corp action adjustment.
   char[12] ext_acc_registrar_s  // External Account Registrar
   char[15] ext_acc_controller_s  // External Account Controller
   char[34] ext_acc_id_s  // External Account ID
   CHAR filler_1_s  // Filler
   UINT16_T dec_in_amount_n  // Decimals, Amount
}
```

5.37 GUARANTEE (18001)

```
struct guarantee {
   struct collateral_base {
      struct account
      struct series  // Named struct no: 50000
      INT64_T amount_q  // Collateral amount or quantity.Decimals according
to dec_in_amount_n. (defined for this struct)
      char[32] passthrough_s  // Passthrough Information
      char[8] effective_date_s  // Date, Effective
      char[32] name_s  // NT User name (defined for this struct)
      char[8] effective_until_s  // Effective Until
   }
   UINT8_T guarantee_type_c  // Guarantee Type
   char[3] filler_3_s  // Filler
}
```

© The NASDAQ OMX Group, Inc. • 2013

5.38 MEMBER_DEPOSIT (18002)

```
struct member_deposit {
   struct collateral_base {
      struct account
      struct series  // Named struct no: 50000
      INT64_T amount_q  // Collateral amount or quantity.Decimals according
to dec_in_amount_n. (defined for this struct)
      char[32] passthrough_s  // Passthrough Information
      char[8] effective_date_s  // Date, Effective
      char[32] name_s  // NT User name (defined for this struct)
      char[8] effective_until_s  // Effective Until
   }
   UINT8_T member_deposit_type_c  // Member Deposit Type
   UINT8_T fund_type_c  // Fund Type
   char[2] filler_2_s  // Filler
}
```

5.39 CASH_COLLATERAL (18003)

```
struct cash_collateral {
   struct collateral_base {
      struct account
      struct series  // Named struct no: 50000
      INT64_T amount_q  // Collateral amount or quantity.Decimals according
to dec_in_amount_n. (defined for this struct)
      char[32] passthrough_s  // Passthrough Information
      char[8] effective_date_s  // Date, Effective
      char[32] name_s  // NT User name (defined for this struct)
      char[8] effective_until_s  // Effective Until
   }
}
```

5.40 SECURITY (18009)

```
struct security {
   struct collateral_base {
      struct account
      struct series  // Named struct no: 50000
      INT64_T amount_q  // Collateral amount or quantity.Decimals according
to dec_in_amount_n. (defined for this struct)
      char[32] passthrough_s  // Passthrough Information
      char[8] effective_date_s  // Date, Effective
      char[32] name_s  // NT User name (defined for this struct)
      char[8] effective_until_s  // Effective Until
   }
   UINT8_T security_type_c  // Security Type
   char[3] filler_3_s  // Filler
}
```


© The NASDAQ OMX Group, Inc. • 2013

5.41 DEPOSIT_WITHDRAW_COLLATERAL (18022)

```
struct deposit_withdraw_collateral {
   INT64_T amount_q  // Amount
   char[12] isin_code_s  // ISIN Code
   char[12] ext_acc_registrar_s  // External Account Registrar
   char[15] ext_acc_controller_s  // External Account Controller
   char[34] ext_acc_id_s  // External Account ID
   char[32] passthrough_s  // Passthrough Information
   char[16] instr_ref_s  // SWIFT reference.
   char[16] cancel_ref_s  // SWIFT reference.
   char[34] csd_code_s  // Code, CSD
   char[80] reason_s  // Reason
   char[3] currency_s  // Currency
   UINT16_T dec_in_amount_n  // Decimals, Amount
   UINT8_T collateral_transaction_type_c  // Collateral transaction type
   UINT8_T collateral_transaction_state_c  // Collateral transaction state
   char[2] filler_2_s  // Filler
}
```

5.42 SEQUENCE_NUMBER_INFO (18023)

```
struct sequence_number_info {
   INT32_T sequence_number_n  // Sequence Number
}
```

5.43 COLLATERAL_TRANSACTION_INFO (18024)

```
struct collateral_transaction_info {
   struct series  // Named struct no: 50000
   struct collateral_account  // Of type: ACCOUNT
   UINT64_T collateral_transaction_nbr_q  // Collateral Transaction Number
   UINT32_T request_nbr_u  // Request 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
   UINT16_T version_n  // Version
   UINT8_T is_direct_debit_c  // Is Direct Debit
   UINT8_T ext_confirm_c  // Is externally confirmed
   char[8] valuation_date_s  // Valuation Date
   UINT8_T collateral_type_c  // Collateral types
   char[3] filler_3_s  // Filler
}
```

© The NASDAQ OMX Group, Inc. • 2013

5.44 COLL_VAL_PER_SERIES_BASE_CUR (18025)

```
struct coll_val_per_series_base_cur {
   struct base_currency  // Of type: CURRENCY
   INT64_T coll_value_base_curr_before_limit_adjust_q  // Collateral Value ; Of type: COLLATERAL_VALUE_Q
   INT64_T coll_value_base_curr_after_limit_adjust_q  // Collateral Value ; Of type: COLLATERAL_VALUE_Q
}
```

5.45 COLL_VAL_PER_SERIES_RISK_CUR (18026)

```
struct coll_val_per_series_risk_cur {
   struct risk_currency  // Of type: CURRENCY
   struct coll_value_currency  // Of type: CURRENCY
   INT64_T collateral_value_q  // Collateral Value
}
```

5.46 COLL_VAL_PER_VAL_GROUP_TSN (18027)

```
struct coll_val_per_val_group_tsn {
   struct margin_account
   struct base_currency  // Of type: CURRENCY
   INT64_T coll_value_base_curr_before_limit_ajust_q  // Collateral Value ; Of type: COLLATERAL_VALUE_Q
   INT64_T coll_value_base_curr_after_limit_adjust_q  // Collateral Value ; Of type: COLLATERAL_VALUE_Q
   INT32_T group_limit_i  // Valuation group limit
   INT32_T actual_group_percentage_i  // Actual group percentage
   char[12] vag_id_s  // Valuation Group Identity
   UINT16_T dec_in_actual_group_percentage_n  // Decimals, Percentage
   char[2] filler_2_s  // Filler
}
```

5.47 COLLATERAL_INFORMATION_BASE (18028)

```
struct collateral_information_base {
   struct margin_account
   struct currency
   struct risk_currency  // Of type: CURRENCY
   INT64_T initial_margin_req_q  // Initial margin requirement.
   INT64_T variation_margin_req_q  // Variation margin requirement.
   INT64_T contingent_variation_margin_req_q  // Contingent variation margin requirement.
   INT64_T margin_maintenance_q  // Margin Maintenance
   INT64_T margin_extraordinary_q  // Margin Extraordinary
   INT64_T margin_total_q  // Margin Total
```


© The NASDAQ OMX Group, Inc. • 2013

```
   INT64_T collateral_guarantee_q  // Collateral Guarantee
   INT64_T collateral_cash_q  // Collateral Cash
   INT64_T collateral_security_q  // Security, Collateral
   INT64_T total_surplus_deficit_q  // Total surplus deficit
   INT64_T total_margin_req_q  // TOTAL_MARGIN_REQ_Q
   UINT8_T excluded_due_to_idmc_c  // Excluded due to IDMC
   char[3] filler_3_s  // Filler
}
```

5.48 COLLATERAL_INFORMATION_DEFAULT_FUND (18029)

```
struct collateral_information_default_fund {
   INT64_T margin_mutual_fund_q  // Margin Mutual Fund
   INT64_T margin_default_fund_q  // Margin Default Fund
}
```

5.49 COLLATERAL_INFORMATION_PAYMENT_DELIVERY (18030)

```
struct collateral_information_payment_delivery {
   INT64_T payment_margin_valuation_date_q  // Payment margin valuation date.
   INT64_T payment_margin_future_date_q  // Payment margin future date.
   INT64_T delivery_margin_valuation_date_q  // Delivery margin valuation date.
   INT64_T payment_margin_overdue_q  // Overdue payment margin.
   INT64_T delivery_margin_overdue_q  // Overdue delivery margin.
}
```

5.50 COLLATERAL_INFORMATION_NPC (18031)

```
struct collateral_information_npc {
   INT64_T cash_requirement_q  // Cash Requirement
   INT64_T settlement_requirement_q  // Settlement Requirement
   INT64_T coll_cash_usage_other_curr_q  // Collateral cash usage other currency
   INT64_T balance_guarantee_q  // Balance Guarantee
   INT64_T balance_account_q  // Balance Account
   INT64_T balance_security_q  // Security, Balance
   INT64_T total_req_balance_account_q  // Balance, Total Required
}
```

5.51 BASE_CURRENCY_CONVERSION (18032)

```
struct base_currency_conversion {
   struct margin_account
   struct currency
   struct base_currency  // Of type: CURRENCY
   INT64_T margin_total_q  // Margin Total
```

© The NASDAQ OMX Group, Inc. • 2013

```
   INT64_T total_collateral_value_q  // Total Collateral Value
   INT64_T total_surplus_deficit_q  // Total surplus deficit
   INT64_T total_surplus_deficit_base_cur_q  // Total surplus deficit in base
 currency
   INT64_T total_surplus_deficit_base_cur_after_fx_haircut_q  // Total surplus
 deficit in base currency
   INT64_T ex_rate_q  // Exchange Rate, Collateral
   INT32_T fx_percentage_after_haircut_i  // Haircut ; Of type: HAIRCUT_I
   UINT16_T dec_in_rate_n  // Decimals, Rate
   UINT8_T excluded_due_to_idmc_c  // Excluded due to IDMC
   CHAR filler_1_s  // Filler
}
```

5.52 COLLATERAL_EVALUATION_RUN_INFO (18033)

```
struct collateral_evaluation_run_info {
   struct account
   UINT32_T request_nbr_u  // Request number
   UINT32_T margin_sequence_nbr_u  // Unique identifier for a margin calculation
 batch run.
   char[12] clh_id_s  // Clearinghouse
   char[8] valuation_date_s  // Valuation Date
   char[8] created_date_s  // Date, Created
   char[6] created_time_s  // Time, Created
   char[8] margin_date_s  // Margin Date
   UINT8_T is_intraday_c  // Intraday, Is
   UINT8_T collateral_state_c  // Collateral State
   UINT8_T is_final_c  // Final, Is
   UINT8_T collateral_evaluation_type_c  // Collateral evaluation type
   UINT8_T create_direct_debit_c  // Create Direct Debit
   CHAR filler_1_s  // Filler
}
```

5.53 BASE_CURRENCY_CONVERSION_GRAND_TOTAL (18035)

```
struct base_currency_conversion_grand_total {
   struct margin_account
   struct base_currency  // Of type: CURRENCY
   INT64_T grand_total_surplus_deficit_base_cur_q  // Grand total surplus
deficit in base currency
   INT64_T grand_total_surplus_deficit_base_cur_after_fx_haircut_q  // Grand
total surplus deficit in base currency
   UINT8_T excluded_due_to_idmc_c  // Excluded due to IDMC
   char[3] filler_3_s  // Filler
}
```


© The NASDAQ OMX Group, Inc. • 2013

5.54 COLL_VAL_PER_SERIES (18036)

```
struct coll_val_per_series {
   struct collateral_account  // Of type: ACCOUNT
   struct margin_account
   struct series  // Named struct no: 50000
   struct currency
   INT64_T collateral_amount_q  // Collateral Amount/Quantity (defined for
this struct)
   INT64_T market_value_q  // Market Value
   INT64_T coll_value_ins_cur_before_limit_adjust_q  // Collateral Value ; Of
 type: COLLATERAL_VALUE_Q
   INT64_T coll_value_ins_cur_after_limit_adjust_q  // Collateral Value ; Of
type: COLLATERAL_VALUE_Q
   INT64_T ex_rate_q  // Exchange Rate, Collateral
   INT32_T collateral_price_i  // Fixing Value ; Of type: FIXING_VALUE_I
   INT32_T percentage_after_haircut_i  // Haircut ; Of type: HAIRCUT_I
   char[12] vag_id_s  // Valuation Group Identity
   UINT16_T dec_in_rate_n  // Decimals, Rate
   UINT16_T dec_in_amount_n  // Decimals, Amount
   UINT16_T dec_in_collateral_price_n  // Decimals, Collateral price
   char[2] filler_2_s  // Filler
}
```

5.55 RUN_INFO (18037)

```
struct run_info {
   UINT32_T request_nbr_u  // Request number
  UINT32_T margin_sequence_nbr_u  // Unique identifier for a margin calculation
 batch run.
   char[8] valuation_date_s  // Valuation Date
   char[8] created_date_s  // Date, Created
   char[6] created_time_s  // Time, Created
   char[8] margin_date_s  // Margin Date
   char[6] margin_time_s  // Margin Time
   char[12] clh_id_s  // Clearinghouse
   UINT8_T collateral_evaluation_type_c  // Collateral evaluation type
   char[3] filler_3_s  // Filler
}
```

5.56 CORPORATE_ACTION_INFO (18038)

```
struct corporate_action_info {
   char[16] corp_action_ref_s  // Corporate action SWIFT reference.
   char[16] corp_event_ref_s  // Corporate action event SWIFT reference.
}
```

© The NASDAQ OMX Group, Inc. • 2013

5.57 BASE_CALL (18043)

```
struct base_call {
   INT64_T base_collateral_req_q  // Base collateral requirement
   INT64_T adjusted_base_collateral_req_q  // Adjusted base collateral
requirement
}
```

5.58 DEFICIT_TO_COVER (18049)

```
struct deficit_to_cover {
   INT64_T deficit_to_cover_q  // Deficit to cover
}
```

5.59 PAYNOTE_INFO_DETAIL (19001)

```
struct paynote_info_detail {
   INT64_T total_amount_q  // Total amount
   INT32_T pay_note_number_i  // Pay note number
   INT32_T delivery_number_i  // Delivery, Number
   UINT32_T instruction_nbr_u  // Instruction number
   char[8] settlement_date_s  // Date, Settlement
   char[8] settlement_instr_date_s  // Date, Settlement instruction
   char[3] currency_s  // Currency
   CHAR filler_1_s  // Filler
   struct party
   char[15] note_name_s  // Note name
   char[6] payment_status_s  // Payment status
   char[8] modified_date_s  // Date, Modified
   char[6] modified_time_s  // Time, Modified
   char[12] user_code_s  // User Code
   CHAR reserved_1_s  // Reserved
   char[8] status_date_s  // Date ; Of type: DATE_S
   char[8] payment_date_s  // Date ; Of type: DATE_S
}
```

5.60 PAYNOTE_INFO_DETAIL_ITEM (19002)

```
struct paynote_info_detail_item {
   struct series  // Named struct no: 50000
   char[8] clearing_date_s  // Clearing Date
   struct account
   INT64_T amount_q  // Amount
   INT32_T event_type_i  // Stimuli Event
   INT32_T class_no_i  // Class Number
   char[15] settlement_product_s  // Settlement product
   UINT8_T source_id_c  // Source for paynote data
```


© The NASDAQ OMX Group, Inc. • 2013

```
   char[60] invc_text_s  // Invoice Text
   UINT8_T acnt_account_type_c  // Account Type for accounting
   char[3] filler_3_s  // Filler
   INT32_T settlement_note_number_i  // Delivery, Number ; Of type:
DELIVERY_NUMBER_I
}
```

5.61 YIELD_CURVE_NAMES (20000)

```
struct yield_curve_names {
   char[12] crv_id_s  // Curve Id
   char[12] filler_12_s  // FILLER_12_S
   INT16_T min_num_nodes_n  // Mininum number of Nodes
   INT16_T min_num_days_n  // Minimum number of days
   UINT16_T dec_in_yield_n  // Decimals, Yield
   UINT16_T dec_in_discount_factor_n  // Decimals, Factors
   char[2] country_id_s  // Name, Country
   char[3] currency_s  // Currency
   UINT8_T curv_construction_method_c  // Curve Construction Method
   UINT8_T day_count_conv_c  // Day Count Convention
   UINT8_T discount_method_c  // Discount Method
   char[12] discount_crv_id_s  // Discount curve
   UINT8_T crv_type_c  // Curve type
   UINT8_T crv_tenor_c  // Curve tenor
   char[2] filler_2_s  // Filler
}
```

5.62 MARG_CALC_RUNS (21000)

```
struct marg_calc_runs {
   char[12] clh_id_s  // Clearinghouse
   char[8] valuation_date_s  // Valuation Date
   char[8] start_date_s  // Date, Start
   char[6] start_time_s  // Time, Start
   UINT8_T incl_t_plus_one_prices_c  // Include T+1 Prices
   UINT8_T incl_t_plus_one_positions_c  // Include T+1 Positions
   INT32_T sequence_number_n  // Sequence Number
   UINT8_T run_type_c  // Run Type
   char[3] filler_3_s  // Filler
}
```

5.63 STRESS_FACTORS_FOR_YIELD_CURVE (21001)

```
struct stress_factors_for_yield_curve {
   char[12] stress_crv_id_s  // Stress Curve Id
   char[12] crv_id_s  // Curve Id
   INT64_T stress_level_pc1_up_q  // Stress Level, PC1 up
   INT64_T stress_level_pc1_down_q  // Stress Level, PC1 down
   INT64_T stress_level_pc2_up_q  // Stress Level, PC2 up
```

© The NASDAQ OMX Group, Inc. • 2013

```
    INT64_T stress_level_pc2_down_q  // Stress Level, PC2 down
    INT64_T stress_level_pc3_up_q  // Stress Level, PC3 up
    INT64_T stress_level_pc3_down_q  // Stress Level, PC3 down
    char[12] ccc_id_s  // Curve Correlation Cube
}
```

5.64 PRINCIPAL_FACTORS (21002)

```
struct principal_factors {
    INT64_T pc1_q  // Principal Component, First
    INT64_T pc2_q  // Principal Component, Second
    INT64_T pc3_q  // Principal Component, Third
    INT16_T pc_years_n  // Principal component, Years
    char[2] filler_2_s  // Filler
}
```

5.65 TRADE_NODE_VALUES (21010)

```
struct trade_node_values {
    INT32_T point_no_pc1_i  // Point number for PC1
    INT32_T point_no_pc2_i  // Point number for PC2
    INT32_T point_no_pc3_i  // Point number for PC3
    INT32_T value_low_i  // Value, low
    INT32_T value_middle_i  // Value, middle
    INT32_T value_high_i  // Value, high
}
```

5.66 INSTRUMENT_CURVE_NODE_VALUES (21011)

```
struct instrument_curve_node_values {
    INT32_T point_no_pc1_i  // Point number for PC1
    INT32_T point_no_pc2_i  // Point number for PC2
    INT32_T point_no_pc3_i  // Point number for PC3
    UINT32_T long_low_i  // Long, Low
    UINT32_T short_low_i  // Short, Low
    UINT32_T long_middle_i  // Long, Middle
    UINT32_T short_middle_i  // Short, Middle
    UINT32_T long_high_i  // Long, High
    UINT32_T short_high_i  // Short, High
    INT32_T discount_long_i  // Discount, long
    INT32_T discount_short_i  // Discount, short
}
```

5.67 MARGIN_CLASS_CURVE (21012)

```
struct margin_class_curve {
    struct series  // Named struct no: 50000
```


© The NASDAQ OMX Group, Inc. • 2013

```
   INT32_T dec_in_margin_value_i  // Decimals, Margin value
   char[12] primary_crv_id_s  // Primary Curve Id
   char[12] primary_ccc_id_s  // Primary Curve Correlation Cube
   char[12] secondary_crv_id_s  // Secondary Curve Id
   char[12] secondary_ccc_id_s  // Secondary Curve Correlation Cube
   UINT16_T dec_in_discount_factor_n  // Decimals, Factors
   char[8] closing_date_s  // Date, Closing
   char[3] margin_class_s  // Margin class
   char[3] filler_3_s  // Filler
}
```

5.68 CRVCORR_PARAM (21013)

```
struct crvcorr_param {
   char[12] ccc_id_s  // Curve Correlation Cube
   char[12] upper_ccc_id_s  // Upper Curve Correlation Cube
   UINT16_T overlap_pc1_n  // Overlap, PC1
   UINT16_T overlap_pc2_n  // Overlap, PC2
   UINT16_T overlap_pc3_n  // Overlap, PC3
   char[3] currency_s  // Currency
   char[3] margin_class_s  // Margin class
   UINT8_T volatility_corr_rm_c  // Volatility correlation
   char[3] filler_3_s  // Filler
}
```

5.69 TRADE_RISK_VALUES (21038)

```
struct trade_risk_values {
   struct series  // Named struct no: 50000
   struct account
   INT64_T trade_number_q  // Trade number
}
```

5.70 TRADE_SUM_MARG (21041)

```
struct trade_sum_marg {
   struct account
   struct series  // Named struct no: 50000
   INT64_T market_value_q  // Market Value
   INT64_T risk_margin_open_q  // Risk Margin Open
   INT64_T naked_risk_margin_q  // Naked Risk Margin
   INT64_T margin_requirement_q  // Margin Requirement Normal
   INT64_T naked_margin_q  // Margin Requirements, Naked
   INT64_T trade_number_q  // Trade number
   char[3] margin_class_s  // Margin class
   UINT8_T marg_meth_inst_c  // Margin method, for instrument class and
instrument series
   UINT8_T account_calculation_type_c  // Margin calculation type ; Of type:
MARGIN_CALCULATION_TYPE_C
```

```
   char[3] filler_3_s  // Filler
   INT64_T pos_unit_id_q  // POS_UNIT_ID_Q
}
```

5.71 RISK_SCALE (21043)

```
struct risk_scale {
   struct account
   INT16_T risk_margin_scaling_factor_n  // Risk margin scaling factor
   char[2] country_id_s  // Name, Country
   char[5] mar_id_s  // Market, Identity
   char[3] filler_3_s  // Filler
}
```

5.72 RM_MARGIN_SIMULATION (21044)

```
struct rm_margin_simulation {
   struct series  // Named struct no: 50000
   struct account
   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
   CHAR filler_1_s  // Filler
   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[2] filler_2_s  // Filler
}
```

5.73 RM_MARGIN_SIM_MARKETS (21045)

```
struct rm_margin_sim_markets {
   struct series  // Named struct no: 50000
}
```

5.74 RM_MARGIN_SIM_TRADES (21046)

```
struct rm_margin_sim_trades {
   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
```


© The NASDAQ OMX Group, Inc. • 2013

```
   char[8] closing_date_s  // Date, Closing
   char[8] date_settlement_s  // Date, Settlement
   char[8] reserved_8_s  // Reserved
}
```

5.75 RM_MARGIN_SIM_PRICES (21047)

```
struct rm_margin_sim_prices {
   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
   INT64_T margin_one_long_q  // Margining Requirements, One Short Position
   INT64_T margin_one_short_q  // Margining Requirements, One Short Position
   UINT16_T dec_in_price_n  // Decimals, Price
   char[2] filler_2_s  // Filler
}
```

5.76 RM_MARGIN_SIM_OMS2_IVL (21048)

```
struct rm_margin_sim_oms2_ivl {
   struct series  // Named struct no: 50000
   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_ivl_n  // Decimals, Price ; Of type: DEC_IN_PRICE_N
   char[2] filler_2_s  // Filler
}
```

5.77 RM_MARGIN_SIM_VOLA (21049)

```
struct rm_margin_sim_vola {
   struct series  // Named struct no: 50000
   INT32_T vol_ivl_long_mid_i  // Volatility Interval Long, Mid
   INT32_T vol_ivl_short_mid_i  // Volatility Interval Short, Mid
   INT32_T vol_ivl_long_low_i  // Volatility Interval Long, Low
   INT32_T vol_ivl_short_low_i  // Volatility Interval Short, Low
   INT32_T vol_ivl_long_high_i  // Volatility Interval Long, High
   INT32_T vol_ivl_short_high_i  // Volatility Interval Short, High
   INT64_T margin_one_short_q  // Margining Requirements, One Short Position
}
```

5.78 RM_MARGIN_SIM_FAILURE_REASON (21050)

```
struct rm_margin_sim_failure_reason {
   char[160] failure_reason_s  // Failure Reason
```

© The NASDAQ OMX Group, Inc. • 2013

```
}
```

5.79 RM_MARGIN_SIM_POS (21051)

```
struct rm_margin_sim_pos {
   INT64_T market_margin_q  // Margin Requirements, Market
   char[3] currency_s  // Currency
   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 price_spread_margin_q  // Price Spread Margin
   INT64_T naked_margin_q  // Margin Requirements, Naked
   struct series  // Named struct no: 50000
   struct account
}
```

5.80 RM_MARGIN_SIM_SUM (21052)

```
struct rm_margin_sim_sum {
   struct series  // Named struct no: 50000
   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
   char[2] filler_2_s  // Filler
}
```

5.81 RM_MARGIN_SIM_DEL (21053)

```
struct rm_margin_sim_del {
   struct series  // Named struct no: 50000
   INT64_T market_margin_q  // Margin Requirements, Market
   char[3] market_currency_s  // Currency, Market
   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 price_spread_margin_q  // Price Spread Margin
   INT64_T naked_margin_q  // Margin Requirements, Naked
}
```

5.82 RM_MARGIN_SIM_SUM_POS_ULG (21054)

```
struct rm_margin_sim_sum_pos_ulg {
   struct series  // Named struct no: 50000
   INT64_T market_margin_q  // Margin Requirements, Market
```


© The NASDAQ OMX Group, Inc. • 2013

```
   char[3] market_currency_s  // Currency, Market
   CHAR filler_1_s  // Filler
   INT64_T naked_margin_q  // Margin Requirements, Naked
   INT32_T marg_price_i  // Margin, Settlement Price
   UINT16_T dec_in_price_n  // Decimals, Price
   char[2] filler_2_s  // Filler
}
```

5.83 RM_MARGIN_SIM_PAY (21055)

```
struct rm_margin_sim_pay {
   struct series  // Named struct no: 50000
   INT64_T market_margin_q  // Margin Requirements, Market
   char[3] market_currency_s  // Currency, Market
   CHAR filler_1_s  // Filler
   INT64_T naked_margin_q  // Margin Requirements, Naked
}
```

5.84 RM_MARGIN_SIM_SUM_PAY_ULG (21056)

```
struct rm_margin_sim_sum_pay_ulg {
   struct series  // Named struct no: 50000
   INT64_T market_margin_q  // Margin Requirements, Market
   char[3] market_currency_s  // Currency, Market
   CHAR filler_1_s  // Filler
}
```

5.85 MARGIN_RESULT_COMPONENTS (21062)

```
struct margin_result_components {
   INT64_T risk_margin_open_q  // Risk Margin Open
   INT64_T risk_margin_deliv_q  // Risk Margin Delivery
   INT64_T spot_val_margin_q  // Spot Value Margin
   INT64_T for_val_margin_q  // Forwards Value Margin
   INT64_T fut_val_margin_q  // Futures Value Margin
   INT64_T opt_val_margin_q  // Options Value Margin
   INT64_T deliv_val_margin_q  // Deliveries Value Margin
   INT64_T payment_margin_future_date_q  // Payment margin future date.
   INT64_T long_opt_min_val_q  // Long Option Minimum Value
   INT64_T today_opt_premium_q  // Todays Option Premium
   char[3] risk_currency_s  // Currency, Risk
   char[3] instr_currency_s  // Instrument Currency
   UINT8_T instrument_or_risk_currency_c  // Instrument or risk currency.
   CHAR filler_1_s  // Filler
}
```

© The NASDAQ OMX Group, Inc. • 2013

5.86 MARGIN_RESULT_OVERDUE (21063)

```
struct margin_result_overdue {
   INT64_T delivery_margin_valuation_date_q  // Delivery margin valuation date.
   INT64_T delivery_margin_overdue_q  // Overdue delivery margin.
   INT64_T payment_margin_valuation_date_q  // Payment margin valuation date.
   INT64_T payment_margin_overdue_q  // Overdue payment margin.
}
```

5.87 MARGIN_RESULT_BASE_API (21064)

```
struct margin_result_base_api {
   INT64_T total_margin_req_q  // TOTAL_MARGIN_REQ_Q
   INT64_T initial_margin_req_q  // Initial margin requirement.
   INT64_T variation_margin_req_q  // Variation margin requirement.
   INT64_T contingent_variation_margin_req_q  // Contingent variation margin
requirement.
   INT64_T info_naked_risk_margin_q  // INFO_NAKED_RISK_MARGIN_Q
}
```

5.88 MARGIN_RESULT_COMPONENTS_PDH (21065)

```
struct margin_result_components_pdh {
   INT64_T financial_margin_q  // FINANCIAL_MARGIN_Q
   INT64_T info_inter_comm_spread_credit_q  // INFO_INTER_COMM_SPREAD_CREDIT_Q
}
```

5.89 MARGIN_RESULT_COMPONENTS_CFM (21066)

```
struct margin_result_components_cfm {
   INT64_T info_market_value_theo_q  // INFO_MARKET_VALUE_THEO_Q
   INT64_T market_value_margin_settled_q  // Market value margin settled
}
```

5.90 MARGIN_AGGREGATION_INFO (21067)

```
struct margin_aggregation_info {
   UINT8_T margin_aggregation_type_c  // Margin Aggregation Type
   UINT8_T gross_or_net_c  // Gross Or Net
}
```


© The NASDAQ OMX Group, Inc. • 2013

5.91 MARGIN_POSITION_INFO (21068)

```
struct margin_position_info {
   struct series  // Named struct no: 50000
   INT64_T nbr_held_q  // Held
   INT64_T nbr_written_q  // Written
   char[3] margin_class_s  // Margin class
   UINT8_T marg_meth_inst_c  // Margin method, for instrument class and
instrument series
}
```

5.92 MARGIN_RESULT_PAYMENT_MARGIN (21069)

```
struct margin_result_payment_margin {
   INT64_T payment_margin_future_date_q  // Payment margin future date.
}
```

5.93 ANSWER_MARGIN_AGGREGATION_GROUP_ROW (21071)

```
struct answer_margin_aggregation_group_row {
   struct account
   UINT8_T account_role_c  // ACCOUNT_ROLE_C
   UINT8_T aggregate_what_c  // AGGREGATE_WHAT_C
   UINT8_T gross_or_net_c  // Gross Or Net
}
```

5.94 RM_MARGIN_SIM_TRADES_ACCOUNT (21072)

```
struct rm_margin_sim_trades_account {
   struct account
}
```

5.95 MARGIN_AGGREGATION_GROUP_INFO (21073)

```
struct margin_aggregation_group_info {
   struct margin_aggregation_group  // Of type: ACCOUNT
   struct trading_code
   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
   char[40] description_s  // Description
}
```

© The NASDAQ OMX Group, Inc. • 2013

5.96 VAR_PRICE_CHANGE_SCENARIO (21076)

```
struct var_price_change_scenario {
   INT32_T price_change_i  // Price change
   char[3] base_currency_s  // Currency, Base
   char[3] price_currency_s  // Currency, Price
   UINT16_T dec_in_price_n  // Decimals, Price
   INT32_T scenario_number_n  // Scenario Number
}
```

5.97 GLOBAL_VAR_PARAMETERS (21077)

```
struct global_var_parameters {
   char[12] clh_id_s  // Clearinghouse
   UINT32_T percentile_for_margin_i  // Percentile for margin
}
```

5.98 GROUP_VAR_PARAMETERS (21078)

```
struct group_var_parameters {
   INT32_T fx_multiplier_i  // FX margin multiplier, 2 implicit decimals
   char[16] fxm_id_s  // FX margin parameters, Identity
   char[3] margin_class_s  // Margin class
   CHAR filler_1_s  // Filler
}
```

5.99 LATEST_TRADE_NUMBER_PER_INS_TYPE (21079)

```
struct latest_trade_number_per_ins_type {
   struct series  // Named struct no: 50000
   INT32_T trade_number_i  // Trade Number
   UINT32_T ext_trade_number_u  // Trade Number, External
}
```

5.100 OB_LEVELS_SEQUENCE_NUMBER (33001)

```
struct ob_levels_sequence_number {
   UINT32_T sequence_number_u  // Sequence Number
}
```


© The NASDAQ OMX Group, Inc. • 2013

5.101 OB_LEVELS_ID (33002)

```
struct ob_levels_id {
   struct series  // Named struct no: 50000
   UINT32_T block_n  // Block Size
}
```

5.102 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
   }
}
```

5.103 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
}
```

5.104 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
}
```

5.105 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
```

© The NASDAQ OMX Group, Inc. • 2013

```
   Array ITEM [max no: 32] {
      INT32_T premium_i  // Premium
   }
}
```

5.106 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
}
```

5.107 OB_LEVELS_QUERY_DATA (33020)

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

5.108 OB_LEVELS_CLOSING (33031)

```
struct ob_levels_closing {
   INT32_T closing_price_i  // Price, Closing
   INT64_T open_balance_u  // Open Interest
}
```

5.109 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
}
```

5.110 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
```


© The NASDAQ OMX Group, Inc. • 2013

```
   UINT8_T items_c  // Item
   Array ITEM [max no: 32] {
      UINT32_T no_of_orders_u  // Orders, Number of
   }
}
```

5.111 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
}
```

5.112 MARKET_INFO_TRD (33036)

```
struct market_info_trd {
   INT32_T last_trade_report_price_i  // Price, Last Trade Report
   INT64_T last_trade_report_qty_u  // Quantity, Last Trade Report
}
```

5.113 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
}
```

5.114 OB_LEVELS_UNDISCLOSED_QUANTITY (33041)

```
struct ob_levels_undisclosed_quantity {
   UINT16_T bid_mask_n  // Mask, Bid
   UINT16_T ask_mask_n  // Mask, Ask
}
```

© The NASDAQ OMX Group, Inc. • 2013

5.115 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
}
```

5.116 MARKET_INFO_OMFI (33047)

```
struct market_info_omfi {
   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
}
```

5.117 PRICE_MEDIAN_ID (33070)

```
struct price_median_id {
   struct series  // Named struct no: 50000
}
```

5.118 PRICE_MEDIAN (33071)

```
struct price_median {
   INT32_T median_bid_price_i  // Price, Median Bid
   INT32_T median_ask_price_i  // Price, Median Ask
}
```

5.119 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
```


© The NASDAQ OMX Group, Inc. • 2013

```
   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
  struct exchange_info_s  // Internally overlayed structure: OM_EXCHANGE_INFO
}
```

5.120 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
  struct exchange_info_s  // Internally overlayed structure: OM_EXCHANGE_INFO
   INT64_T total_volume_i  // Total Volume
}
```

5.121 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
  struct exchange_info_s  // Internally overlayed structure: OM_EXCHANGE_INFO
   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
   }
}
```

5.122 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
   struct exchange_info_s  // Internally overlayed structure: OM_EXCHANGE_INFO
    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
}
```

5.123 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
   struct exchange_info_s  // Internally overlayed structure: OM_EXCHANGE_INFO
}
```

5.124 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
   struct exchange_info_s  // Internally overlayed structure: OM_EXCHANGE_INFO
    INT64_T total_volume_i  // Total Volume
}
```

5.125 TRADE_REPORT_1_TRANS (34021)

```
struct trade_report_1_trans {
    struct transaction_type
    struct series  // Named struct no: 50000
    struct order_var
    struct party
   struct exchange_info_s  // Internally overlayed structure: OM_EXCHANGE_INFO
    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
```


© The NASDAQ OMX Group, Inc. • 2013

```
    UINT8_T deferred_publication_c  // Deferred Publication
    CHAR filler_1_s  // Filler
}
```

5.126 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
}
```

5.127 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
}
```

5.128 INDICATIVE_QUOTE_BASE (34026)

```
struct indicative_quote_base {
    struct series  // Named struct no: 50000
    QUAD_WORD order_number_u  // Order Number
    INT64_T quantity_i  // Quantity
    UINT32_T sequence_number_u  // Sequence Number
    UINT32_T ob_position_u  // Order Book Position
    INT32_T price_i  // Price
    struct owner  // Of type: TRADING_CODE
    UINT8_T ob_command_c  // Order-Book Command
    UINT8_T bid_or_ask_c  // Bid or Ask
    char[2] filler_2_s  // Filler
}
```

© The NASDAQ OMX Group, Inc. • 2013

5.129 INDICATIVE_QUOTE_FIXED_INCOME (34027)

```
struct indicative_quote_fixed_income {
   INT32_T corresponding_yield_price_i  // Corresponding Yield/Price
}
```

5.130 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
  struct exchange_info_s  // Internally overlayed structure: OM_EXCHANGE_INFO
}
```

5.131 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
  struct exchange_info_s  // Internally overlayed structure: OM_EXCHANGE_INFO
   INT64_T total_volume_i  // Total Volume
}
```

5.132 BLOCK_PRICE_TRANS_P (34107)

```
struct block_price_trans_p {
   struct transaction_type
   struct series  // Named struct no: 50000
   struct trading_code
```


© The NASDAQ OMX Group, Inc. • 2013

```
   struct give_up_member  // Named struct no: 50002
   struct exchange_info_s  // Internally overlayed structure: OM_EXCHANGE_INFO
   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
   }
}
```

5.133 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
   struct exchange_info_s  // Internally overlayed structure: OM_EXCHANGE_INFO
   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
}
```

5.134 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
   struct exchange_info_s  // Internally overlayed structure: OM_EXCHANGE_INFO
}
```

© The NASDAQ OMX Group, Inc. • 2013

5.135 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
   struct exchange_info_s  // Internally overlayed structure: OM_EXCHANGE_INFO
   INT64_T total_volume_i  // Total Volume
}
```

5.136 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
   struct exchange_info_s  // Internally overlayed structure: OM_EXCHANGE_INFO
   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
}
```

5.137 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
```


© The NASDAQ OMX Group, Inc. • 2013

```
        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
    }
}
```

5.138 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
}
```

5.139 EXTENDED_TRADE_TICKER (34402)

```
struct extended_trade_ticker {
    UINT16_T trade_condition_n  // Trade Condition
    UINT16_T deal_info_n  // Deal Information
}
```

5.140 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
}
```

5.141 FIXED_INCOME_TRADE_TICKER (34404)

```
struct fixed_income_trade_ticker {
    INT32_T corresponding_yield_price_i  // Corresponding Yield/Price
}
```

© The NASDAQ OMX Group, Inc. • 2013

5.142 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
}
```

5.143 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[3] filler_3_s  // Filler
}
```

5.144 FREE_TEXT (34801)

```
struct free_text {
   char[15] customer_info_s  // Customer, Information
   CHAR filler_1_s  // Filler
}
```

5.145 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
}
```

5.146 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
```


© The NASDAQ OMX Group, Inc. • 2013

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

5.147 ORDER_OWNER (34804)

```
struct order_owner {
   struct owner  // Of type: TRADING_CODE
}
```

5.148 ORDER_NUMBER (34805)

```
struct order_number {
   QUAD_WORD order_number_u  // Order Number
}
```

5.149 TIME_IN_FORCE (34807)

```
struct time_in_force {
   UINT16_T time_validity_n  // Validity Time
   char[2] filler_2_s  // Filler
}
```

5.150 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
}
```

5.151 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
```

© The NASDAQ OMX Group, Inc. • 2013

```
}
```

5.152 LINKED_ORDER_BASE (34810)

```
struct linked_order_base {
   struct timestamp_in  // Of type: TIME_SPEC
   struct timestamp_created  // Of type: TIME_SPEC
}
```

5.153 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
   struct exchange_info_s  // Internally overlayed structure: OM_EXCHANGE_INFO
   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
   }
}
```

5.154 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
}
```

5.155 MULTI_LEG_ORDER_INSERT_P (34819)

```
struct multi_leg_order_insert_p {
   struct transaction_type
   struct series  // Named struct no: 50000
   struct trading_code
```


© The NASDAQ OMX Group, Inc. • 2013

```
   INT32_T premium_i  // Premium
   struct give_up_member  // Named struct no: 50002
  struct exchange_info_s  // Internally overlayed structure: OM_EXCHANGE_INFO
   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
   }
}
```

5.156 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
}
```

5.157 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
}
```

5.158 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
```

© The NASDAQ OMX Group, Inc. • 2013

```
   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
}
```

5.159 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
}
```

5.160 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
}
```

5.161 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
}
```

5.162 QUERY_ORDER_BROADCAST_NEXT (34911)

```
struct query_order_broadcast_next {
  UINT32_T sequence_first_next_u  // Sequence First ; Of type: SEQUENCE_FIRST_U
   UINT8_T instance_next_c  // Next Instance Number
   char[3] filler_3_s  // Filler
}
```


© The NASDAQ OMX Group, Inc. • 2013

5.163 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
}
```

5.164 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
}
```

5.165 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
}
```

5.166 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
}
```

© The NASDAQ OMX Group, Inc. • 2013

5.167 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
}
```

5.168 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
   }
}
```

5.169 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
}
```

5.170 DOCUMENT_URL (35004)

```
struct document_url {
   UINT8_T items_c  // Item
   CHAR[255] url_link_s  // Link, URL
}
```

5.171 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
```


© The NASDAQ OMX Group, Inc. • 2013

```
   char[3] filler_3_s  // Filler
}
```

5.172 NS_REMOVE (37002)

```
struct ns_remove {
   struct series  // Named struct no: 50000
}
```

5.173 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
}
```

5.174 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
```

© The NASDAQ OMX Group, Inc. • 2013

```
}
```

5.175 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
}
```

5.176 NS_CALC_RULE (37104)

```
struct ns_calc_rule {
   UINT32_T accr_intr_round_u  // Accrued Interest Rounding
   UINT32_T clean_pr_round_u  // Clean Price Rounding
   UINT16_T yield_conv_n  // Yield Convention
   UINT16_T ex_coupon_n  // Period, Ex Coupon
   UINT8_T accr_intr_ud_c  // Accrued Interest Up or Down
   UINT8_T clean_pr_ud_c  // Clean Price Up or Down
   UINT8_T day_count_conv_c  // Day Count Convention
   UINT8_T eom_count_conv_c  // End of Month Count Convention
   UINT8_T set_start_consid_c  // Calculate Settlement Amount
   UINT8_T set_end_consid_c  // Set End Consideration
   UINT8_T calculation_conv_c  // Calculation Convention
   UINT8_T cadj_trade_price_c  // Cadj. Trade Price
   UINT8_T ex_coupon_calc_type_c  // Ex-coupon calculation type
   char[3] filler_3_s  // Filler
}
```

5.177 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
}
```

5.178 NS_PRICE_TICK_CORR (37113)

```
struct ns_price_tick_corr {
```


© The NASDAQ OMX Group, Inc. • 2013

```
   struct tick_size
   UINT16_T dec_in_premium_n  // Decimals, Premium
   char[2] filler_2_s  // Filler
}
```

5.179 NS_INST_CLASS_CMS (37114)

```
struct ns_inst_class_cms {
   char[12] valuation_group_id_s  // Valuation Group Identity ; Of type:
VAG_ID_S
   char[12] haircut_id_s  // Haircut ; Of type: HCT_ID_S
   INT32_T vag_limit_i  // Valuation Group Limit (%)
   UINT8_T collateral_type_c  // Collateral types
   UINT8_T eligible_as_margin_coll_c  // Is eligible as margin collateral
   UINT8_T eligible_as_def_fund_coll_c  // Is eligible as margin collateral
   CHAR filler_1_s  // Filler
}
```

5.180 NS_INST_CLASS_LEG_CALC_RULE (37115)

```
struct ns_inst_class_leg_calc_rule {
   struct currency  // Of type: SERIES ; Named struct no: 50000
   struct rate_index  // Of type: SERIES ; Named struct no: 50000
   UINT16_T settlement_days_n  // Settlement, Days or Month
   char[5] settlement_calender_s  // Non-trading Days, Identity ; Of type:
NTD_ID_S
   char[5] reset_day_calender_s  // Non-trading Days, Identity ; Of type:
NTD_ID_S
   UINT8_T rate_type_c  // Fixed or Float ; Of type: FIXED_OR_FLOAT_C
   UINT8_T rollover_period_c  // Rollover Period
   UINT8_T day_count_conv_c  // Day Count Convention
   UINT8_T payment_set_c  // Payment Set
   UINT8_T business_day_conv_c  // BUSINESS_DAY_CONV_C
   UINT8_T reset_days_c  // Reset Days
   UINT8_T reset_days_type_c  // Reset days type
   UINT8_T leg_number_c  // Leg Number
}
```

5.181 NS_INST_CLASS_TRR_DEF_PUBL (37118)

```
struct ns_inst_class_trr_def_publ {
   INT64_T traded_quantity_q  // Traded Quantity
   INT32_T time_delay_i  // Time Delay
   UINT8_T publ_at_end_of_day_c  // Publish at End of Day
   char[3] filler_3_s  // Filler
}
```

© The NASDAQ OMX Group, Inc. • 2013

5.182 NS_INST_CLASS_EXT6 (37120)

```
struct ns_inst_class_ext6 {
   INT32_T min_qty_increment_i  // Minimum Quantity Increment
}
```

5.183 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
}
```

5.184 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
}
```


© The NASDAQ OMX Group, Inc. • 2013

5.185 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
}
```

5.186 NS_INDEX_LINKED (37204)

```
struct ns_index_linked {
   INT32_T index_at_dated_i  // INDEX_AT_DATED_I
   UINT16_T lag_in_index_n  // LAG_IN_INDEX_N
   UINT16_T dec_in_index_n  // DEC_IN_INDEX_N
   char[16] ixv_id_s  // IXV_ID_S
   UINT8_T protect_coupon_c  // PROTECT_COUPON_C
   UINT8_T protect_redempt_c  // PROTECT_REDEMPT_C
   UINT8_T rounding_before_index_c  // Rounding before index
   CHAR filler_1_s  // Filler
}
```

5.187 NS_UNDERLYING_POWER (37206)

```
struct ns_underlying_power {
   char[6] time_delivery_start_s  // Time, Delivery Start
   char[6] time_delivery_stop_s  // Time, Delivery Stop
}
```

5.188 NS_UNDERLYING_EXT3 (37209)

```
struct ns_underlying_ext3 {
   INT64_T outstanding_amount_q  // Outstanding Amount
   UINT32_T issued_price_u  // Issued Price
   char[32] long_underlying_id_s  // Long Underlying Id
   char[32] abbrev_name_s  // Abbreviation Name
   char[9] loan_number_s  // Loan Number
   char[12] benchmark_bond_code_s  // Benchmark Bond Code
   char[64] long_free_text_s  // Free Text, Long
   char[32] sub_fix_income_type_s  // Sub Fixed Income Type
   char[2] lead_manager_country_id_s  // Lead Manager, Country
   char[5] lead_manager_ex_customer_s  // Lead Manager, Customer
   char[2] arranger_country_id_s  // Arranger, Country
   char[5] arranger_ex_customer_s  // Arranger, Customer
   UINT8_T has_amortiziation_c  // Has Amortiziation
}
```

© The NASDAQ OMX Group, Inc. • 2013

5.189 NS_REFERENCE_RATE (37210)

```
struct ns_reference_rate {
   char[32] name_s  // Name
   char[8] date_determination_s  // Date, Determination
   char[8] date_from_s  // Date, From
   INT32_T rate_i  // Rate
}
```

5.190 NS_INDEX_VALUE (37211)

```
struct ns_index_value {
   char[8] date_index_s  // Date, Index
   INT32_T index_value_i  // INDEX_VALUE_I
   UINT16_T dec_in_index_n  // DEC_IN_INDEX_N
   char[2] filler_2_s  // Filler
}
```

5.191 NS_LOTTERY_BONDS (37212)

```
struct ns_lottery_bonds {
   char[32] name_s  // Name
   char[8] date_lottery_s  // Date, Lottery
   char[8] date_payout_s  // Date, Payout
}
```

5.192 NS_CONVERTIBLES (37213)

```
struct ns_convertibles {
   char[8] date_convert_from_s  // Date, Convert From
   char[8] date_convert_through_s  // Date, Convert Through
}
```

5.193 NS_DERIVED_FROM (37214)

```
struct ns_derived_from {
   UINT32_T derived_percentage_u  // Derived Percentage
   UINT32_T base_price_u  // Base Price
   char[128] derived_from_s  // Derived From
   char[3] base_cur_s  // Currency, Trading
   CHAR filler_1_s  // Filler
}
```


© The NASDAQ OMX Group, Inc. • 2013

5.194 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
   UINT8_T traded_c  // Traded
}
```

5.195 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
}
```

5.196 NS_INST_SERIES_POWER (37303)

```
struct ns_inst_series_power {
   char[8] date_delivery_start_s  // Date, Delivery Start
   char[8] date_delivery_stop_s  // Date, Delivery Stop
}
```

5.197 NS_INST_SERIES_REPO (37304)

```
struct ns_inst_series_repo {
```

© The NASDAQ OMX Group, Inc. • 2013

```
   UINT16_T no_of_sub_n  // Substitution, Max Number
   UINT16_T delta_alloc_time_n  // Time, Allocation
   char[8] start_date_s  // Date, Start
   char[8] end_date_s  // Date, End
   UINT8_T money_or_par_c  // Money or Par
   char[12] term_code_s  // TERM_CODE_S
   char[3] filler_3_s  // Filler
}
```

5.198 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
   CHAR filler_1_s  // Filler
}
```

5.199 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
}
```

5.200 NS_INST_SERIES_LEG_FLOW (37309)

```
struct ns_inst_series_leg_flow {
   char[8] start_date_s  // Date ; Of type: YYYYMMDD_S
   char[8] end_date_s  // Date ; Of type: YYYYMMDD_S
   char[8] payment_date_s  // Date ; Of type: YYYYMMDD_S
   char[8] reset_date_s  // Date ; Of type: YYYYMMDD_S
   UINT16_T days_in_period_n  // Days in Period
   UINT16_T days_in_year_n  // Days in year
   UINT8_T rate_type_c  // Fixed or Float ; Of type: FIXED_OR_FLOAT_C
   UINT8_T leg_number_c  // Leg Number
   char[2] filler_2_s  // Filler
}
```

5.201 NS_INST_SERIES_EXT5 (37313)

```
struct ns_inst_series_ext5 {
   char[8] date_first_clearing_s  // Date, First Clearing
}
```


© The NASDAQ OMX Group, Inc. • 2013

5.202 NS_INST_TYPE_BASIC (37601)

```
struct ns_inst_type_basic {
   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 traded_c  // Traded
   UINT8_T directed_trade_information_c  // Directed Trade Information
   UINT8_T public_deal_information_c  // Public Deal Information
   UINT8_T pricing_method_c  // Pricing method
}
```

5.203 NS_INST_TYPE_SECUR (37602)

```
struct ns_inst_type_secur {
   char[15] settlement_product_s  // Settlement product
   UINT8_T maintain_positions_c  // Maintain Positions
   UINT8_T post_trade_proc_c  // Post Trade processed
   UINT8_T pos_handling_c  // Position handling
   UINT8_T pre_novation_collateral_check_c  // Pre novation collateral check
   UINT8_T settlement_type_c  // Settlement, Type
}
```

5.204 NS_TURNOVER_LIST_BASE (37701)

```
struct ns_turnover_list_base {
   char[32] turnover_list_name_s  // Turnover List Name
   char[40] description_s  // Description
   char[3] base_cur_s  // Currency, Trading
   char[2] country_id_s  // Name, Country
   UINT8_T list_type_c  // List type
   char[2] filler_2_s  // Filler
}
```

5.205 NS_TURNOVER_LIST_ITEM (37702)

```
struct ns_turnover_list_item {
   struct series  // Named struct no: 50000
   UINT16_T sort_item_n  // Sort item
   char[64] list_heading_s  // List heading
   char[2] filler_2_s  // Filler
}
```

© The NASDAQ OMX Group, Inc. • 2013

5.206 NS_PRE_TRADE_LIMIT (37801)

```
struct ns_pre_trade_limit {
   INT32_T order_rate_limit_i  // Order Rate Limit
   char[16] ptl_suffix_s  // Pre Trade Limit Suffix
   char[2] country_id_s  // Name, Country
   char[5] ex_customer_s  // Customer, Identity
   char[32] spons_user_name_s  // Sponsoring User
   char[2] sponsored_client_country_id_s  // Sponsored Client, Country
   char[5] sponsored_client_ex_customer_s  // Sponsored Client, Customer
   INT16_T warning_breach_lvl_n  // Warning Breach Level
   INT16_T not_breach_lvl_n  // Notification Breach Level
   UINT8_T enable_warn_email_c  // Enable warning emails
   UINT8_T enable_not_email_c  // Enable notification emails
   UINT8_T enable_breach_email_c  // Enable breach emails
   UINT8_T db_operation_c  // Operation
   UINT8_T intraday_c  // Intraday.
   char[8] valid_from_date_s  // Valid From Date
   UINT8_T enable_restr_instr_c  // Enable Restricted Instruments
   UINT8_T enable_def_user_c  // Enable Default User
   char[3] filler_3_s  // Filler
}
```

5.207 NS_PRE_TRADE_LIMIT_USER (37802)

```
struct ns_pre_trade_limit_user {
   struct user_code
   char[8] valid_from_date_s  // Valid From Date
}
```

5.208 NS_PRE_TRADE_LIMIT_PARAM (37803)

```
struct ns_pre_trade_limit_param {
   struct series  // Named struct no: 50000
   INT64_T max_order_size_q  // Max Order Size
   INT64_T open_buy_q  // Open Buy
   INT64_T open_sell_q  // Open Sell
   INT64_T traded_bought_q  // Traded Bought
   INT64_T traded_sold_q  // Traded Sold
   INT64_T traded_net_q  // Traded Net
   INT64_T total_buy_q  // Total Buy
   INT64_T total_sell_q  // Total Sell
   INT64_T total_net_buy_q  // Total Net Buy
   INT64_T total_net_sell_q  // Total Net Sell
   UINT8_T pre_trade_limit_param_unit_c  // Pre Trade Limit Param Unit
   char[8] valid_from_date_s  // Valid From Date
   char[3] filler_3_s  // Filler
}
```


© The NASDAQ OMX Group, Inc. • 2013

5.209 NS_PRE_TRADE_LIMIT_NOT (37804)

```
struct ns_pre_trade_limit_not {
   char[128] not_email_addr_s  // Notification email address
   char[8] valid_from_date_s  // Valid From Date
}
```

5.210 NS_PRE_TRADE_LIMIT_ID (37805)

```
struct ns_pre_trade_limit_id {
   char[32] ptl_id_s  // Pre Trade Limit Identity
}
```

5.211 NS_ACCOUNT_TYPE_BASIC (37901)

```
struct ns_account_type_basic {
   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
   UINT8_T collaterals_only_c  // Allow collateral
   CHAR acct_type_c  // Counterparty Type
   char[2] filler_2_s  // Filler
}
```

5.212 NS_PRICE_QUOTE_RESP (37951)

```
struct ns_price_quote_resp {
   struct series  // Named struct no: 50000
   UINT16_T resp_fulfilled_n  // Required fulfilled resp. in % with 0 decimals
   UINT16_T min_hold_time_n  // Min lifetime of placed quote(sec)
   CHAR mm_resp_type_c  // Market Maker, Type
   char[3] filler_3_s  // Filler
}
```

5.213 NS_VLD_MAX_SPREAD (37952)

```
struct ns_vld_max_spread {
```

© The NASDAQ OMX Group, Inc. • 2013

```
   INT32_T lower_limit_i  // Premium/Price, Low Limit
   INT32_T upper_limit_i  // Premium/Price, High Limit
   INT32_T spread_i  // Spread
   UINT32_T no_bid_quote_req_i  // No bid quote required if ask price below
   UINT16_T decimals_n  // Decimals
   char[5] spread_id_s  // Max spread id
   CHAR spread_unit_c  // Spread Unit
}
```

5.214 NS_PRICE_QUOTE_CRITERIA (37953)

```
struct ns_price_quote_criteria {
   INT32_T min_vol_n  // Minimum volume required
   UINT16_T nbr_days_to_exp_n  // Number of cycles or calendar days
   UINT16_T min_otm_n  // Number of OTM for single supervision
   UINT16_T min_itm_n  // Number of ITM for single supervision
   UINT16_T nbr_of_strk_n  // Number of strikes for coupled supervision
   char[5] spread_id_s  // Max spread id
   CHAR days_or_exp_c  // Days or expiration unit
   CHAR atm_supervise_c  // Supervise ATM
   CHAR all_supervise_c  // Supervise all series
   UINT8_T alw_roll_exp_dat_c  // Shift responsibility on exp.date
   char[3] filler_3_s  // Filler
}
```

5.215 OTC_BASE_TRADE_REPORT (38001)

```
struct otc_base_trade_report {
   struct party
   struct account
   struct give_up_account  // Of type: ACCOUNT
   struct series  // Named struct no: 50000
   char[32] passthrough_s  // Passthrough Information
   char[8] settlement_date_s  // Date, Settlement
   char[8] asof_date_s  // Date, As Of
   char[80] participant_info_s  // Participant Info
   char[32] name_s  // Name
   UINT8_T bought_or_sold_c  // Bought or Sold
   UINT8_T trade_report_category_c  // Trade Report Category
   char[52] private_match_field_s  // Private match field
   char[2] filler_2_s  // Filler
}
```

5.216 OTC_TRADE_REPORT_DATA (38002)

```
struct otc_trade_report_data {
   struct trading_code
   struct user_code
   struct auth_by_whom
```


© The NASDAQ OMX Group, Inc. • 2013

```
   UINT32_T delivery_unit_u  // Delivery Unit
   UINT32_T trade_report_type_i  // Trade Report Type
   UINT64_T trade_report_nbr_q  // Trade report number
   UINT64_T party_trade_report_nbr_q  // Party trade report number
   INT32_T sequence_number_i  // Sequence Number
   UINT32_T netting_req_nbr_u  // Netting request number
   UINT32_T pay_calc_req_nbr_u  // Pay calc request number
   INT32_T deal_number_i  // Deal Number
   UINT16_T trade_report_version_n  // Trade report version
   char[8] timestamp_date_s  // Timestamp, Date
   char[6] timestamp_time_s  // Timestamp, Time
   char[12] isin_code_s  // ISIN Code
   UINT8_T trade_report_state_c  // Trade Report State
   UINT8_T trade_report_sub_state_c  // Trade Report Substate
   UINT8_T trade_report_reason_c  // Trade report reason
   UINT8_T authorization_state_c  // Authorization State
   struct reported_by  // Of type: TRADING_CODE
   UINT8_T affirmation_state_c  // Affirmation state
   struct affirmed_by  // Of type: TRADING_CODE
   char[3] filler_3_s  // Filler
}
```

5.217 OTC_FRA_TRADE_REPORT (38003)

```
struct otc_fra_trade_report {
   struct float_rate_index  // Of type: SERIES ; Named struct no: 50000
   INT64_T notional_amount_q  // Notional amount
   INT32_T fixed_interest_rate_i  // Fixed Interest Rate
   char[8] float_rate_fixing_date_s  // Float Rate Fixing Date
   char[8] date_termination_s  // Date, Maturity
   UINT8_T day_count_conv_c  // Day Count Convention
   char[3] filler_3_s  // Filler
}
```

5.218 OTC_FRA_DATA (38004)

```
struct otc_fra_data {
   struct float_rate_series  // Of type: SERIES ; Named struct no: 50000
   INT64_T fixed_consideration_q  // Fixed Consideration
   INT64_T float_consideration_q  // Float Consideration
   INT64_T pay_amount_q  // Pay Amount
   INT32_T float_interest_rate_i  // Float Interest Rate
}
```

5.219 OTC_IRS_DATA (38005)

```
struct otc_irs_data {
   UINT16_T flow_version_n  // Trade report version ; Of type:
TRADE_REPORT_VERSION_N
```

© The NASDAQ OMX Group, Inc. • 2013

```
   char[8] delivery_unit_date_s  // DELIVERY_UNIT_DATE_S
   UINT8_T termination_state_c  // Termination State
   char[3] filler_3_s  // Filler
}
```

5.220 OTC_IRS_TRADE_REPORT (38006)

```
struct otc_irs_trade_report {
   struct upfront  // Of type: PAYMENT
   char[8] date_termination_s  // Date, Maturity
   INT64_T notional_amount_q  // Notional amount
   UINT8_T business_day_conv_c  // BUSINESS_DAY_CONV_C
   UINT8_T rate_reset_c  // Rate Reset
   UINT8_T reset_days_c  // Reset Days
   UINT8_T payment_set_c  // Payment Set
}
```

5.221 IRS_MEMBER_PAY (38007)

```
struct irs_member_pay {
   struct irs_leg {
      INT32_T fixed_interest_rate_i  // Fixed Interest Rate
      struct float_rate_index  // Of type: SERIES ; Named struct no: 50000
      INT32_T spread_i  // Spread
      INT32_T init_interest_rate_i  // Init Interest Rate
      char[8] first_rollover_date_s  // First Rollover Date
      UINT8_T day_count_conv_c  // Day Count Convention
      UINT8_T rollover_period_c  // Rollover Period
      UINT8_T rollover_day_c  // Rollover Day
      UINT8_T fixed_or_float_c  // Fixed or Float
      struct party_pay  // Of type: PARTY
   }
}
```

5.222 IRS_COUNTERPARTY_PAY (38008)

```
struct irs_counterparty_pay {
   struct irs_leg {
      INT32_T fixed_interest_rate_i  // Fixed Interest Rate
      struct float_rate_index  // Of type: SERIES ; Named struct no: 50000
      INT32_T spread_i  // Spread
      INT32_T init_interest_rate_i  // Init Interest Rate
      char[8] first_rollover_date_s  // First Rollover Date
      UINT8_T day_count_conv_c  // Day Count Convention
      UINT8_T rollover_period_c  // Rollover Period
      UINT8_T rollover_day_c  // Rollover Day
      UINT8_T fixed_or_float_c  // Fixed or Float
      struct party_pay  // Of type: PARTY
   }
```


© The NASDAQ OMX Group, Inc. • 2013

```
}
```

5.223 STANDARD_TRADE_REPORT (38009)

```
struct standard_trade_report {
   INT64_T quantity_i  // Quantity
   INT32_T premium_i  // Premium
   char[8] filler_8_s  // Filler
   char[15] customer_info_s  // Customer, Information
   UINT8_T open_close_req_c  // Open  Close Request
   UINT8_T ext_t_state_c  // Trade Report Type
   CHAR[32] exchange_info_s  // Exchange, Information
   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 filler_1_s  // Filler
   struct match_id
   QUAD_WORD order_number_u  // Order Number
}
```

5.224 OTC_OPERATION_INFO (38012)

```
struct otc_operation_info {
   INT32_T sequence_number_i  // Sequence Number
   UINT8_T trade_operation_c  // Trade Operation
   char[3] filler_3_s  // Filler
}
```

5.225 OTC_TRADE_OPERATION (38013)

```
struct otc_trade_operation {
   struct account
   struct trading_code
   UINT64_T trade_report_number_q  // TRADE_REPORT_NUMBER
   UINT64_T party_trade_report_number_q  // TRADE_REPORT_NUMBER ; Of type:
TRADE_REPORT_NUMBER_Q
   INT64_T trade_operation_number_q  // TRADE_OPERATION_NUMBER_Q
   char[80] participant_info_s  // Participant Info
   UINT8_T trade_report_state_c  // Trade Report State
   UINT8_T trade_report_sub_state_c  // Trade Report Substate
   UINT8_T trade_report_reason_c  // Trade report reason
   UINT8_T trade_operation_c  // Trade Operation
}
```

5.226 OTC_TRADE (38014)

```
struct otc_trade {
   struct account
```

© The NASDAQ OMX Group, Inc. • 2013

```
   struct pos_account  // Of type: ACCOUNT
   struct series  // Named struct no: 50000
   UINT64_T trade_report_number_q  // TRADE REPORT NUMBER
   INT64_T trade_quantity_i  // Quantity, Trade
   INT32_T trade_price_i  // Price, Trade
   INT32_T trade_clean_price  // Clean price ; Of type: CLEAN_PRICE
   UINT8_T bought_or_sold_c  // Bought or Sold
   char[3] filler_3_s  // Filler
}
```

5.227 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
}
```

5.228 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
}
```

5.229 EXCHANGE_INFO (50004)

```
struct exchange_info {
   struct exchange_info_s  // Internally overlayed structure: OM_EXCHANGE_INFO
}
```

5.230 ACCOUNT_VIM (50005)

```
struct account_vim {
   struct account
}
```

5.231 MARGIN_AGGREGATION_GROUP_VIM (50006)

```
struct margin_aggregation_group_vim {
```


© The NASDAQ OMX Group, Inc. • 2013

```
   struct account
}
```

5.232 MRA_ACCOUNT_VIM (50007)

```
struct mra_account_vim {
   struct account
}
```

5.233 RISK_EXPOSURE_LIMIT_VIM (50010)

```
struct risk_exposure_limit_vim {
   struct mra_account  // Of type: ACCOUNT
   UINT64_T trade_report_nbr_q  // Trade report number
   INT64_T margin_requirement_q  // Margin Requirement Normal
   INT64_T margin_requirement_without_trade_q  // Margin Requirement Normal ;
 Of type: MARGIN_REQUIREMENT_Q
   INT64_T exposure_limit_q  // EXPOSURE_LIMIT_Q
   char[3] currency_s  // Currency
   CHAR filler_1_s  // Filler
}
```

© The NASDAQ OMX Group, Inc. • 2013

© The NASDAQ OMX Group, Inc. • 2013

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
BD29	Directed Give Up	Standard	dedicated	4
BD39	Dedicated Trade Change Information	Standard	dedicated	4
BD41	DC Holding Trade	Variable	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
BI26	Pay note information ready	Standard	general	1
BI27	Clearing message	Standard	general	1
BI28	Bond Index Parameters	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
BI74	Dedicated Broker to Broker Message Info	Standard	dedicated	4

© The NASDAQ OMX Group, Inc. • 2013

Transaction Type	Name	Design	Information Type	Information Type Value
BI75	General Broker to Broker Message Info	Standard	general	1
BI76	Broker to Broker Message Status	Standard	dedicated	4
BI81	Market Announcement Information	Variable	general	1
BI93	Report ready	Standard	general	1
BI94	Planned Instrument Session Info	Standard	general	1
BI95	One Sided Auction Result	Standard	dedicated	4
BL8	Request with Volume	Standard	dedicated	4
BL22	Dedicated Market Maker Alarm	Standard	dedicated	4
BO1	Order Book Changes, with Identity	Standard	instrument class	7
BO2	Order Book Changes, without Identity	Standard	instrument class	7
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
BO49	Price Median	Variable	instrument class	7
BO55	Trade Report Notification	Variable	dedicated	4
BO61	Issuer Order Book Changes	Standard	instrument class	7
BO98	Indicative Quote Changes	Variable	instrument class	7
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

© The NASDAQ OMX Group, Inc. • 2013

Transaction Type	Name	Design	Information Type	Information Type Value
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
BU44	Legal Account Instrument Update	Standard	general	1
BU47	Haircut Update	Standard	general	1
BU50	Non-Settlement Days Update	Standard	general	1
BU53	Corporate Action Update	Standard	general	1
BU54	Valid Sector Codes Update	Standard	general	1
BU87	Market Maker Protection Update	Standard	dedicated	4
BU88	Turnover List Update	Variable	general	1
BU90	Pre Trade Limit Update	Variable	dedicated	4
BU92	Strip Series Update	Standard	general	1
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
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
BU134	Account Type update	Variable	general	1
BU135	Market Maker Obligations update	Variable	general	1

© The NASDAQ OMX Group, Inc. • 2013

Transaction Type	Name	Design	Information Type	Information Type Value
CB3	Directed OTC Trade Report	Variable	dedicated	4
CB146	CL OTC Trade Operation Rejected	Variable	general	1
FB1	Directed Collateral	Variable	dedicated	4
FB6	Collateral Transaction broadcast (VIM)	Variable	dedicated	4
FB17	Collateral Evaluation Run Broadcast (VIM)	Variable	general	1
FB18	Collateral Evaluation Run Broadcast, dedicated (VIM)	Variable	dedicated	4
JB1	Margin Calculation Runs	Variable	general	1
JB2	Margin Calculation Runs, dedicated	Variable	dedicated	4
KB1	Directed OTC Trade Report	Variable	dedicated	4
KB10	OTC Trade Operation on Hold	Variable	general	1
MI3	Market established	Standard	dedicated	4
MI4	Quote Request with Volume Information	Standard	derivative	2
MI5	Market Maker Underlying Price	Standard	dedicated	4
SB1	DvP Instruction	Standard	dedicated	4

© The NASDAQ OMX Group, Inc. • 2013

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.

abbrev_name_s (Abbreviation Name)	
Datatype	char[32]
Description	Specifies the abbreviation name for the underlying.
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_collateral_handling_c (Account Collateral Handling)	
Datatype	UINT8_T
Description	Sets where collaterals are handled for a margin requirement account.
Value Set	**name** / **value** / **description** Acc Coll Handling None / 0 / None Used for other accounts than margin requirement accounts Acc Coll Handling CMS With DD / 1 / CMS with Direct Debit Used for margin requirement accounts where collaterals are in CMS and Direct Debits are created for deficit. Acc Coll Handling CMS No DD / 2 / CMS without Direct Debit Used for margin requirement accounts where collaterals are in CMS and Direct Debits are not created.

© The NASDAQ OMX Group, Inc. • 2013

name	value	description
Acc Coll Handling At Custodian	3	At custodian Used for margin requirement accounts where collaterals are at custodian bank.

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_role_c (ACCOUNT_ROLE_C)	
Datatype	UINT8_T
Description	How to include an account

Value Set

name	value
POSITION_ACCOUNT	1
MARGIN_CALCULATION_ACCOUNT	2
MARGIN_REQUIREMENT_ACCOUNT	3

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
accr_intr_round_u (Accrued Interest Rounding)	

© The NASDAQ OMX Group, Inc. • 2013

Datatype	UINT32_T
Description	Accrued Interest Rounding
accr_intr_ud_c (Accrued Interest Up or Down)	
Datatype	UINT8_T
Description	Accrued Interest Up/Down
Value Set	**name** / **value** Up / 1 Down / 2
acct_type_c (Counterparty Type)	
Datatype	CHAR
Description	Counterparty Type
Value Set	**name** / **value** Not applicable / Direct / D Member / M Omnibus / O Indirect Pledging / I Individual Clearing Account / A Clearing Client / C
accumulated_consideration_q (Consideration, Accumulated)	
Datatype	INT64_T
Description	The accumulated consideration for OIS swaps.
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

© The NASDAQ OMX Group, Inc. • 2013

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>
acc_risk_type_c (Account Risk Type)	
Datatype	UINT8_T
Description	Defines account properties for margin requirements.
Value Set	<table><tr><th>name</th><th>value</th></tr><tr><td>Not used</td><td>1</td></tr><tr><td>Not used</td><td>2</td></tr><tr><td>Direct Pledging Account</td><td>3</td></tr><tr><td>Participant</td><td>4</td></tr><tr><td>Omnibus Account</td><td>5</td></tr><tr><td>Indirect Pledging Account</td><td>6</td></tr><tr><td>Clearing Client</td><td>7</td></tr><tr><td>Individual Clearing Account</td><td>8</td></tr></table>
acc_state_c (Account State)	
Datatype	UINT8_T
Description	Defines the state that the account is in.
Value Set	<table><tr><th>value</th><th>description</th></tr><tr><td>0</td><td>None</td></tr><tr><td>1</td><td>Registered Account has been registered but not validated.</td></tr><tr><td>2</td><td>Inactive Account has been active and then inactivated.</td></tr><tr><td>3</td><td>Active Account is validated and open for position or trade.</td></tr><tr><td>4</td><td>Deleted Account is deleted.</td></tr></table>
acc_type_s (Account Type)	
Datatype	char[12]
Description	Tells what type of account it is
acnt_account_type_c (Account Type for accounting)	
Datatype	UINT8_T

© The NASDAQ OMX Group, Inc. • 2013

Description	The type of account for accounting.
Value Set	**name** / **value** AAT_Any / 1 AAT_Balance / 2 AAT_House / 3 AAT_MarketMaker / 4
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_group_percentage_i (Actual group percentage)	
Datatype	INT32_T
Description	Actual valuation group percentage
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. 1 / Trades added in the query are saved as frozen in the back end. These frozen added

value	description
	trades can be queried via API transaction RQ72.

adjusted_base_collateral_req_q (Adjusted base collateral requirement)	
Datatype	INT64_T
Description	Adjusted base collateral requirement. The number of decimals equals decimals in premium price of currency.

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.

affirmation_state_c (Affirmation state)	
Datatype	UINT8_T
Description	Enumeration describing the affirmation state of a Trade Report

Value Set

name	value
Not_required	0
Holding	1
Affirmed_by_party	2
Automatically_affirmed	3
Rejected	4
Auto_limit_exceeded	5

aggregate_what_c (AGGREGATE_WHAT_C)	
Datatype	UINT8_T
Description	What should be aggregated

Value Set

name	value
POSITION	1
REQUIREMENT	2

aggressive_c (Aggressive)	
Datatype	UINT8_T

© The NASDAQ OMX Group, Inc. • 2013

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
agreement_date_s (Date, Agreement)	
Datatype	char[8]
agreement_type_s (Agreement, Type)	
Datatype	char[24]
Description	Agreement type
agreement_version_s (Agreement, Version)	
Datatype	char[24]
Description	Specifies the agreement version.
alarm_status_u (Alarm Status)	
Datatype	UINT32_T
Description	This field describes the severity of the alarm.
Value Set	**value** / **description** 1 / Warning 2 / Alarm/Penalty 3 / Disconnected alarm/penalty
allow_delayed_c (Allow delayed trade reporting)	
Datatype	UINT8_T
Description	Specifies if this trade report is allow to report with deferred publication.
Value Set	**name** / **value** Yes / 1 No / 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

© The NASDAQ OMX Group, Inc. • 2013

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	**name** / **value** Yes / 1 No / 2
all_or_none_c (All Or None)	
Datatype	UINT8_T
Description	Specifies whether the information relates to the All or None Orderbook.
Value Set	**value** / **description** 1 / Yes 2 / No
all_supervise_c (Supervise all series)	
Datatype	CHAR
Value Set	**name** / **value** Yes / Y No / N
alw_roll_exp_dat_c (Shift responsibility on exp.date)	
Datatype	UINT8_T
Description	Shift responsibility on exp.date
Value Set	**value** / **description** 1 / Yes 2 / No
amount_q (Amount; Collateral amount or quantity.Decimals according to dec_in_amount_n.)	
Datatype	INT64_T
Description	Trade Value; Nominal * Quantity

© The NASDAQ OMX Group, Inc. • 2013

amount_u (Amount)	
Datatype	INT64_T
Description	The amount of money.
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.
apply_holiday_c (State holiday applied, Yes/No)	
Datatype	UINT8_T
Description	State holiday applied Yes (1)/ No (2)
Value Set	**name** / **value** Yes / 1 No / 2
arranger_country_id_s (Arranger, Country)	
Datatype	char[2]
Description	The exchange identity that together with Arranger, Customer represents the arranger.
arranger_ex_customer_s (Arranger, Customer)	
Datatype	char[5]
Description	This field together with Arranger, Country, identifies the member/participant that represents the arranger.
ascii_bin_c (ASCII or Binary)	
Datatype	UINT8_T
Description	ASCII or Binary?
Value Set	**value** / **description** 1 / ASCII 2 / Binary
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

© The NASDAQ OMX Group, Inc. • 2013

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.
as_of_date_s (Date, As Of)	
Datatype	char[8]

© The NASDAQ OMX Group, Inc. • 2013

Description	The date an object is valid for. Date in YYYYMMDD.
atm_price_i (Price, At-The-Money)	
Datatype	INT32_T
Description	At-The-Money price, used for options.
atm_supervise_c (Supervise ATM)	
Datatype	CHAR
Value Set	**name** / **value** Yes / Y No / N
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
auction_type_c (Auction Type)	
Datatype	UINT8_T
Description	Specifies the type of the issuing auction.
Value Set	**name** / **value** Issuing / 1 Buy Back / 2

© The NASDAQ OMX Group, Inc. • 2013

<table>
<tr><td colspan="2">auction_uncross_date_s (Auction Uncross Date)</td></tr>
<tr><td>Datatype</td><td>char[8]</td></tr>
<tr><td>Description</td><td>The date when the uncross will be performed. The date is together with Auction Uncross Time specified as UTC.
Format: YYYYMMDD.</td></tr>
<tr><td colspan="2">auction_uncross_time_s (Auction Uncross Time)</td></tr>
<tr><td>Datatype</td><td>char[6]</td></tr>
<tr><td>Description</td><td>The time when the uncross will be performed at Auction Uncross Date. The time is together with Auction Uncross Date specified as UTC.
Time in ASCII, format is HHMMSS.</td></tr>
<tr><td colspan="2">authorization_state_c (Authorization State)</td></tr>
<tr><td>Datatype</td><td>UINT8_T</td></tr>
<tr><td>Description</td><td>Enumeration for the various authorization options.</td></tr>
<tr><td>Value Set</td><td><table><tr><th>name</th><th>value</th></tr><tr><td>none</td><td>0</td></tr><tr><td>Authorized</td><td>1</td></tr><tr><td>Needed</td><td>2</td></tr><tr><td>Not needed</td><td>3</td></tr></table></td></tr>
<tr><td colspan="2">authorized_c (Authorized)</td></tr>
<tr><td>Datatype</td><td>UINT8_T</td></tr>
<tr><td>Description</td><td>Defines if the user sending the query is authorized to use the Trade Report Type.</td></tr>
<tr><td>Value Set</td><td><table><tr><th>value</th><th>description</th></tr><tr><td>1</td><td>Yes
The trade report type is allowed for the user.</td></tr><tr><td>2</td><td>No
The trade report type is not allowed for the user.</td></tr></table></td></tr>
<tr><td colspan="2">auto_net_c (Auto Netting)</td></tr>
<tr><td>Datatype</td><td>UINT8_T</td></tr>
<tr><td>Description</td><td>If position on this account will be netted automatically in after business operation.</td></tr>
<tr><td>Value Set</td><td><table><tr><th>value</th><th>description</th></tr><tr><td>0</td><td>Not netted</td></tr><tr><td>1</td><td>Netted</td></tr></table></td></tr>
<tr><td colspan="2">average_c (Average)</td></tr>
<tr><td>Datatype</td><td>UINT8_T</td></tr>
<tr><td>Description</td><td>Not applicable.</td></tr>
</table>

© The NASDAQ OMX Group, Inc. • 2013

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>
average_period_c (Average Period)	
Datatype	UINT8_T
Description	Not applicable.
Value Set	<table><tr><th>value</th><th>description</th></tr><tr><td>0</td><td>Not applicable</td></tr><tr><td>1</td><td>Quarterly</td></tr><tr><td>2</td><td>Half Year</td></tr><tr><td>3</td><td>Year</td></tr></table>
balance_account_q (Balance Account)	
Datatype	INT64_T
Description	The balance on an account. The number of decimals equals decimals in premium price of currency.
balance_guarantee_q (Balance Guarantee)	
Datatype	INT64_T
Description	The guarantee balance on an account. The number of decimals equals decimals in premium price of currency.
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.
balance_security_q (Security, Balance)	
Datatype	INT64_T
Description	The excess security amount. A negative number indicates a deficit. The number of decimals equals decimals in premium price of currency.
base_collateral_req_q (Base collateral requirement)	
Datatype	INT64_T
Description	Base collateral requirement. The number of decimals equals decimals in premium price of currency.
base_currency_s (Currency, Base)	
Datatype	char[3]
Description	The base currency

© The NASDAQ OMX Group, Inc. • 2013

base_cur_id_s (Currency, Base)	
Datatype	char[3]
Description	Defines the base currency for the account. 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_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_price_u (Base Price)	
Datatype	UINT32_T
Description	Defines the base price for the derived from with three implicit decimals.
basis_swap_relation_c (The relation of cash flows)	
Datatype	UINT8_T
Description	The relation of this cash flow vs corresponding cash flow.
Value Set	

name	value	description
Bs_Relation_Unknown	0	UNKNOWN The relation is not determined.
Bs_Relation_Short	1	SHORT This cash flow is shorter.
Bs_Relation_Long	2	LONG This cash flow is longest.
Bs_Relation_Equal	3	EAUAL This cash flow have same length.

bc_adjustment_factor_i (Base collateral requirement adjustment factor.)	
Datatype	INT32_T
Description	The reduction factor in percent used to derive adjusted base collateral requirement.
benchmark_bond_code_s (Benchmark Bond Code)	
Datatype	char[12]
Description	Defines the benchmark bond code for the underlying.
best_ask_i (BEST_ASK_I)	
Datatype	INT32_T
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)	

© The NASDAQ OMX Group, Inc. • 2013

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_i (BEST_BID_I)	
Datatype	INT32_T
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

© The NASDAQ OMX Group, Inc. • 2013

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.

© The NASDAQ OMX Group, Inc. • 2013

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-

© The NASDAQ OMX Group, Inc. • 2013

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.

© The NASDAQ OMX Group, Inc. • 2013

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.

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.

block_n (Block Size)	
Datatype	UINT32_T
Description	Minimum number of units (options, futures, forwards and so on) in an order transaction.
bond_quotation_i (Bond Quotation)	
Datatype	INT32_T
Description	Bond quotation describes a quote relation between an amount and a quantity for bonds, i.e. Amount = Bond Quotation * Quantity
book_transparancy_c (Book Transparancy)	
Datatype	UINT8_T
Description	Specifies if the action is open or hidden.

Value Set

name	value
Open	1
Hidden	2

boolean (BOOLEAN)	
Datatype	CHAR
Description	Intermediate field.
bought_or_sold_c (Bought or Sold)	

© The NASDAQ OMX Group, Inc. • 2013

Datatype	UINT8_T
Description	Defines if the item or amount in question is bought or sold.
Value Set	**value** / **description** 1 / Bought 2 / Sold
broadcast_number_n (Broadcast Number)	
Datatype	UINT16_T
Description	A number used to distinguish between different broadcasts.
broadcast_reason_c (Broadcast Reason)	
Datatype	UINT8_T
Description	Enumeration for the various reasons for sending a broadcast concerning a particular trade report.
Value Set	**name** / **value** Trade report is new / 1 Trade report has changed state / 2 Trade report has been authorized / 3 Trade report has been rectified / 4 Trade report has been assigned a delivery unit number / 5 Trade report has sent off a letter confirmation tx / 6
broker_id_s (Broker, Identity)	
Datatype	char[5]
Description	The broker id is optional and may be used to identify brokers on a firm.
buffer_length_n (Buffer Length)	
Datatype	UINT16_T
Description	Actual length of sent report buffer
business_date_s (Date, Business)	
Datatype	char[8]
Description	Date in ASCII. Format: YYYYMMDD
business_day_conv_c (BUSINESS_DAY_CONV_C)	
Datatype	UINT8_T
Description	Used to find out the nearest business date to calculated end date of a period.
Value Set	**name** / **value** Following / 1 Modified following / 2

© The NASDAQ OMX Group, Inc. • 2013

name	value
Preceding	3

buy_amount_q (Buy Amount)	
Datatype	INT64_T
Description	Defines the buy amount.
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

buy_sell_c (BUY_SELL_C)	
Datatype	UINT8_T
buy_si_s (Buy Settlement Instruction)	
Datatype	char[120]
Description	Specifies the buy settlement instruction.
buy_use_ssi_c (Special settlement instruction)	
Datatype	UINT8_T
Description	Specifies the special settlement instruction.
cabinet_format_c (Cabinet Format)	

© The NASDAQ OMX Group, Inc. • 2013

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

cadj_trade_price_c (Cadj. Trade Price)

Datatype	UINT8_T
Description	Specifies if trade price is adjusted.

Value Set

name	value
Yes	1
No	2

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

calculation_conv_c (Calculation Convention)

Datatype	UINT8_T
Description	Calculation Convention

Value Set

name	value
Compound	1
CompoundSimple	2
Simple_MM	3
Discount	4
US Treasury	5
Proceed	6

© The NASDAQ OMX Group, Inc. • 2013

calc_ask_marg_vol_i (Calculation Margin Volatility, Ask)	
Datatype	INT32_T
Description	Defines the volatility used in margin calculations. For instruments other than options, this field always equals zero. Expressed in percent, 4 implicit decimals.
calc_ask_price_i (Calculation Price, Ask)	
Datatype	INT32_T
Description	Ask price used in margin calculations.
calc_ask_theo_c (Calculation Ask Price, Theoretical Mark)	
Datatype	UINT8_T
Description	Defines the origin of the calculation ask settlement price.
Value Set	**name** / **value** Missing / 0 Theoretically calculated / 1 From the order book / 2 Manually updated / 3 Artificial / 4
calc_bid_marg_vol_i (Calculation Margin Volatility, Bid)	
Datatype	INT32_T
Description	Defines the volatility used in margin calculations. For instruments other than options, this field always equals zero. Expressed in percent, 4 implicit decimals.
calc_bid_price_i (Calculation Price, Bid)	
Datatype	INT32_T
Description	Bid price used in margin calculations.
calc_bid_theo_c (Calculation Bid Price, Theoretical Mark)	
Datatype	UINT8_T
Description	Defines the origin of the calculation bid settlement price.
Value Set	**name** / **value** Missing / 0 Theoretically calculated / 1 From the order book / 2 Manually updated / 3 Artificial / 4
calc_delta_protection_q (Calculated Delta Protection quantity)	
Datatype	INT64_T
Description	Calculated delta value for market maker protection
calc_fixing_value_i (Calculation Price, Fixing)	

© The NASDAQ OMX Group, Inc. • 2013

Datatype	INT32_T
Description	Fixing value used in margin calculations.
calc_fix_theo_c (Calculation price, Fixing Origin)	
Datatype	UINT8_T
Description	Defines the origin of the calculation fixing value.
Value Set	**name** / **value** Missing / 0 Theoretically calculated / 1 From the order book / 2 Manually updated / 3 Artificial / 4
calc_marg_price_i (Calculation Price, Margin)	
Datatype	INT32_T
Description	Margin settlement price used in margin calculations.
calc_marg_theo_c (Calculation Margin Settlement Price, Origin)	
Datatype	UINT8_T
Description	Defines the origin of the calculation margin settlement price.
Value Set	**name** / **value** Missing / 0 Theoretically calculated / 1 From the order book / 2 Manually updated / 3 Artificial / 4
calc_mid_marg_vol_i (Calculation Margin Volatility, Mid)	
Datatype	INT32_T
Description	Defines the volatility used in margin calculations. For instruments other than options, this field always equals zero. Expressed in percent, 4 implicit decimals.
calc_quantity_protection_q (Calculated Quantity Protection)	
Datatype	INT64_T
Description	Calculated quantity value for market maker protection
cancel_ref_s (SWIFT reference.)	
Datatype	char[16]
Description	SWIFT reference for instruction requested to be cancelled.
cash_account_s (Account, Cash)	
Datatype	char[24]

© The NASDAQ OMX Group, Inc. • 2013

Description	A Cash Account (Cash Record) is unique within a Member. Allowed characters are (A-Z), (a-z), (0-9) and space and hyphen.
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.
cash_rate_i (CASH_RATE_I)	
Datatype	INT32_T
Description	The interest rate for the REPO. That is, the the interest rate for borrowing money between the settlement date and the unwind settlement date
cash_requirement_q (Cash Requirement)	
Datatype	INT64_T
Description	The requirement on cash being at immediate disposal. The number of decimals equals decimals in premium price of currency.
cash_transfer_code_s (Cash transfer code)	
Datatype	char[12]
Description	Cash transfer code.
cash_transfer_group_s (Cash transfer group)	
Datatype	char[12]
Description	Cash transfer group.
cash_type_s (Cash Type)	
Datatype	char[4]
Description	Cash type, reason for generating Cash payment. Valid cash types are exchange specific and also not dynamically definable in the CDB. Currently exist at least:

Value Set

name	value
11AM Call	11AM
24HR Call	24HR
Foreign Exchange	FORX
Swaps	SWAP
Fixed Interest	FINT
Coupon Payment	COUP
Options	OPTN
Funds Transfer	FNTR
Repurchase	REPO
Electricity Payment	ELEC
Term Cash	TCSH

© The NASDAQ OMX Group, Inc. • 2013

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.
ccc_id_s (Curve Correlation Cube)	
Datatype	char[12]
Description	Name of Curve Correlation Cube
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)
chain_info_c (Chain Info)	
Datatype	UINT8_T
change_previous_i (Change, Since Previous)	

© The NASDAQ OMX Group, Inc. • 2013

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
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_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_lnk_leg_delete	17	Linked order leg deleted (customer specific)

© The NASDAQ OMX Group, Inc. • 2013

name	value	description
change_reason_lnk_leg_mod	18	Linked order leg modified (customer specific)
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_auction_delete	34	Market (Auction) order deleted during auction
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_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.

© The NASDAQ OMX Group, Inc. • 2013

name	value	description
change	3	Modification 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
Commission	11	Commission
Settlement_intraday_collect	12	Intraday settlement collect
Accrued_interest	13	The interest accrued on cash instruments.
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
Balance_adjustment	21	Balance adjustment
	23	Fee 3
	24	Fee 4
	25	Fee 5
	26	Fee 6
	27	Fee 7
	28	Fee 8

© The NASDAQ OMX Group, Inc. • 2013

name	value	description
	29	Fee 9
	30	Fair value
Market_Value_Margin	31	Market_Value_Margin Market Value Margin
Market_Value_Interest	32	Market_Value_Interest Market Value Interest

clean_price (Clean price)	
Datatype	INT32_T
Description	Clean price for repo trades
clean_price_i (Clean price)	
Datatype	INT32_T
Description	Defines the clean price. Number of decimals in the clean price can be retrieved from DQ123 substruct ns_calc_rule and field clean_pr_round_u.
clean_price_q (Price, Clean)	
Datatype	UINT64_T
clean_pr_round_u (Clean Price Rounding)	
Datatype	UINT32_T
Description	Clean Price Rounding
clean_pr_ud_c (Clean Price Up or Down)	
Datatype	UINT8_T
Description	Clean Price Up/Down
Value Set	

name	value
Up	1
Down	2

cleared_dec_in_qty_n (Decimals, Quantity)	
Datatype	UINT16_T
Description	Defines decimals in quantity in clearing related quantities.
clearing_account_s (Clearing Account)	
Datatype	CHAR[12]
clearing_date_s (Clearing Date; Clearing date of Exercise/Closing)	
Datatype	char[8]
Description	Date in ASCII for clearing trade, format is YYYYMMDD.
clearing_firm_s (Clearing Firm)	
Datatype	CHAR[4]

© The NASDAQ OMX Group, Inc. • 2013

clh_id_s (Clearinghouse)	
Datatype	char[12]
Description	Clearinghouse identity.
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
closeout_qty_i (Quantity, Close out)	
Datatype	INT64_T
Description	A quantity by which a position should be closed out
closeout_status_i (Status, Close out)	
Datatype	INT32_T
Description	Status from a position close out request
closing_date_s (Date, Closing)	

© The NASDAQ OMX Group, Inc. • 2013

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_otc_trade_operation_c (CL OTC Trade Operation)	
Datatype	UINT8_T
Description	Defines the type CL OTC trade operation.
Value Set	**name** / **value** None / 0 Rectify / 1 Cancel / 2 TransferFromTransitory / 3 Retry / 4 GiveUp / 5 PositionTransfer / 6
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	**name** / **value** / **description** Suspended / S / Suspended from Clearing Active / A / Active
collaterals_only_c (Allow collateral)	
Datatype	UINT8_T
Value Set	**name** / **value** Margin Collateral / 1 No / 2 Default Fund Financial Market / 3 Default Fund Commodities Market / 4 Default Fund Loss Sharing Pool Contribution / 5

© The NASDAQ OMX Group, Inc. • 2013

name	value
Default Fund FX	6
Default Fund Seafood	7

collateral_amount_q (Collateral Amount; Collateral Amount/Quantity)	
Datatype	INT64_T
Description	The currency is implicitly given by the name of the series (and possibly account). The number of decimals equals decimals in premium price of instrument series, except for securities when it equals number of decimals in quantity of instrument series.
collateral_cash_q (Collateral Cash)	
Datatype	INT64_T
Description	Collateral in the form of cash. The number of decimals equals decimals in premium price of currency.
collateral_evaluation_type_c (Collateral evaluation type)	
Datatype	UINT8_T
Description	The enum describes why the collateral evaluation was made.

Value Set

name	value	description
Collateral Evaluation Type None	0	None
Collateral Evaluation	1	Evaluate all collaterals
Update Acc Balance	2	Update the account balance
Update Exp Collateral	3	Update expired collaterals
Collateral Evaluation Deposit	4	Evaluate collateral deposit
Collateral Evaluation Withdraw	5	Evaluate collateral withdraw
Collateral Evaluation Intraday Margin Call	6	Evaluate collateral intraday margin call
Collateral Evaluation Pre Novation	7	Evaluate collateral pre novation
Collateral Evaluation Internal Transfer	10	Evaluate collateral internal transfer

collateral_guarantee_q (Collateral Guarantee)	
Datatype	INT64_T
Description	Collateral in the form of a bank guarantee to a certain amount. The number of decimals equals decimals in premium price of currency.
collateral_nbr_q (Collateral Number)	
Datatype	UINT64_T
Description	Unique number that identifies a collateral position.
collateral_security_q (Security, Collateral)	

© The NASDAQ OMX Group, Inc. • 2013

Datatype	INT64_T
Description	The amount of security collateral after haircut and currency conversion (if applicable).
collateral_state_c (Collateral State)	
Datatype	UINT8_T
Description	Status on a collateral evaluation request.
Value Set	

name	value	description
Idle	1	No processing on going. Awaiting next evaluation request.
Pending	2	
Processing	3	Processing an evaluation request.
Completed	4	The processing ended sucessfully.
Stopped	6	The processing ended unsucessfully.

collateral_transaction_nbr_q (Collateral Transaction Number)	
Datatype	UINT64_T
Description	Unique number that identifies a collateral transaction.
collateral_transaction_state_c (Collateral transaction state)	
Datatype	UINT8_T
Description	Defines the state of the collateral transaction.
Value Set	

name	value	description
Coll Trans State Created	1	Created The collateral transaction has been created.
Coll Trans State Pending	2	Pending The collateral transaction is pending.
Coll Trans State Accepted	3	Accepted The collateral transaction is accepted.
Coll Trans State Settled	4	Settled The collateral transaction is settled.
Coll Trans State Rejected	5	Rejected The collateral transaction is rejected.
Coll Trans State Manually Created	6	Manually created The collateral transaction is manually created.

© The NASDAQ OMX Group, Inc. • 2013

name	value	description
Coll Trans State Manually Settled	7	Manually settled The collateral transaction is manually settled.
Coll Trans State Manually Rejected	8	Manually rejected The collateral transaction is manually-rejected.
Coll Trans State Cancelled	9	Cancelled The collateral transaction is cancelled.

collateral_transaction_type_c (Collateral transaction type)	
Datatype	UINT8_T
Description	Defines the type of the collateral transaction.

Value Set

name	value	description
Coll Trans Type None	0	None No collateral transaction type.
Coll Trans Type Deposit Cash	1	Deposit cash The collateral transaction is of type Deposit Cash.
Coll Trans Type Deposit Security	2	Deposit Security The collateral transaction is of type Deposit Security.
Coll Trans Type Withdraw Cash	3	Withdraw cash The collateral transaction is of type Withdraw Cash.
Coll Trans Type Withdraw Security	4	Withdraw security The collateral transaction is of type Withdraw Security.
Coll Trans Type Deposit Security Corporate Action	5	Deposit security as a result of a corprate action. The collateral transaction is of type Deposit Security as a result of a corporate action.
Coll Trans Type Withdraw Security Corporate Action	6	Withdraw security as a result of a corprate action. The collateral transaction is of type Withdraw Security as a result of a corporate action.
Coll Trans Type Transfer Deposit Cash	7	Deposit cash as a result of an internal transfer. The collateral transaction is of type Internal Transfer Deposit Cash.

© The NASDAQ OMX Group, Inc. • 2013

name	value	description
Coll Trans Type Transfer Withdraw Cash	8	Withdraw cash as a result of an internal transfer. The collateral transaction is of type Internal Transfer Withdraw Cash.
Coll Trans Type Transfer Deposit Security	9	Deposit security as a result of an internal transfer. The collateral transaction is of type Internal Transfer Deposit Security.
Coll Trans Type Transfer Withdraw Security	10	Withdraw security as a result of an internal transfer. The collateral transaction is of type Internal Transfer Withdraw Security.

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

collateral_value_q (Collateral Value)	
Datatype	INT64_T
Description	The collateral value of the collateral. The number of decimals equals decimals in premium price of collateral value currency.

coll_cash_usage_other_curr_q (Collateral cash usage other currency)	
Datatype	INT64_T
Description	The amount of cash collateral used as backup (other currency) after currency conversion (if applicable).

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.

© The NASDAQ OMX Group, Inc. • 2013

	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.
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_confirmed_c (CONDITION_CONFIRMED_C)	
Datatype	UINT8_T

© The NASDAQ OMX Group, Inc. • 2013

Description	Signal if conditions have been confirmed
Value Set	**name** / **value** No condition specified / 0 Confirmation needed / 1 Confirmed / 2
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
consideration_q (Consideration)	
Datatype	INT64_T
consideration_u (Consideration)	
Datatype	UINT64_T
Description	Consideration value.
contingent_variation_margin_req_q (Contingent variation margin requirement.)	
Datatype	INT64_T
Description	Contingent variation margin, i.e. market value for instruments with no daily settlement.
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

© The NASDAQ OMX Group, Inc. • 2013

value	description
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.
corp_action_code_s (Code, Corporate Action)	
Datatype	char[2]
Description	Corporate Action Code
corp_action_level_c (Level, Corporate Action)	
Datatype	UINT8_T
Description	The instrument level the corporate action is assigned to:

Value Set

value	description
1	Underlying
2	Linked Underlying

© The NASDAQ OMX Group, Inc. • 2013

value	description
3	Instrument Class
4	Instrument Series

corp_action_ref_s (Corporate action SWIFT reference.)	
Datatype	char[16]
Description	SWIFT reference.
corp_action_status_c (Status, Corporate Action)	
Datatype	UINT8_T
Description	It is possible to remove a corporate action or notification code during the day. This field indicates if the code is active or not.

Value Set

value	description
1	Enabled
2	Disabled

corp_action_type_c (Corporate Action Type)	
Datatype	UINT8_T

Value Set

name	value
Corporate Action/Basis of Quotation	1
Notification Code/Status Note	2

corp_event_ref_s (Corporate action event SWIFT reference.)	
Datatype	char[16]
Description	SWIFT reference.
corresponding_yield_price_i (Corresponding Yield/Price)	
Datatype	INT32_T
Description	Specifies the corresponding yield if the instrument is traded in price. Specifies the corresponding price if the instrument is traded in yield.
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

© The NASDAQ OMX Group, Inc. • 2013

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_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.
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_direct_debit_c (Create Direct Debit)	
Datatype	UINT8_T
Description	Sets if a collateral evaluation run should create direct debits
Value Set	**value** / **description**: 1 – Yes; 2 – No

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.
crv_id_s (Curve Id)	
Datatype	char[12]
Description	Curve Definition object (ANSWER_YIELD_CURVE_NAMES)
crv_tenor_c (Curve tenor)	
Datatype	UINT8_T
Description	The tenor for the Forward curve.
Value Set	**name** / **value** / **description** None / 0 / One_Day / 1 / 1 Day One_Week / 2 / 1 Week One_Month / 3 / 1 Month Three_Month / 4 / 3 Months Six_Month / 5 / 6 Months One_Year / 6 / 1 Year
crv_type_c (Curve type)	
Datatype	UINT8_T
Description	Which type of curve it is.
Value Set	**name** / **value** / **description** Undefined / 0 / Discount / 1 / The curve is bootstrapped as a discount function. The curve can be used to estimate the present value of future cash flows, i.e. to discount them, but also to estimate their size. Yield / 2 /

© The NASDAQ OMX Group, Inc. • 2013

name	value	description
Forward	3	A forward curve is a curve used to estimate future floating rates for a given tenor.
Ois_Curve	4	A OIS curve is used to estimate and discount future cash flows.

csd_account_from_s (CSD Account, From)	
Datatype	char[20]
Description	The CSD account related to the delivering side in a delivery.
csd_account_to_s (CSD Account, To)	
Datatype	char[20]
Description	The CSD account related to the receiving side in a delivery.
csd_code_s (Code, CSD)	
Datatype	char[34]
Description	Identifies the CSD account number or BIC.
csd_id_s (CSD, Identity)	
Datatype	char[12]
Description	Specifies the clearance system that is connected to instrument class.
csd_status_s (CSD Status)	
Datatype	char[16]
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.
curv_construction_method_c (Curve Construction Method)	
Datatype	UINT8_T

© The NASDAQ OMX Group, Inc. • 2013

Value Set	<table><tr><th>name</th><th>value</th><th>description</th></tr><tr><td>Cubic</td><td>1</td><td>The function between two nodes will be approximated by a third degree polynomial. In each node, where two such functions meet, the value as well as the first and second derivative will be continuous.</td></tr><tr><td>Linear</td><td>2</td><td>The function between two nodes will be approximated by a straight line.</td></tr><tr><td>Piecewise_Constant</td><td>3</td><td>The function between two nodes is constant, i.e. a flat line. At the nodes, the function may discontinuous.</td></tr></table>
cur_unit_c (Currency Unit)	
Datatype	UINT8_T
Description	Specifies the currency unit for underlying prices.
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>
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
data_buffer_s (Data, Buffer)	
Datatype	UINT8_T[61440]
Description	The data buffer contains tha binary information in a file .
date_adjust_s (Date, Adjust)	
Datatype	char[8]
Description	Date of the adjustment. In ASCII format: YYYYMMDD
date_and_time (DATE_AND_TIME)	
Datatype	INT64_T
Description	Intermediate field.
date_booksclose_s (Booksclose Date)	

© The NASDAQ OMX Group, Inc. • 2013

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_convert_from_s (Date, Convert From)	
Datatype	char[8]
Description	The convert from date for convertibles. Format: YYYYMMDD.
date_convert_through_s (Date, Convert Through)	
Datatype	char[8]
Description	The convert through date for convertibles. 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_determination_s (Date, Determination)	
Datatype	char[8]
Description	The determination date for the reference rate. Format: YYYYMMDD.
date_expiration_s (Date, Expiration)	
Datatype	char[8]
Description	Date in ASCII. Format: YYYYMMDD.
date_first_clearing_s (Date, First Clearing)	
Datatype	char[8]

© The NASDAQ OMX Group, Inc. • 2013

Description	The first valid clearing date of the series. 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_from_s (Date, From)	
Datatype	char[8]
Description	The from date for the reference rate. Format: YYYYMMDD.
date_implementation_s (Date, Implementation)	
Datatype	char[8]
Description	Implementation date. Format: YYYYMMDD.
date_index_s (Date, Index)	
Datatype	char[8]
Description	The index date for linked index bonds. 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_lottery_s (Date, Lottery)	
Datatype	char[8]
Description	The lottery date for lottery bonds. 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_payout_s (Date, Payout)	
Datatype	char[8]

© The NASDAQ OMX Group, Inc. • 2013

Description	The payout date for lottery bonds. Format: 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; Date, Settlement Instruction)	
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_span_type_c (Date Span Type)	
Datatype	UINT8_T
Description	Identifies the type of date to be used in trade report queries.
Value Set	**name** / **value** Not applicable / 0 Trade date / 1 Clearing date / 2 Settlement date / 3
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_from_n (DAYS_FROM_N)	
Datatype	UINT16_T
days_in_interest_year_n (Days In Interest Year)	
Datatype	UINT16_T
Description	Number of days in coupon period used for interest rate calculations.
days_in_period_n (Days in Period)	

© The NASDAQ OMX Group, Inc. • 2013

Datatype	UINT16_T
Description	Number of days in a period
days_in_year_n (Days in year)	
Datatype	UINT16_T
Description	Number of days in the year according to the day count convention.
days_or_exp_c (Days or expiration unit)	
Datatype	CHAR
Value Set	

name	value
Days	D
Expiration	E

days_per_year_n (DAYS_PER_YEAR_N)	
Datatype	UINT16_T
Description	Number of days per year
days_to_n (DAYS_TO_N)	
Datatype	UINT16_T
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
TBILL1	10
TBILL2	11

day_count_conv_c (Day Count Convention)	
Datatype	UINT8_T
Description	Day Count Convention
Value Set	

name	value	description
ACT_ACT_ISMA	1	
ACT_ACT_AFB	2	
EURO_BOND_30E360	3	30E-360 EuroBond
US30_360	4	30-360 US convention

© The NASDAQ OMX Group, Inc. • 2013

name	value	description
ACT_365	5	
ACT_360	6	
ACT_ACT_ISDA	7	
BOND_BASIS_30360	8	
ISDA_30E360	9	

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
DCD_Normal	1	The TM deal has been accepted by the Clearinghouse.
DCD_Holding_TM	7	The TM deal is created from an order that is matched with the counterpart order but not yet accepted by the Clearinghouse.
DCD_Deleted	9	The TM deal has been rejected by the Clearinghouse.
DCD_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.

© The NASDAQ OMX Group, Inc. • 2013

Value Set	

name	value	description
deal_info_no_info	0	No info
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.

© The NASDAQ OMX Group, Inc. • 2013

name	value	description
deal_source_manually	2	Matched by system, manually.
deal_source_outside_different	3	Matched Outside Exchange, Different participants
deal_source_outside_different_om	4	Matched outside exchange, different participants, reg. by exchange.
deal_source_outside_same	5	Matched Outside Exchange, One participant
deal_source_outside_same_om	6	Matched outside exchange, one participant, reg. by exchange.
deal_source_auto_combo	7	Combination order matched against another combination order when matched by the Exchange, electronically.
deal_source_swap_box	8	Deal in a Swap Box instrument.
deal_source_auto_internal	9	Matched electronically, member internal.
deal_source_swap_box_internal	10	Deal in a Swap Box instrument, member internal.
deal_source_after_outside_diff	11	After market closure, outside system, different brokers
deal_source_after_outside_diff_om	12	After market closure, outside system, different brokers, registered by the exchange.
deal_source_after_outside_same	13	After market closure, outside system, one broker
deal_source_after_outside_same_om	14	After market closure, outside system, one broker, registered by the exchange.
deal_source_internally_basis	15	Internally created basis trade.
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_pqr	21	Private request for quote.
deal_source_pqr_package	22	Package private request for quote.
deal_source_internal_combo	23	Internally from combo.

© The NASDAQ OMX Group, Inc. • 2013

name	value	description
deal_source_internal_tm	24	Internally from TM.
deal_source_internal_average	25	Internally from average.
deal_source_internal_strip	26	Internally from strip.
deal_source_delta_hedge	27	Delta hedge.
deal_source_internal_bundle	28	CL bundle deal.
deal_source_bb_trade	32	Trade from Bulletin Board.
deal_source_bb_trade_st_combo	33	Trade from Bulletin Board, standard combo.
deal_source_bb_trade_nost_combo	34	Trade from Bulletin Board, non-standard combo.
deal_source_bb_trade_nost_combo_e	35	Trade from Bulletin Board, non-standard combo.
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_no_price	41	No Deal Price.
deal_source_priority_crossing	42	Priority crossing.
deal_source_combo_vs_outright	43	Combination matched outright legs.
deal_source_outside_otc	44	Matched outside exchange, broker.
deal_source_imp_rotation	100	
deal_source_imp_normal	101	
deal_source_imp_out_of_sequence	102	
deal_source_imp_cab_trade	103	
deal_source_imp_combo_single	104	
deal_source_imp_combo_mix	105	
deal_source_fac_orig_order	110	
deal_source_fac_counter_order	111	
deal_source_exp_orig_order	112	

© The NASDAQ OMX Group, Inc. • 2013

name	value	description
deal_source_exp_counter_order	113	
deal_source_unsolicited_order	114	
deal_source_solicited_order	115	
deal_source_block_order	116	
deal_source_trade_rep	117	
deal_source_trade_rep_no_settl	118	
deal_source_imp_combo_buy_write	122	
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.
deal_source_aps	135	Average price system (APS).
deal_source_adjust_wo_price	136	Adjustment without price.
deal_source_adjust_with_price	137	Adjustment with price.
deal_source_ctrade	138	Deal executed at CTrade.
deal_source_cross_product_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).
decimals_n (Decimals)	
Datatype	UINT16_T
Description	Number of decimals.
dec_in_actual_group_percentage_n (Decimals, Percentage)	
Datatype	UINT16_T
Description	Number of implicit decimals.
dec_in_amount_n (Decimals, Amount)	

© The NASDAQ OMX Group, Inc. • 2013

Datatype	UINT16_T
Description	Number of implicit decimals in amount.
dec_in_bq_n (Decimals, Bond Quotation)	
Datatype	UINT16_T
Description	Number of implicit decimals in Bond Quotation.
dec_in_clean_price_n (DEC_IN_CLEAN_PRICE_N)	
Datatype	UINT16_T
dec_in_collateral_price_n (Decimals, Collateral price)	
Datatype	UINT16_T
Description	Number of implicit decimals in collateral price.
dec_in_consideration_n (DEC_IN_CONSIDERATION_N)	
Datatype	UINT16_T
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_dirty_price_n (DEC_IN_DIRTY_PRICE_N)	
Datatype	UINT16_T
dec_in_discount_factor_n (Decimals, Factors)	
Datatype	UINT16_T
Description	Number of decimals in the discount factors.
dec_in_first_quantity_n (DEC_IN_FIRST_QUANTITY_N)	
Datatype	UINT16_T
dec_in_fixing_n (Decimals, Fixing)	
Datatype	UINT16_T
Description	Number of implicit decimals in Fixing.
dec_in_index_n (DEC_IN_INDEX_N)	
Datatype	UINT16_T
Description	Number of decimals used when calculating index.
dec_in_margin_value_i (Decimals, Margin value)	
Datatype	INT32_T
dec_in_nominal_n (Decimals, Nominal)	
Datatype	UINT16_T
Description	Number of implicit decimals in the Nominal Value.
dec_in_premium_n (Decimals, Premium)	

© The NASDAQ OMX Group, Inc. • 2013

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_second_quantity_n (DEC_IN_SECOND_QUANTITY_N)	
Datatype	UINT16_T
dec_in_strike_price_n (Decimals, Strike Price)	
Datatype	UINT16_T
Description	Number of implicit decimals in the strike price.
dec_in_yield_n (Decimals, Yield)	
Datatype	UINT16_T
Description	Number of implicit decimals in Yield value
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
deferred_time_n (Deferred Publication time)	
Datatype	UINT16_T
Description	Number of minutes a trade report publication is deferred
deficit_to_cover_q (Deficit to cover)	
Datatype	INT64_T
Description	Deficit to Cover. The number of decimals equals decimals in premium price of currency.
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

© The NASDAQ OMX Group, Inc. • 2013

delivery_margin_overdue_q (Overdue delivery margin.)	
Datatype	INT64_T
Description	Overdue delivery margin due to unfulfilled delivery engagements. The number of decimals equals decimals in premium price of currency.
delivery_margin_valuation_date_q (Delivery margin valuation date.)	
Datatype	INT64_T
Description	Delivery margin for deliveries settled on valuation date. The number of decimals equals decimals in premium price of currency.
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
2048	REPO first leg
4096	REPO second leg
8192	New contract ODVP
16384	Next clearing date
32768	Cascade event

© The NASDAQ OMX Group, Inc. • 2013

value	description
65536	VAT eligible

delivery_quantity_q (Quantity, Delivery; Settlement Amount to Pay(-)/Receive(+))	
Datatype	INT64_T
Description	Defines the quantity the delivery is based on.
delivery_state_c (Delivery, State)	
Datatype	UINT8_T
Description	Defines what state the delivery is in.

Value Set

value	description
1	Normal
2	Rectified The delivery is rolled back. There exists another rollback delivery that points to this delivery.

delivery_type_c (Delivery, Type)	
Datatype	UINT8_T
Description	Defines what type the delivery is.

Value Set

value	description
1	Normal
2	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.
3	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.
4	Backdated The delivery is backdated which entails that it concerns an event occuring on a previous clearing date.

delivery_unit_date_s (DELIVERY_UNIT_DATE_S)	
Datatype	char[8]
Description	Date when a specific delivery unit number was created
delivery_unit_u (Delivery Unit)	

© The NASDAQ OMX Group, Inc. • 2013

Datatype	UINT32_T
Description	Trade reports, delivery items and dvp-instructions belong to a delivery unit.
deliv_base_quantity_q (Quantity, Delivery Base)	
Datatype	INT64_T
Description	Defines the quantity of the delivery base that is delivered.
deliv_isin_quantity_q (ISIN Underlying, Quantity of Shares; Nbr of underlying to be delivered(-)/Recieved(+))	
Datatype	INT64_T
Description	Quantity of shares etc for ISIN underlying
deliv_val_margin_q (Deliveries Value Margin)	
Datatype	INT64_T
Description	Margin component, deliveries value margin.
delta_alloc_time_n (Time, Allocation)	
Datatype	UINT16_T
Description	Delta allocation time in minutes after last trading time
delta_i (Delta)	
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.

© The NASDAQ OMX Group, Inc. • 2013

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
derived_from_s (Derived From)	
Datatype	char[128]
Description	Defines what the underlying is derived from.
derived_percentage_u (Derived Percentage)	
Datatype	UINT32_T
Description	Defined how many percent the Derived From represent. Expressed with six implicit decimals.
description_s (Description)	
Datatype	char[40]
Description	Description field.
desc_abbreviated_s (Description, Abbreviated)	
Datatype	char[32]
Description	An abbreviated textual description.
desc_long_s (Description, Long)	
Datatype	char[128]
Description	A textual description.
destination_level_c (Destination, Level)	
Datatype	UINT8_T
Description	Defines the destination level.

© The NASDAQ OMX Group, Inc. • 2013

Value Set	<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>
diary_number_s (Diary Number)	
Datatype	char[15]
Description	The diary number for this account.
difflen (DIFFLEN)	
Datatype	char[8]
Description	intermediate field.
directed_trade_information_c (Directed Trade Information)	
Datatype	UINT8_T
Description	Specifies how the directed trade broadcast is distributed.
Value Set	<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>
dirty_price_q (DIRTY_PRICE_Q)	
Datatype	UINT64_T
discount_crv_id_s (Discount curve)	
Datatype	char[12]
Description	The discount curve used when bootstrapping this forward curve.
discount_long_i (Discount, long)	
Datatype	INT32_T
Description	Discount factor to use for long positions.
discount_method_c (Discount Method)	
Datatype	UINT8_T
Value Set	<table><tr><th>name</th><th>value</th></tr><tr><td>Simple yearly rate</td><td>1</td></tr><tr><td>Yearly compound</td><td>2</td></tr><tr><td>Continuous compound</td><td>3</td></tr></table>
discount_short_i (Discount, short)	
Datatype	INT32_T

© The NASDAQ OMX Group, Inc. • 2013

Description	Discount factor to use for short positions.
display_quantity_i (Quantity, Display)	
Datatype	INT64_T
Description	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.
dividend_i (Dividend)	
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
duration_i (Duration)	
Datatype	INT32_T
Description	Defines the duration on a bond index underlying, represented with 3 implicit decimals.
dvp_item_number_u (DVP_ITEM_NUMBER_U)	
Datatype	UINT32_T
dvp_item_properties_u (DVP_ITEM_PROPERTIES_U)	
Datatype	UINT32_T
dvp_length_n (DVP_LENGTH_N)	
Datatype	UINT16_T

© The NASDAQ OMX Group, Inc. • 2013

dvp_properties_u (Delivery Properties)	
Datatype	UINT32_T
Description	Bit mask provides specific information about the delivery.
Value Set	**name** / **value** DvP (Create DvP instruction) / 1 SWIFT (Entered by SWIFT) / 2 Transfer (Other quantity is zero) / 4 Reversing (Reversing BD18) / 8 Overtaking (Overtaking BD18) / 16 Confirm (Holding DvP instruction needs confirmation) / 32 Settled_ext (Don't create DvP instruction - the delivery will be settled externally) / 64
dvp_sequence_number_u (DVP_SEQUENCE_NUMBER_U)	
Datatype	UINT32_T
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	**name** / **value** / **description** edited_price_reason_none / 0 / Void and not used edited_price_reason_refresh / 1 / Sent due to refresh of data edited_price_reason_deal / 2 / Sent due to execution of deal edited_price_reason_cor / 3 / Sent due to correction of data edited_price_reason_delete / 4 / Sent due to deletion of deal edited_price_reason_exclude / 5 / Sent due to exclusion of deal in trade statistics edited_price_reason_include / 6 / Sent due to reinclusion of deal in trade statstics edited_price_reason_reset / 7 / Sent due to reset of trade statstics

© The NASDAQ OMX Group, Inc. • 2013

effective_date_s (Date, Effective)	
Datatype	char[8]
Description	The date in ASCII from when the new cash rate or collateral position is effective. Format: YYYYMMDD
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.
effective_until_s (Effective Until)	
Datatype	char[8]
Description	The date until the collateral position is effective.
eligible_as_def_fund_coll_c (Is eligible as margin collateral)	
Datatype	UINT8_T
Description	Sets if the instrument class is allowed to be used as collateral covering default fund requirements.
Value Set	**value** / **description** 1 / Yes 2 / No
eligible_as_margin_coll_c (Is eligible as margin collateral)	
Datatype	UINT8_T
Description	Sets if the instrument class is allowed to be used as collateral covering margin requirements.
Value Set	**value** / **description** 1 / Yes 2 / No
enable_breach_email_c (Enable breach emails)	
Datatype	UINT8_T
Description	Specifies if breach emails should be sent.
Value Set	**name** / **value** Yes / 1 No / 2
enable_def_user_c (Enable Default User)	
Datatype	UINT8_T
Description	Defines if the default user for the sponsored client is included in the pre-trade limit group.

© The NASDAQ OMX Group, Inc. • 2013

Value Set	**name** / **value** Yes / 1 No / 2
enable_not_email_c (Enable notification emails)	
Datatype	UINT8_T
Description	Specifies if notification emails should be sent.
Value Set	**name** / **value** Yes / 1 No / 2
enable_restr_instr_c (Enable Restricted Instruments)	
Datatype	UINT8_T
Description	Defines if the sponsored client users are restricted to trade only in the instruments defined in the pre-trade limit group.
Value Set	**name** / **value** Yes / 1 No / 2
enable_warn_email_c (Enable warning emails)	
Datatype	UINT8_T
Description	Specifies if warning emails should be sent.
Value Set	**name** / **value** Yes / 1 No / 2
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	**value** / **description** 1 / Yes 2 / No
end_time (END_TIME)	

© The NASDAQ OMX Group, Inc. • 2013

Datatype	INT32_T
eom_count_conv_c (End of Month Count Convention)	
Datatype	UINT8_T
Description	End of Month Count Convention
Value Set	**name** / **value** SAME / 1 LAST360 / 2 LAST / 3
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.
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.
estimated_accumulated_consideration_q (Estimated Consideration, Accumulated)	
Datatype	INT64_T
Description	The estimated accumulated consideration for OIS swaps.

© The NASDAQ OMX Group, Inc. • 2013

estimated_consideration_date_s (Estimated Consideration Date)	
Datatype	char[8]
Description	The consideration is estimated up to this date
event_origin_i (Event, Origin)	
Datatype	INT32_T
Description	Reference to the origin event number.
event_type_c (Event Type)	
Datatype	UINT8_T
Description	Define why a delivery is created.
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
11	Closing Trade
16	Principal Exchange
20	Supervision
21	Manual
22	Rebate
23	Balance events

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

© The NASDAQ OMX Group, Inc. • 2013

value	description
6	Exercise
7	Assign
8	Dividend
9	New Contract Trade
10	Give Up
11	Closing Trade
12	Delivery Flow
13	DVP Settled
14	Member Fee Entrance
15	Member Fee Periodic
16	Principal Exchange
17	Settle Accrued
18	Market Value Calculation

exchange_code_s (Exchange Code)	
Datatype	char[2]
Description	Exchange code in ASCII format, e.g. SE, GB, ED, DK
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_rate_q (Exchange rate)	
Datatype	INT64_T
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.

© The NASDAQ OMX Group, Inc. • 2013

name	value	description
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. 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

excluded_due_to_idmc_c (Excluded due to IDMC)	
Datatype	UINT8_T
Description	Sets if the account is excluded due to IDMC
Value Set	value: 1 — description: True value: 2 — description: False

exclusive_opening_sell_c (Exclusive Opening Sell)	
Datatype	UINT8_T
Description	Is the account allowed to exclusive opening sell?

© The NASDAQ OMX Group, Inc. • 2013

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_limit_q (EXPOSURE_LIMIT_Q)	
Datatype	INT64_T
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)	

© The NASDAQ OMX Group, Inc. • 2013

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
external_ref_s (External reference)	
Datatype	char[16]
Description	Value from GSC/Wizer
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_confirm_c (Is externally confirmed)	
Datatype	UINT8_T
Description	Sets if the collateral transaction is externally confirmed.
Value Set	**value** / **description** 1 / Yes 2 / No

© The NASDAQ OMX Group, Inc. • 2013

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

© The NASDAQ OMX Group, Inc. • 2013

<table>
<tr><td colspan="2">ext_trade_fee_type_c (External Trade, Fee Type)</td></tr>
<tr><td>Datatype</td><td>CHAR</td></tr>
<tr><td>Description</td><td>The external fee type is used to look up the fee table that will be used to calculate the fee for the trade.</td></tr>
<tr><td colspan="2">ext_trade_number_u (Trade Number, External)</td></tr>
<tr><td>Datatype</td><td>UINT32_T</td></tr>
<tr><td>Description</td><td>Trade number assigned by external system</td></tr>
<tr><td colspan="2">ext_t_state_c (Trade Report Type)</td></tr>
<tr><td>Datatype</td><td>UINT8_T</td></tr>
<tr><td>Description</td><td>Defines the type of Trade Report. The available types can be retrieved by Query Trade Report.
This field also contains cancellation status for TM report.</td></tr>
<tr><td>Value Set</td><td><table><tr><th>value</th><th>description</th></tr><tr><td>0</td><td>Not applicable.</td></tr><tr><td>253</td><td>TM report cancelled by exchange
Valid for answers only.</td></tr><tr><td>254</td><td>TM report cancelled by own customer
Valid for answers only.</td></tr><tr><td>255</td><td>TM report cancelled by owner
Valid for answers only.</td></tr></table></td></tr>
<tr><td colspan="2">ex_client_s (Client)</td></tr>
<tr><td>Datatype</td><td>char[10]</td></tr>
<tr><td>Description</td><td>Exchange client is the name of the participant's client.</td></tr>
<tr><td colspan="2">ex_coupon_calc_type_c (Ex-coupon calculation type)</td></tr>
<tr><td>Datatype</td><td>UINT8_T</td></tr>
<tr><td>Description</td><td>Specifies if the ex-coupon period is stated in business days or calendar days.</td></tr>
<tr><td>Value Set</td><td><table><tr><th>name</th><th>value</th></tr><tr><td>Business Days</td><td>1</td></tr><tr><td>Calendar Days</td><td>2</td></tr></table></td></tr>
<tr><td colspan="2">ex_coupon_n (Period, Ex Coupon)</td></tr>
<tr><td>Datatype</td><td>UINT16_T</td></tr>
<tr><td>Description</td><td>Ex Coupon period</td></tr>
<tr><td colspan="2">ex_customer_s (Customer, Identity)</td></tr>
<tr><td>Datatype</td><td>char[5]</td></tr>
<tr><td>Description</td><td>This field together with Country Name, identifies a member/participant of the exchange (such as a bank or broker firm).</td></tr>
<tr><td colspan="2">ex_rate_q (Exchange Rate, Collateral)</td></tr>
</table>

© The NASDAQ OMX Group, Inc. • 2013

Datatype	INT64_T
Description	Exchange rate between the collateral positions market value currency and the collateral value currency.
face_value_q (Face Value)	
Datatype	INT64_T
Description	Face value.
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.
field_s (Field)	
Datatype	char[32]
Description	Name of field in account where validation failed.
file_name_s (File Name)	
Datatype	char[80]
Description	File name representation.
file_type_s (File Type)	
Datatype	char[8]
Description	The string representing the file type, i.e. suffix.
filler_12_s (FILLER_12_S)	
Datatype	char[12]
Description	Filler for alignment
filler_16_s (Filler)	
Datatype	char[16]
Description	Filler
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)	

© The NASDAQ OMX Group, Inc. • 2013

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_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
final_held_q (Held/Long position, After closeout)	
Datatype	INT64_T
Description	The requested held/long position after position closeout
final_open_interest_q (Final Open Interest)	
Datatype	UINT64_T
Description	The number of outstanding contracts at end of the business day.
financial_margin_q (FINANCIAL_MARGIN_Q)	
Datatype	INT64_T
first_dvp_account_s (FIRST_DVP_ACCOUNT_S)	
Datatype	char[24]
first_holiday_id_s (First State Holiday ID)	
Datatype	char[5]

© The NASDAQ OMX Group, Inc. • 2013

Description	First State holiday ID.
first_isin_code_s (FIRST_ISIN_CODE_S)	
Datatype	char[12]
first_quantity_q (Quantity, First)	
Datatype	INT64_T
first_rollover_date_s (First Rollover Date)	
Datatype	char[8]
Description	The end date of the first rollover period
first_settlement_date_s (Date, First Settlement)	
Datatype	char[8]
Description	First Settlement Date in format YYYYMMDD.
fixed_consideration_q (Fixed Consideration)	
Datatype	INT64_T
Description	The consideration for the fixed leg of an OTC contract
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_interest_rate_i (Fixed Interest Rate)	
Datatype	INT32_T
Description	The interest rate for the fixed leg of an OTC contract
fixed_or_float_c (Fixed or Float)	
Datatype	UINT8_T
Description	Fixed or float rate

© The NASDAQ OMX Group, Inc. • 2013

Value Set	**name** / **value** Fixed / 1 Float / 2
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_date_s (Fixing Date)	
Datatype	char[8]
Description	The date (YYYYMMDD) when the consideration should be calculated
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_theo_c (Fixing value, Origin)	
Datatype	UINT8_T
Description	Defines the origin of the fixing value.
Value Set	**name** / **value** Missing / 0 Theoretically calculated / 1 From the order book / 2 Manually updated / 3 Artificial / 4
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.

© The NASDAQ OMX Group, Inc. • 2013

flat_rate_increase_i (Flat rate increase)	
Datatype	INT32_T
Description	Always equal zero.
float_consideration_q (Float Consideration)	
Datatype	INT64_T
Description	The consideration for the floating leg of an OTC contract
float_interest_rate_i (Float Interest Rate)	
Datatype	INT32_T
Description	The interest rate for the floating leg of an OTC contract
float_rate_fixing_date_s (Float Rate Fixing Date)	
Datatype	char[8]
Description	The date (YYYYMMDD) when the consideration should be calculated
flow_number_u (FLOW_NUMBER_U)	
Datatype	UINT32_T
Description	Number for this SWAP flow
flow_operation_c (FLOW_OPERATION_C)	
Datatype	UINT8_T
Description	flow operation is used when a flow is rectified
Value Set	

name	value
No change	0
Enter (i.e. new)	1
Rectify	2
Cancel (i.e. delete)	4

flow_state_c (FLOW_STATE_C)	
Datatype	UINT8_T
Description	Flow state is used to distinguish different flow states
Value Set	

name	value
No change	0
New	1
changed	2
Deleted	4

forward_style_c (Style, Forward)	
Datatype	UINT8_T
Description	Defines if this an Instrument Group where corresponding Instrument Series are forward styled.

© The NASDAQ OMX Group, Inc. • 2013

Value Set	<table><tr><th>name</th><th>value</th><th>description</th></tr><tr><td>NOT_APPLICABLE</td><td>0</td><td>Not applicable</td></tr><tr><td>NORMAL</td><td>1</td><td>Normal</td></tr><tr><td>CFD</td><td>2</td><td>CfD</td></tr><tr><td>INCLUDE_DIVIDEND</td><td>3</td><td>Include dividend</td></tr></table>
for_val_margin_q (Forwards Value Margin)	
Datatype	INT64_T
Description	Margin component, forwards value margin.
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_settlement_date_s (From Settlement Date)	
Datatype	char[8]
Description	Specifies from settlement date.
from_termination_agree_date_s (From Termination Agree Date)	
Datatype	char[8]
Description	The answer to the query should return records from this termination date
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	<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>

© The NASDAQ OMX Group, Inc. • 2013

full_collect_date_s (Full collect date)	
Datatype	char[8]
Description	Timestamp together with full_collect_time_s when a full collect was done
full_collect_time_s (Full collect time)	
Datatype	char[6]
Description	Timestamp together with full_collect_date_s when a full collect was done
full_termination_c (Full Termination)	
Datatype	UINT8_T
Description	Signal if this is partial or full termination
Value Set	**name** / **value** Yes / 1 No / 2
fund_type_c (Fund_Type)	
Datatype	UINT8_T
Description	Defines the type of fund.
Value Set	**name** / **value** Fund_Type_None / 0 Financial_Markets / 1 Commodities_Market / 2 Loss_Sharing_Pool / 3 FX / 4 Seafood / 5
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.

© The NASDAQ OMX Group, Inc. • 2013

<table>
<tr><td colspan="2">fut_val_margin_q (Futures Value Margin)</td></tr>
<tr><td>Datatype</td><td>INT64_T</td></tr>
<tr><td>Description</td><td>Margin component, futures value margin.</td></tr>
<tr><td colspan="2">fxm_id_s (FX margin parameters, Identity)</td></tr>
<tr><td>Datatype</td><td>char[16]</td></tr>
<tr><td>Description</td><td>FX margin parameters id</td></tr>
<tr><td colspan="2">fx_multiplier_i (FX margin multiplier, 2 implicit decimals)</td></tr>
<tr><td>Datatype</td><td>INT32_T</td></tr>
<tr><td colspan="2">gamma_i (Gamma)</td></tr>
<tr><td>Datatype</td><td>INT32_T</td></tr>
<tr><td>Description</td><td>The rate of change in an options delta, due to a change in the price of the underlying. Given with 4 decimals.</td></tr>
<tr><td colspan="2">give_up_number_i (Give Up, Number)</td></tr>
<tr><td>Datatype</td><td>INT32_T</td></tr>
<tr><td>Description</td><td>Unique, within each instrument type (country, market, instrument group) system generated number, for a give-up.</td></tr>
<tr><td colspan="2">give_up_state_c (Give Up, State)</td></tr>
<tr><td>Datatype</td><td>UINT8_T</td></tr>
<tr><td>Description</td><td>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:</td></tr>
<tr><td>Value Set</td><td>
<table>
<tr><th>value</th><th>description</th></tr>
<tr><td>0</td><td>None</td></tr>
<tr><td>1</td><td>Holding</td></tr>
<tr><td>2</td><td>Confirmed
.</td></tr>
<tr><td>4</td><td>Rejected
.</td></tr>
<tr><td>8</td><td>Holding Rectify Trade
.</td></tr>
<tr><td>16</td><td>Holding Rectify Deal
.</td></tr>
<tr><td>32</td><td>Deleted
.</td></tr>
<tr><td>64</td><td>Delete Holding
.</td></tr>
</table>
</td></tr>
<tr><td colspan="2">give_up_text_s (Give Up, Free Text)</td></tr>
<tr><td>Datatype</td><td>char[30]</td></tr>
</table>

© The NASDAQ OMX Group, Inc. • 2013

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.
grand_total_surplus_deficit_base_cur_after_fx_haircut_q (Grand total surplus deficit in base currency)	
Datatype	INT64_T
Description	Grand total surplus or deficit in base currency after fx haircut. The number of decimals equals decimals in premium price of currency.
grand_total_surplus_deficit_base_cur_q (Grand total surplus deficit in base currency)	
Datatype	INT64_T
Description	Grand total surplus or deficit in base currency. The number of decimals equals decimals in premium price of currency.
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_limit_i (Valuation group limit)	
Datatype	INT32_T
Description	Valuation group limit in per cent
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

© The NASDAQ OMX Group, Inc. • 2013

name	value	description
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
group_type_non_deliver-able_rolling_spot	16	Non deliverable rolling spot
group_type_strip	17	Strip

guarantee_type_c (Guarantee Type)

Datatype	UINT8_T
Description	Defines the type of guarantee.

Value Set

name	value
Cash_Settlement_types	1
Margin	2
Bank	3

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
2	Confirmed .
3	Rejected .

© The NASDAQ OMX Group, Inc. • 2013

value	description
4	Delete Holding .
5	Deleted .
6	Extended .

haircut_i (Haircut)	
Datatype	INT32_T
Description	The reduction factor in percent used to derive the collateral value from market value.
haircut_rate_u (Haircut Rate)	
Datatype	UINT32_T
Description	Haircut rate in percent with 4 decimals
has_amortiziation_c (Has Amortiziation)	
Datatype	UINT8_T
Description	Defines if the underlying has amortiziation or not.
Value Set	

name	value
Yes	1
No	2

hct_id_s (Haircut)	
Datatype	char[12]
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_high_i (Held, High)	
Datatype	UINT32_T
Description	Margin vector value for a held series at a high volatility, and at the corresponding spot price, 2 implicit decimals.
held_low_i (Held, Low)	
Datatype	UINT32_T
Description	Margin vector value for a held series at a low volatility, and at the corresponding spot price, 2 implicit decimals.
held_marg_q (Marginables, Held)	

© The NASDAQ OMX Group, Inc. • 2013

Datatype	INT64_T
Description	The number of held marginables in a position.
held_middle_i (Held, Middle)	
Datatype	UINT32_T
Description	Margin vector value for a held series at a medium volatility, and at the corresponding spot price, 2 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_up_i (Volatility Held Up)	
Datatype	INT32_T
Description	Volatility interval up for held options in margin calculations. Expressed in percent, 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

© The NASDAQ OMX Group, Inc. • 2013

Description	Defines the highest traded price during the day.
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
incl_manual_registrations_c (Include manual, not invoiced, registrations)	
Datatype	UINT8_T
Value Set	**value** / **description** 1 / Yes 2 / No
incl_paynotes_c (Include invoiced or payed paynotes)	
Datatype	UINT8_T
Value Set	**value** / **description** 1 / Yes 2 / No
incl_pending_settlements_c (Include pending, not invoiced settlements)	
Datatype	UINT8_T
Value Set	**value** / **description** 1 / Yes 2 / No
incl_t_plus_one_positions_c (Include T+1 Positions)	
Datatype	UINT8_T
Description	Specifies if positions from T+1 sessions should be included in calculations
Value Set	**name** / **value** Yes / 1 No / 2
incl_t_plus_one_prices_c (Include T+1 Prices)	

© The NASDAQ OMX Group, Inc. • 2013

Datatype	UINT8_T
Description	Specifies if prices from T+1 sessions should be included in calculations
Value Set	**name** / **value** / **description** Yes / 1 / Yes No / 2 / No
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_at_dated_i (INDEX_AT_DATED_I)	
Datatype	INT32_T
Description	Index Value at Dated Date, 2 decimals
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.
index_value_i (INDEX_VALUE_I)	
Datatype	INT32_T
Description	Index Value, 2 decimals
indicative_prices_c (Indicative Prices)	
Datatype	UINT8_T
Description	Indicative Prices
Value Set	**name** / **value** Yes / 1 No / 2
info_inter_comm_spread_credit_q (INFO_INTER_COMM_SPREAD_CREDIT_Q)	
Datatype	INT64_T

© The NASDAQ OMX Group, Inc. • 2013

Description	Inter commodity spread credit for SPAN.
info_market_value_theo_q (INFO_MARKET_VALUE_THEO_Q)	
Datatype	INT64_T
Description	Calculated theoretical market value for the position. When used in F*-messages, the number of decimals equals decimals in premium price of series.
info_naked_risk_margin_q (INFO_NAKED_RISK_MARGIN_Q)	
Datatype	INT64_T
Description	Informational field, naked risk margin.
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
21	Start of Evening job for market.
22	Extracted data for report generating are ready (Kofex)
23	NRS batch data loaded completed

© The NASDAQ OMX Group, Inc. • 2013

value	description
24	NRS batch data loaded started
26	Stock deliveries ready
27	Reversed Stock deliveries ready
28	Bilateral Delivery Instructions ready
29	Stock DVP ready
30	Reversed Stock DVP ready
31	Freight spot prices ready
32	Delivery
41	Margin Evening Prices ready
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
90	Prices Daily Trade statistics information
91	Settlement Daily Trade statistics information
98	Final Fixing value established
100	Final 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

© The NASDAQ OMX Group, Inc. • 2013

value	description
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)
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

© The NASDAQ OMX Group, Inc. • 2013

value	description
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
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

© The NASDAQ OMX Group, Inc. • 2013

value	description
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
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

© The NASDAQ OMX Group, Inc. • 2013

value	description
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
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

© The NASDAQ OMX Group, Inc. • 2013

value	description
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
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

© The NASDAQ OMX Group, Inc. • 2013

value	description
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)
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

© The NASDAQ OMX Group, Inc. • 2013

value	description
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_margin_req_q (Initial margin requirement.)	
Datatype	INT64_T
Description	Inital margin, i.e. margin requirement without market value.
initial_trr_min_value_u (Initial Trade Report, Minimum Order Value.)	
Datatype	INT64_T
Description	Not applicable.
init_consideration_q (Initial consideration)	
Datatype	INT64_T
Description	Initial consideration for repo.
init_face_value_q (Initial face value)	
Datatype	INT64_T
Description	Initial face value for repo.
init_interest_rate_i (Init Interest Rate)	
Datatype	INT32_T
Description	The interest rate for the first payment flow
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.

© The NASDAQ OMX Group, Inc. • 2013

Value Set	<table><tr><th>value</th><th>description</th></tr><tr><td>0</td><td>Not instigating part</td></tr><tr><td>1</td><td>Instigating part</td></tr><tr><td>2</td><td>Instigating part unknown or N/A</td></tr></table>
instruction_nbr_u (Instruction number)	
Datatype	UINT32_T
Description	Unique number that identifies a bank/payment instruction.
instrument_group_c (Instrument Group)	
Datatype	UINT8_T
Description	A unique binary representation of the instrument group.
instrument_level_c (INSTRUMENT_LEVEL_C)	
Datatype	UINT8_T
Description	Instrument level.
Value Set	<table><tr><th>name</th><th>value</th></tr><tr><td>None</td><td>0</td></tr><tr><td>Wildcard</td><td>1</td></tr><tr><td>Instrument type</td><td>2</td></tr><tr><td>Instrument class</td><td>3</td></tr><tr><td>Instrument series</td><td>4</td></tr></table>
instrument_or_risk_currency_c (Instrument or risk currency.)	
Datatype	UINT8_T
Description	Sets where collaterals are handled for a margin requirement account.
Value Set	<table><tr><th>name</th><th>value</th><th>description</th></tr><tr><td>both_currencies</td><td>0</td><td>BOTH Used when information is given in both currencies.</td></tr><tr><td>risk_currency</td><td>1</td><td>RISK Used when information is given in risk currency.</td></tr><tr><td>instrument_currency</td><td>2</td><td>INSTRUMENT Used when information is given in instrument currency.</td></tr></table>
instr_currency_s (Instrument Currency)	
Datatype	char[3]
Description	The instrument currency that is used for trading (before currency conversion).
instr_ref_s (SWIFT reference.)	

© The NASDAQ OMX Group, Inc. • 2013

Datatype	char[16]
Description	SWIFT reference.
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.
internal_full_depth_c (Full Depth, Internal)	
Datatype	UINT8_T
Description	Not applicable.
Value Set	**value** / **description** 2 / No
internal_interest_rate_i (Internal Interest Rate)	
Datatype	INT32_T
Description	Internal interest rate on a bond index underlying, represented with 3 implicit decimals.
intraday_c (Intraday.)	
Datatype	UINT8_T
Description	Defines if the change should be activated immediately or next day.
Value Set	**name** / **value** Yes / 1 No / 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

© The NASDAQ OMX Group, Inc. • 2013

name	value	description
		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_evening_no-prop_data	10	Evening data non propagated Use results from evening margin calculations, on non-propagated position level Applicable for RQ2055 only
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 and RQ2073 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, RQ222, RQ2055, RQ2057, RQ2070 and RQ2073 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_day3_c (Intra Day3)	
Datatype	UINT8_T
Description	Defines from which margin calculation result should be fetched.

Value Set

name	value	description
Evening data	0	Evening data Use results from evening margin calculations.
intra day data	1	intra day data

© The NASDAQ OMX Group, Inc. • 2013

name	value	description
		Use results from latest available intra day margin calculations.
preliminary evening data	3	preliminary evening data Use results from calculation of preliminary evening prices.

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.
invc_text_s (Invoice Text)	
Datatype	char[60]
Description	Text describing the manual fee.
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; ISIN Code of delivered underlying)	
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.
issued_price_u (Issued Price)	
Datatype	UINT32_T
Description	Defined the issued price for the underlying with three implicit decimals.

© The NASDAQ OMX Group, Inc. • 2013

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_apply_spread_rule_n (Apply spread rule)	
Datatype	UINT16_T
Description	Apply Spread Rule for margin collect and CDB param for spread rules
Value Set	**name** / **value** TRUE / 1 FALSE / 0
is_direct_debit_c (Is Direct Debit)	
Datatype	UINT8_T
Description	Sets if the collateral transaction is the result of a direct debit.
Value Set	**value** / **description** 1 / Yes 2 / No
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_final_c (Final, Is)	
Datatype	UINT8_T
Description	Is the action taken the final or not.
Value Set	**name** / **value** Yes / 1 No / 2
is_fractions_c (Fraction, Premium)	
Datatype	CHAR

© The NASDAQ OMX Group, Inc. • 2013

Description	Is the premium internally represented as fractions?
Value Set	**name** / **value** Yes / Y No / N
is_intraday_c (Intraday, Is)	
Datatype	UINT8_T
Description	Is the action taken intraday or not.
Value Set	**name** / **value** Yes / 1 No / 2
is_preliminary_c (Is Preliminary)	
Datatype	UINT8_T
Description	Specifies if the prices received are preliminary or definitive.
Value Set	**name** / **value** Definitive / 0 Preliminary / 1
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)	

© The NASDAQ OMX Group, Inc. • 2013

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 Item is a bought delivery
item_type_sold_delivery	6	Sold delivery Item is a sold delivery

ixv_id_s (IXV_ID_S)	
Datatype	char[16]
Description	Index Value Id
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

© The NASDAQ OMX Group, Inc. • 2013

lag_in_index_n (LAG_IN_INDEX_N)	
Datatype	UINT16_T
Description	Number of month the index is lagging
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
last_trade_report_price_i (Price, Last Trade Report)	
Datatype	INT32_T
Description	The price of the last trade report for the instrument.
last_trade_report_qty_u (Quantity, Last Trade Report)	
Datatype	INT64_T
Description	The quantity of the last trade report for the instrument.
lead_manager_country_id_s (Lead Manager, Country)	
Datatype	char[2]
Description	The exchange identity that together with Lead Manager, Customer represents the lead manager.
lead_manager_ex_customer_s (Lead Manager, Customer)	
Datatype	char[5]

© The NASDAQ OMX Group, Inc. • 2013

Description	This field together with Lead Manager, Country, identifies the member/participant that represents the lead manager.
leg_number_c (Leg Number)	
Datatype	UINT8_T
Description	Member or Party leg.
Value Set	**name** / **value** None / 0 Member leg / 1 Party leg / 2
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 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.

© The NASDAQ OMX Group, Inc. • 2013

name	value	description
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_heading_s (List heading)	
Datatype	char[64]
Description	Defines the name of the list heading.
list_name_s (Name, List)	
Datatype	char[40]
Description	List file name
list_type_c (List type)	
Datatype	UINT8_T
Description	Defines the type of the turnover list.
Value Set	

name	value
Convertibles	1

name	value
Bonds	2
Lottery bonds	3
Derivatives	4

loan_number_s (Loan Number)	
Datatype	char[9]
Description	Defines the loan number for the underlying.
long_adjustment_i (Long Adjustment)	
Datatype	INT32_T
Description	The number of contracts to net.
long_free_text_s (Free Text, Long)	
Datatype	char[64]
Description	Specifies a free text field for the underlying.
long_high_i (Long, High)	
Datatype	UINT32_T
Description	Margin value for a long position at a given valuation point at high volatility.
long_ins_id_s (Series Name, Long)	
Datatype	char[32]
Description	Defines an additional instrument information to an instrument series.
long_low_i (Long, Low)	
Datatype	UINT32_T
Description	Margin value for a long position at a given valuation point at low volatility.
long_middle_i (Long, Middle)	
Datatype	UINT32_T
Description	Margin value for a long position at a given valuation point at middle volatility.
long_name (LONG_NAME)	
Datatype	char[32]
Description	Intermediate field.
long_opt_min_val_q (Long Option Minimum Value)	
Datatype	INT64_T
Description	Margin component, long option minimum value.
long_underlying_id_s (Long Underlying Id)	
Datatype	char[32]
Description	Specifies an additional the long name for the underlying.
lot_type_c (Lot, Type)	
Datatype	UINT8_T

© The NASDAQ OMX Group, Inc. • 2013

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_aggregation_type_c (Margin Aggregation Type)	
Datatype	UINT8_T
Description	Margin aggregation type.
Value Set	**name** / **value** / **description** None / 0 / None ReqFromMarginReqAcc / 1 / Requirement, Margin Requirement Account ReqFromMarginCalcAcc / 2 / Requirement, Margin Calculation Account ReqFromPosAcc / 3 / Requirement, Position Account PosFromMarginCalcAcc / 4 / Position, Margin Calculation Account

© The NASDAQ OMX Group, Inc. • 2013

name	value	description
PosFromPosAcc	5	Position, Position Account

margin_calculation_type_c (Margin calculation type)	
Datatype	UINT8_T
Value Set	

name	value
Undefined	0
Margin Calc	1
Margin And Pos Calc	2
Pos Calc	3

margin_class_filter_c (Margin Class Filter)	
Datatype	UINT8_T
Description	How to filter for margin class
Value Set	

name	value
None	1
Specific	2
RelevantForMe	3
All	4
Default	5

margin_class_s (Margin class)	
Datatype	char[3]
Description	Margin Class id
margin_date_s (Margin Date)	
Datatype	char[8]
Description	The margin date used in the selected valuation. Format: YYYYMMDD
margin_default_fund_q (Margin Default Fund)	
Datatype	INT64_T
Description	Member deposit of type Default Fund. The number of decimals equals decimals in premium price of currency.
margin_extraordinary_q (Margin Extraordinary)	
Datatype	INT64_T
Description	Member deposit of type Extraordinary. The number of decimals equals decimals in premium price of currency.
margin_maintenance_q (Margin Maintenance)	
Datatype	INT64_T

© The NASDAQ OMX Group, Inc. • 2013

Description	Member deposit of type Maintenance. The number of decimals equals decimals in premium price of currency.
margin_mutual_fund_q (Margin Mutual Fund)	
Datatype	INT64_T
Description	Member deposit of type Mutual Fund. The number of decimals equals decimals in premium price of currency.
margin_one_long_q (Margining Requirements, One Short Position)	
Datatype	INT64_T
Description	Margin Requirements for one short position. The field contains an integer.
margin_one_short_q (Margining Requirements, One Short Position)	
Datatype	INT64_T
Description	Margin Requirements for one short position. The field contains an integer.
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_ratio_i (Margin Ratio)	
Datatype	INT32_T
Description	Margin ratio is a premium or a haircut added to the cash rate to reflect the credit worthiness of the counterparty
margin_requirement_q (Margin Requirement Normal)	
Datatype	INT64_T
Description	The amount required at normal risk. The number of decimals equals decimals in premium price of currency.
margin_req_u (Margin Requirements)	
Datatype	INT64_T
Description	The margining requirements needed as security.
margin_sequence_nbr_u (Unique identifier for a margin calculation batch run.)	
Datatype	UINT32_T
margin_time_s (Margin Time)	
Datatype	char[6]
Description	The margin time used in the selected valuation. Margin time is significant only for intraday valuations. Format: HHMMSS
margin_total_q (Margin Total)	
Datatype	INT64_T
Description	This is the total of all margin requirements, including any fixed margin, to be covered for an account. The number of decimals equals decimals in premium price of currency.
marg_call_nbr_n (Margin call number)	
Datatype	UINT16_T

© The NASDAQ OMX Group, Inc. • 2013

Description	Intra-day margin call number.
marg_item_type_c (Margin item type)	
Datatype	UINT8_T
Description	Indicates type of margin for an item
Value Set	

name	value
Spot	1
Forward	2
Future	3
Option	4
Delivery	5
Payment	6
Risk Neutral Position	7
Power Delta Hedge Payment	8

marg_meth_inst_c (Margin method, for instrument class and instrument series)	
Datatype	UINT8_T
Value Set	

name	value
Not set	0
Delta Hedge	1
OMS2	2
Cash flow margin	3
Externally calculated	5
No margin	6
Power Delta Hedge	7
FX margin	8

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_theo_c (Margin, Settlement Price Theoretical Mark)	
Datatype	UINT8_T

© The NASDAQ OMX Group, Inc. • 2013

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

© The NASDAQ OMX Group, Inc. • 2013

value	description
2	Fixed Income
3	Currency
4	Power/Energy
5	Commodity
6	Payment
7	Index
8	General

market_value_margin_settled_q (Market value margin settled)	
Datatype	INT64_T
Description	Market value margin settled
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_equilib-rium	1	matching_price_type_equilib-rium 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

© The NASDAQ OMX Group, Inc. • 2013

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.
max_order_size_q (Max Order Size)	
Datatype	INT64_T
Description	Specifies the maximum quantity that is allowed to enter ny the users connected to the Pre-trade limit.
mbs_id_s (Minimum Bid Schedule)	
Datatype	CHAR[2]
Description	Not applicable.
median_ask_price_i (Price, Median Ask)	
Datatype	INT32_T
Description	Defines the current median ask price.
median_bid_price_i (Price, Median Bid)	
Datatype	INT32_T
Description	Defines the current median bid price.
member_circ_numb_s (Member, Circular Number)	
Datatype	char[4]
Description	Not applicable.
member_deposit_type_c (Member_Deposit_Type)	
Datatype	UINT8_T
Description	Defines the type of member deposit.

© The NASDAQ OMX Group, Inc. • 2013

Value Set	

name	value
Initial_Maintenance_Margin	1
Extraordinary_Margin	2
Default_Fund	3
Mutual_Fund	4
Default_Fund_Add_On	5
Base_Collateral	6

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_id_q (Message, Identity)	
Datatype	UINT64_T
Description	Identification value that uniquely defines a Broker to Broker 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

© The NASDAQ OMX Group, Inc. • 2013

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.

message_type_s (Message Type)	
Datatype	char[3]
Description	The message type identifies the variant of XvY for the object.

Value Set

name	value
Delivery versus payment, also DaP, Delivery and payment, from the sign of the quantity. Derivation: both qty:s non-zero; one series is currency; one series is non-currency.	DvP
Delivery versus delivery. Derivation: both qty:s non-zero; both series non-currency.	DvD
Free of payment. Derivation: Only one qty non-zero and that qty's series is non-currency.	FoP
Payment versus payment. Derivation: both series non-zero and currencies.	PvP
Payment versus nothing (or FOD - free of delivery). Derivation: Only one qty non-zero and that qty's series is currency.	PvN
Recall of a DvP instruction. Done by external transaction.	Rec

method_dealt_s (Method)	
Datatype	char[16]
Description	Method dealt

mic_code_s (MIC Code)	
Datatype	char[8]
Description	Specifies the MIC Code for the market.

mid_marg_vol_i (Margin, Volatility Mid)	
Datatype	INT32_T
Description	Implied volatility based on mid price for an option. Expressed in percent. 4 implicit decimals

minimum_size_n (Block Size, Minimum Volume)	
Datatype	UINT32_T
Description	The minimum volume required for the order per block size.

© The NASDAQ OMX Group, Inc. • 2013

	Note! A value of 0 means no limit.
min_hold_time_n (Min lifetime of placed quote(sec))	
Datatype	UINT16_T
Description	Min lifetime of placed quote(sec)
min_itm_n (Number of ITM for single supervision)	
Datatype	UINT16_T
min_num_days_n (Minimum number of days)	
Datatype	INT16_T
Description	Minimum number of days between calibration instruments.
min_num_nodes_n (Mininum number of Nodes)	
Datatype	INT16_T
Description	Minimum number of nodes (calibration instruments with a price) required to bootstrap a curve.
min_otm_n (Number of OTM for single supervision)	
Datatype	UINT16_T
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.
min_vol_n (Minimum volume required)	
Datatype	INT32_T
mmsup_status_u (Alarm, Type)	
Datatype	UINT32_T
Description	This field describes the reason of a market maker alarm.
Value Set	

value	description
1	Prices are missing.
2	BID price is missing and ASK Qty too Small.
3	BID price is missing.
4	ASK price is missing and BID Qty too Small.
5	ASK price is missing.
6	The price spread is too big and both ASK Qty and BID Qty are too Small.
7	The price spread is too big and ASK Qty too Small.
8	The price spread is too big and BID Qty too Small.

© The NASDAQ OMX Group, Inc. • 2013

value	description
9	The price spread is too big.
10	Quantities are too Small.
11	BID quantity is too Small.
12	ASK quantity is too Small.

mm_resp_type_c (Market Maker, Type)	
Datatype	CHAR
Description	Market Maker Resp Type

Value Set

name	value
Quotation	Q
on Request	R
Both	B

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.

money_or_par_c (Money or Par)	
Datatype	UINT8_T
Description	Money or Par filled repo

Value Set

name	value
Money	1
Par	2

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

© The NASDAQ OMX Group, Inc. • 2013

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.
naked_risk_margin_q (Naked Risk Margin)	
Datatype	INT64_T
Description	Informational field, naked risk margin.
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; NT User 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_days_to_exp_n (Number of cycles or calendar days)	
Datatype	UINT16_T
nbr_held_q (Held)	
Datatype	INT64_T
Description	Number of held (long) contracts
nbr_of_strk_n (Number of strikes for coupled supervision)	
Datatype	UINT16_T
nbr_written_q (Written)	

© The NASDAQ OMX Group, Inc. • 2013

Datatype	INT64_T
Description	Number of written (short) contracts
netting_req_nbr_u (Netting request number)	
Datatype	UINT32_T
Description	Netting request number.
net_open_interest_q (Net Open Interest)	
Datatype	UINT64_T
Description	Defines the net open interest.
net_price_for_settlement_i (Net Price for Settlement)	
Datatype	INT32_T
Description	The net price used when calculating settlement price in an one-sided auction.
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.
new_deal_price_i (Price, New Deal)	
Datatype	INT32_T
Description	Defines the new deal price on a rectified deal..
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

© The NASDAQ OMX Group, Inc. • 2013

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.
note_name_s (Note name)	
Datatype	char[15]
Description	Content defined by Group settlement list parameter on Participant
notional_amount_q (Notional amount)	
Datatype	INT64_T
Description	Notional amount
not_breach_lvl_n (Notification Breach Level)	
Datatype	INT16_T
Description	Specifies the percentage of the limits when notification emails can be sent.
not_email_addr_s (Notification email address)	
Datatype	char[128]
Description	Defines a kist of email addresses where to send warning notifications when the level are breached.
novation_c (Novation)	
Datatype	UINT8_T
Description	Defines the novation options.
Value Set	**name** / **value**: Yes / 1; No / 2
novation_sequence_nbr_u (Novation sequence number)	

© The NASDAQ OMX Group, Inc. • 2013

Datatype	UINT32_T
Description	Specified the novation sequence number
no_bid_quote_req_i (No bid quote required if ask price below)	
Datatype	UINT32_T
Description	No bid quote required if ask price below.
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.
no_of_sub_n (Substitution, Max Number)	
Datatype	UINT16_T
Description	Maximum allowed number of substitutions
ntd_id_s (Non-trading Days, Identity)	
Datatype	char[5]
Description	Defines the identity of holiday table.
number_of_deals_u (Deals, Number)	
Datatype	UINT32_T
Description	Number of deals executed.
number_of_orders_n (Number of orders)	
Datatype	UINT16_T
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>
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_account_reports_c (Only Account Reports)	
Datatype	UINT8_T
Description	Return only account reports, sets if only account reports should be returned.
Value Set	<table><tr><th>value</th><th>description</th></tr><tr><td>1</td><td>True</td></tr><tr><td>2</td><td>False</td></tr></table>

© The NASDAQ OMX Group, Inc. • 2013

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
only_traded_c (Traded series only)	
Datatype	UINT8_T
Description	Specifies if only traded series should be returned.
Value Set	**name** / **value** All series / 0 Only tradeable series / 1
on_behalf_of_type_c (On Behalf of Type)	
Datatype	UINT8_T
Description	Specifies if the query should return participants with trade on behalf, trade report on behalf or both trade and trade report on behalf, given to the querying participant. Any value different from 1 or 2 will return only participants with trade on behalf rights.
Value Set	**name** / **value** Trade Report on Behalf / 1 Both Trade and Trade Report on Behalf / 2
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

© The NASDAQ OMX Group, Inc. • 2013

Description	The number of outstanding contracts (not updated during the day).
open_buy_q (Open Buy)	
Datatype	INT64_T
Description	Specifies the maximum allowed quantity of buy orders in the market.
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 OPEN_CLOSE_REQ_MND_CLOSE / 3 / Mandatory close OPEN_CLOSE_REQ_RESET / 4 / Set to default to the account (valid only for alter order)
open_contract_c (Open Contract)	
Datatype	UINT8_T
Description	Open Contract search criteria.
Value Set	**name** / **value** None / 0 At Call (Repo) / Yes (Swap) / 1 Fixed (Repo) / 2 All (Repo) / 3
open_sell_q (Open Sell)	
Datatype	INT64_T
Description	Specifies the maximum allowed quantity of sell orders in the market.

© The NASDAQ OMX Group, Inc. • 2013

<table>
<tr><td colspan="2">operation_c (Operation)</td></tr>
<tr><td>Datatype</td><td>UINT8_T</td></tr>
<tr><td>Description</td><td>Used for two purposes:
1. Tells if the Rectify Deal is a Delete part, Create part or combined.
2. Defines the operation in external write transactions.
3. Logout request. Only value Logout is allowed.</td></tr>
<tr><td>Value Set</td><td>
<table>
<tr><th>value</th><th>description</th></tr>
<tr><td>1</td><td>Delete
Purpose 1</td></tr>
<tr><td>2</td><td>Create
Purpose 1</td></tr>
<tr><td>3</td><td>Delete and Create
Purpose 1</td></tr>
<tr><td>1</td><td>Add
Purpose 2</td></tr>
<tr><td>2</td><td>Change
Purpose 2</td></tr>
<tr><td>3</td><td>Delete
Purpose 2</td></tr>
<tr><td>2</td><td>Logout
Purpose 3</td></tr>
</table>
</td></tr>
<tr><td colspan="2">operation_type_s (Operation Type)</td></tr>
<tr><td>Datatype</td><td>char[4]</td></tr>
<tr><td>Value Set</td><td>
<table>
<tr><th>name</th><th>value</th></tr>
<tr><td>Normal DvP instruction</td><td>[blank]</td></tr>
<tr><td>Maturity payment</td><td>MATP</td></tr>
<tr><td>SFE transaction, special cash payment for margin requirement.</td><td>SFET</td></tr>
<tr><td>Coupon Payment</td><td>CPON</td></tr>
<tr><td>Payment</td><td>PAYM</td></tr>
</table>
</td></tr>
<tr><td colspan="2">opra_indicator_c (OPRA Indicator)</td></tr>
<tr><td>Datatype</td><td>CHAR</td></tr>
<tr><td>Description</td><td>Not applicable.</td></tr>
<tr><td colspan="2">option_style_c (Option, Style)</td></tr>
<tr><td>Datatype</td><td>UINT8_T</td></tr>
<tr><td>Description</td><td>Defines the style of the option.</td></tr>
</table>

© The NASDAQ OMX Group, Inc. • 2013

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 NASDAQ OMX Group, Inc. • 2013

	The value is defined as quantity*price*price quotation factor.
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.
Bachelier	8	Bachelier Model
Asian	9	Asian Option Model

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. This is the future/forward with the same country, market, underlying and expiration as the option.
Synthetic Future	4	Synthetic future

opt_val_margin_q (Options Value Margin)	
Datatype	INT64_T
Description	Margin component, options value margin.
op_if_buy_c (Operation if Buy)	
Datatype	CHAR

© The NASDAQ OMX Group, Inc. • 2013

Description	Specifies whether to buy or sell the Series when buying the combination.
Value Set	**value** / **description** B / Buy S / Sell
op_if_sell_c (Operation if Sell)	
Datatype	CHAR
Description	Specifies whether to buy or sell the Series when buying the combination.
Value Set	**value** / **description** B / Buy S / Sell
order_capacity_c (Order Capacity)	
Datatype	UINT8_T
Description	Defines the.owner capacity for orders and trades.
Value Set	**name** / **value** Not applicable / 0 Agent / 1 Principal / 2 Acting as Market Maker or Specialist / 3 Issuer Holding / 4 Issue Price Stabilization / 6 Riskless Principal / 7
order_category_c (Order Category)	
Datatype	UINT8_T
Description	Defines the order category.
Value Set	**name** / **value** Undefined / 0 Firm Order/Quote / 1 Indicative Order/Quote / 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)	

© The NASDAQ OMX Group, Inc. • 2013

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_rate_limit_i (Order Rate Limit)	
Datatype	INT32_T
Description	Specifies number of allowed new orders during one second.
order_reference_s (Order Reference)	
Datatype	char[10]
Description	Enables a user to send a broker firm internal order reference that is passed through the system.
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

© The NASDAQ OMX Group, Inc. • 2013

name	value	description
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
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)

© The NASDAQ OMX Group, Inc. • 2013

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_deal_number_i (Deal Number, Original)	
Datatype	INT32_T
Description	Original dealnumber. Differs from Deal number if Deal is rectified.
orig_dvp_sequence_number_u (ORIG_DVP_SEQUENCE_NUMBER_U)	
Datatype	UINT32_T
orig_ext_trade_number_u (Trade Number, Original External)	
Datatype	UINT32_T
Description	Original trade number assigned by external system.
orig_flow_number_end_u (Original Flow Number, End Date)	
Datatype	UINT32_T
Description	Original flow number for end date of this SWAP flow
orig_flow_number_start_u (Original Flow Number, Start Date)	
Datatype	UINT32_T
Description	Original flow number for start date of this SWAP flow
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)	

© The NASDAQ OMX Group, Inc. • 2013

Datatype	UINT8_T
Description	Defines the original trade type, for further description see Trade Type.
otc_cash_flow_type_c (OTC cash flow type)	
Datatype	UINT8_T
Description	Describes the source of the cash flow.
Value Set	**name** / **value** Swap Flow / 1 Upfront payment / 2 Termination payment / 3
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 require-ments / 1 / Only sum margin require-ments Level 1 and margin results per series / 2 / 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
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
outstanding_amount_q (Outstanding Amount)	
Datatype	INT64_T
Description	The outstanding amount for the underlying.
overlap_pc1_n (Overlap, PC1)	
Datatype	UINT16_T
Description	Size of the allowed overlap in PC1 when correlating yield curves (or middle level curve correlation cubes) belonging to this curve correlation cube. Given in number of nodes

© The NASDAQ OMX Group, Inc. • 2013

overlap_pc2_n (Overlap, PC2)	
Datatype	UINT16_T
Description	Size of the allowed overlap in PC2 when correlating yield curves (or middle level curve correlation cubes) belonging to this curve correlation cube. Given in number of nodes
overlap_pc3_n (Overlap, PC3)	
Datatype	UINT16_T
Description	Size of the allowed overlap in PC3 when correlating yield curves (or middle level curve correlation cubes) belonging to this curve correlation cube. Given in number of nodes
overnight_index_swap_c (OIS Overnight index swap)	
Datatype	UINT8_T
Description	Specifies if the instrument group is used for Overnight Index Swaps.
Value Set	**name** / **value** Yes / 1 No / 2
own_inventory_c (Own Inventory)	
Datatype	UINT8_T
Description	Is the account an own inventory account?
Value Set	**value** / **description** 1 / Yes 2 / No
participant_info_s (Participant Info)	
Datatype	char[80]
Description	Information about the participant.
party_account_id_s (Cash transfer group)	
Datatype	char[10]
Description	Cash transfer group.
party_condition_confirmed_c (Party Condition Confirmed)	
Datatype	UINT8_T
Description	Signal if counterparty's conditions have been confirmed
Value Set	**name** / **value** No condition specified / 0 Confirmation needed / 1 Confirmed / 2
party_csd_code_s (CSD code, Counterpart)	

© The NASDAQ OMX Group, Inc. • 2013

Datatype	char[34]
Description	Identifies the CSD account number for the counterpart.
party_swap_condition_s (Party swap condition)	
Datatype	char[256]
Description	Swap condition for party in swap trade
party_trade_report_nbr_q (Party trade report number)	
Datatype	UINT64_T
Description	Trade report number for party trade.
part_collect_date_s (Partial collect date)	
Datatype	char[8]
Description	Timestamp together with part_collect_time_s when a partial collect was done
part_collect_time_s (Partial collect time)	
Datatype	char[6]
Description	Timestamp together with part_collect_date_s when a partial collect was done
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.
payment_date_s (Date, Payment)	
Datatype	char[8]
Description	Payment date. Format: YYYYMMDD.
payment_margin_future_date_q (Payment margin future date.)	
Datatype	INT64_T
Description	Payment margin for settlement settled on future dates > valuation date. The number of decimals equals decimals in premium price of currency.
payment_margin_overdue_q (Overdue payment margin.)	
Datatype	INT64_T
Description	Overdue payment margin due to unpaid settlement amounts. The number of decimals equals decimals in premium price of currency.
payment_margin_valuation_date_q (Payment margin valuation date.)	
Datatype	INT64_T
Description	Payment margin for settlement settled on valuation date. The number of decimals equals decimals in premium price of currency.
payment_notional_amount_q (Payment notional amount)	
Datatype	INT64_T
Description	Payment notional amount for swap
payment_q (Payment)	
Datatype	INT64_T

© The NASDAQ OMX Group, Inc. • 2013

Description	Payment for swap
payment_settlement_c (Payment settled by CSD Yes/ No)	
Datatype	UINT8_T
Description	Payment settled by CSD Yes (1)/ No (2)
Value Set	**name** / **value** Yes / 1 No / 2
payment_set_c (Payment Set)	
Datatype	UINT8_T
Description	Decides if payment should occur in the beginning or in the end of a period.
Value Set	**name** / **value** First / 1 Last / 2
payment_status_s (Payment status)	
Datatype	char[6]
Description	Status for the payment. Ex ANEW, INVC, PAID
pay_amount_q (Pay Amount)	
Datatype	INT64_T
Description	The amount to be payed, differens between float and fixed consideration
pay_calc_req_nbr_u (Pay calc request number)	
Datatype	UINT32_T
Description	Payment calculate request number
pay_margin_q (Payment Margin)	
Datatype	INT64_T
Description	Defines the payment margin.
pay_note_number_i (Pay note number)	
Datatype	INT32_T
Description	Paynote number, Settlement
pay_or_receive_c (Deliver/Pay or Receive)	
Datatype	UINT8_T
Description	Deliver/Pay or Receive?
Value Set	**value** / **description** 1 / Deliver securities or money 2 / Receive securities or money

© The NASDAQ OMX Group, Inc. • 2013

pc1_q (Principal Component, First)	
Datatype	INT64_T
Description	Value for PC1 for a given maturity
pc2_q (Principal Component, Second)	
Datatype	INT64_T
Description	Value for PC2 for a given maturity
pc3_q (Principal Component, Third)	
Datatype	INT64_T
Description	Value for PC3 for a given maturity
pc_years_n (Principal component, Years)	
Datatype	INT16_T
Description	Years to Maturity for principal component
percentile_for_margin_i (Percentile for margin)	
Datatype	UINT32_T
Description	Percentile for margin, 2 implicit decimals
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.
point_i (Point number)	
Datatype	UINT32_T
Description	Margin vector point number.
point_no_pc1_i (Point number for PC1)	
Datatype	INT32_T
Description	A point number for PC1 ranging from -(number of nodes)/2 to +number of nodes)/2.
point_no_pc2_i (Point number for PC2)	
Datatype	INT32_T
Description	A point number for PC2 ranging from -(number of nodes)/2 to +number of nodes)/2.
point_no_pc3_i (Point number for PC3)	
Datatype	INT32_T

© The NASDAQ OMX Group, Inc. • 2013

Description	A point number for PC3 ranging from -(number of nodes)/2 to +number of nodes)/2.
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 Use real time position / 1 / 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 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

name	value	description
		as 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 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 for the account specified in the Account field, totgether with trades specified in the query
Use real time position margin requirement account	6	Use real time position, result on margin requirement account level.
Use start of day position margin requirement account	7	Use start of day positions, result on margin requirement account level.

pos_unit_id_q (POS_UNIT_ID_Q)	
Datatype	INT64_T

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
Description	The modifier is used to recalculate the item after an underlying adjustment. The field is stored with 7 implicit decimals

© The NASDAQ OMX Group, Inc. • 2013

preliminary_amount_ca_adjusted_q (Preliminary Collateral Balance or Holding after corp action adjustment.)	
Datatype	INT64_T
Description	Decimals according to dec_in_amount_n.
preliminary_amount_q (Preliminary Collateral Balance or Holding adjusted for not yet settled collateral withdraw requests.)	
Datatype	INT64_T
Description	Decimals according to dec_in_amount_n.
premium_i (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).
premium_levels_c (Premium Levels)	
Datatype	UINT8_T
Description	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.
prev_clearing_date_s (Clearing Date, Previous)	
Datatype	char[8]
Description	Date in ASCII for clearing trade, format is YYYYMMDD.
pre_novation_collateral_check_c (Pre novation collateral check)	
Datatype	UINT8_T
Description	Sets if the instrument type should be subject for collateral checks before deals are accepted for clearing. Deal will get Trade Report State Matched and Trade Report Sub State Pending Clearinghouse Confirmation in OTC_RTS, until check has been made. After check, deal is either accepted for clearing, i.e. Trade Report State is Novated or kept in state Matched with a Trade Report Sub State indicating which part of the check that failed.
Value Set	**value** / **description** 1 / Yes 2 / No
pre_trade_limit_param_unit_c (Pre Trade Limit Param Unit)	
Datatype	UINT8_T
Description	Defines the unit the limits are defined in.

© The NASDAQ OMX Group, Inc. • 2013

Value Set	**name** / **value** Quantity / 1 Volume / 3
price (PRICE)	
Datatype	INT32_T
Description	Intermediate field.
price_change_i (Price change)	
Datatype	INT32_T
Description	Define price change
price_currency_s (Currency, Price)	
Datatype	char[3]
Description	The currency in which an exchange rate is defined.
price_format_c (Premium/Price Format)	
Datatype	UINT8_T
Description	Not applicable.
price_i (Price)	
Datatype	INT32_T
Description	Price
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	**value** / **description** 1 / Yes 2 / No
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	**name** / **value** / **description** Use real time prices / 0 / Use real time prices

© The NASDAQ OMX Group, Inc. • 2013

name	value	description
Use real time prices some ignored	1	Use real time prices special 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.
Use real time prices frozen	2	Use real time prices frozen 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.
Use prices previously frozen	3	Use prices previously frozen for the user sending the query.
Use start of day prices	4	Use start of day prices
Use official end of day prices	5	Use official end of day prices
Use private price list	6	Use private price list for the user sending the query.
Last intraday or EOD Run	7	Use prices from last official margin run

price_spread_margin_q (Price Spread Margin)	
Datatype	INT64_T
Description	Spread contribution to margin requirement.

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

© The NASDAQ OMX Group, Inc. • 2013

value	description
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
11	Volatility

price_unit_strike_c (Price Unit, Strike)	
Datatype	UINT8_T
Description	The strike price unit for the class 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

pricing_method_c (Pricing method)	
Datatype	UINT8_T
Description	Specifies the pricing method used for the combo type.

Value Set

name	value
Not applicable	0
Net price	1
Net value	2

© The NASDAQ OMX Group, Inc. • 2013

primary_ccc_id_s (Primary Curve Correlation Cube)	
Datatype	char[12]
Description	Name of Curve Correlation Cube applicable for primary curve
primary_crv_id_s (Primary Curve Id)	
Datatype	char[12]
Description	Curve Stressing objects (see struct STRESS_CRV_ID_S)
principal_exchange_c (Principal Exchange)	
Datatype	UINT8_T
Description	Principal exchange denotes whether the notional amounts of each leg will be exchanged.
Value Set	**name** / **value** No principal exchange / 0 Initial only / 1 Final only / 2 Both / 3
principal_exchange_date_s (Principal Exchange Date)	
Datatype	char[8]
Description	Date when exchange of principal takes place
private_match_field_s (Private match field)	
Datatype	char[52]
Description	A string used as a private match criteria agreed by the parties when sending in a Trade Report.
private_price_list_cmd_c (Private price list command)	
Datatype	UINT8_T
Description	Command for private price list.
Value Set	**name** / **value** Rolling full / 1 Rolling partial / 2 Start of day / 3 Evening prices / 4
private_price_list_src_c (Private price list source)	
Datatype	UINT8_T
Description	Source for private price list.
Value Set	**name** / **value** None / 0 Rolling / 1 Start of day / 2

© The NASDAQ OMX Group, Inc. • 2013

	name / **value** Evening / 3
pri_not_s (Notation, Primary)	
Datatype	char[5]
Description	The currency primary notation, e.g. $.
pri_unit_s (Unit, Primary)	
Datatype	char[15]
Description	Primary Unit. The currency unit, e.g. DOLLAR, CENT.
prod_area_c (Product Area, RIVA)	
Datatype	UINT8_T
Description	Define the RIVA product area.
prod_area_text_s (Product Area Text, RIVA)	
Datatype	char[10]
Description	Description of a product area in ASCII.
program_trader_c (Program Trader)	
Datatype	UINT8_T
Description	Defines if the User is a program trader ot not:
Value Set	**value** / **description** 1 / Yes 2 / No
propagated_margin_position_c (PROPAGATED_MARGIN_POSITION_C)	
Datatype	UINT8_T
Description	Result is for Propagated Margin position. If False, result is for Non-propagated positions
Value Set	**name** / **value** True / 1 False / 2
propagation_u (Propagation)	
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 /

© The NASDAQ OMX Group, Inc. • 2013

name	value	description
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
6	Call
7	Delivery
8	Intraday Funding
9	Base Collateral

protect_coupon_c (PROTECT_COUPON_C)	
Datatype	UINT8_T
Description	Protect index from beeing negative for coupons

Value Set

name	value
Yes	1
No	2

protect_redempt_c (PROTECT_REDEMPT_C)	
Datatype	UINT8_T
Description	Protect index from beeing negative for redempt.

Value Set

name	value
Yes	1
No	2

ptl_id_s (Pre Trade Limit Identity)	
Datatype	char[32]
Description	A unique identity for the Sponsoring Participant and the connected sponsored clients. The naming standard will be "Sponsored Participant_ClientParticipant_free-text".

© The NASDAQ OMX Group, Inc. • 2013

<table>
<tr><td colspan="2">ptl_suffix_s (Pre Trade Limit Suffix)</td></tr>
<tr><td>Datatype</td><td>char[16]</td></tr>
<tr><td>Description</td><td>This is a free text added last in the generated Pre Trade Limit ID after the sponsoring participant and the sponsored client id.</td></tr>
<tr><td colspan="2">public_deal_information_c (Public Deal Information)</td></tr>
<tr><td>Datatype</td><td>UINT8_T</td></tr>
<tr><td>Description</td><td>Specifies how the post trade public deal information is distributed.</td></tr>
<tr><td>Value Set</td><td><table><tr><th>name</th><th>value</th></tr><tr><td>No information</td><td>0</td></tr><tr><td>Without identity</td><td>1</td></tr><tr><td>With identity</td><td>2</td></tr></table></td></tr>
<tr><td colspan="2">publ_at_end_of_day_c (Publish at End of Day)</td></tr>
<tr><td>Datatype</td><td>UINT8_T</td></tr>
<tr><td>Description</td><td>Instead of specifying Time Delay, the publishing of BD2 and BD70 is triggered at end of day.</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">pub_inf_id_n (Public Order Info)</td></tr>
<tr><td>Datatype</td><td>UINT16_T</td></tr>
<tr><td>Description</td><td>Specifies how order information is distributed</td></tr>
<tr><td>Value Set</td><td><table><tr><th>name</th><th>value</th><th>description</th></tr><tr><td>Without identity</td><td>1</td><td>The order information is distributed with broadcast BO2 and the answer of query MQ7 is without identity.</td></tr><tr><td>With identity</td><td>2</td><td>The order information is distributed with broadcast BO1 and the answer of query MQ7 is with identity.</td></tr><tr><td>Query information without identity</td><td>3</td><td>The answer of MQ7 is without identity. No BO2 generated.</td></tr><tr><td>Query information with identity</td><td>4</td><td>The answer of MQ7 is with identity. No BO1 generated.</td></tr><tr><td>No information</td><td>5</td><td>No MQ7 generated, No BO1 or BO2 generated.</td></tr></table></td></tr>
<tr><td colspan="2">qry_segment_number_n (Segment Number, Query)</td></tr>
<tr><td>Datatype</td><td>UINT16_T</td></tr>
<tr><td>Description</td><td>Defines the segment number in the query.</td></tr>
</table>

© The NASDAQ OMX Group, Inc. • 2013

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_difference_i (Quantity, Difference)	
Datatype	INT64_T
Description	When an existing order (in the OB) is changed regarding the mp_quantity, the difference is stored here (negative if the order volume became lower or positive if higher). Used as a reference.
quantity_i (Quantity)	
Datatype	INT64_T
Description	Defines the quantity.
quantity_limit_q (Quantity limit used for One sided auction)	
Datatype	INT64_T
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_q (Quantity)	
Datatype	INT64_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
query_type_c (Query type)	
Datatype	UINT8_T
Description	Type indicator instrument type=1, all=2
quote_action_c (Quote Action)	
Datatype	UINT8_T
Value Set	**name** / **value** None / 1 Update / 2

© The NASDAQ OMX Group, Inc. • 2013

	name / **value** 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_i (Rate)	
Datatype	INT32_T
Description	Specifies the rate value for the reference rate and date. Given with 4 decimals.
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.
rate_reset_c (Rate Reset)	
Datatype	UINT8_T
Description	Number of business days prior to payment date that rate will be set.
Value Set	**name** / **value** First / 1 Last / 2
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

© The NASDAQ OMX Group, Inc. • 2013

value	description
2	Trade

reason_s (Reason)	
Datatype	char[80]
Description	Text field typically holding the reason for returned status.
reason_u (Reason)	
Datatype	UINT8_T
Description	There are two possible reasons for why this message was sent (and consequently, the original message was not sent): USER_INVALID = 1, USER_LOGGED_OUT = 2
rectify_deal_number_q (Rectify Deal Number)	
Datatype	INT64_T
Description	A number that together with series identifies a specific rectified deal.
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.
reference_price_i (REFERENCE_PRICE_I)	
Datatype	INT32_T
ref_price_i (Price, Reference)	
Datatype	INT32_T
Description	Reference price of the underlying/instrument series.
rejected_date_s (Date, Rejected)	
Datatype	char[8]
Description	Date in ASCII. Format: YYYYMMDD
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:

© The NASDAQ OMX Group, Inc. • 2013

	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_no_i (Report Number)	
Datatype	INT32_T
Description	Each report template is assigned a unique number. This number is used to identify the report.
report_owner_s (Report owner)	
Datatype	char[12]
Description	Name of member or customer that is the owner of the report.
report_spec_s (Report Specification)	
Datatype	char[5]
Description	Specification for which products the report is created for. Appended after the Report File Prefix when generating the report file name.
report_version_s (Report Version)	
Datatype	char[3]
Description	Zero padded sequence number of the report.
repo_category_c (REPO_CATEGORY_C)	
Datatype	UINT8_T
Description	Repo category
Value Set	**name** / **value** Not applicable / 0 Intraday / 1 Fixed / 2 At Call / 3
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

© The NASDAQ OMX Group, Inc. • 2013

request_nbr_u (Request number)	
Datatype	UINT32_T
Description	Unique request number.
reserved_12_s (Reserved)	
Datatype	char[12]
Description	Filler for alignment
reserved_1_c (Reserved)	
Datatype	CHAR
Description	Filler for alignment.
reserved_1_s (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
reset_date_s (Date, Reset)	
Datatype	char[8]
Description	The reset date is the date when the fixing price is set for the floating leg of a SWAP or FRA Format: YYYYMMDD.
reset_days_c (Reset Days)	
Datatype	UINT8_T
Description	Specifies the number of reset days to use for a leg
reset_days_type_c (Reset days type)	

© The NASDAQ OMX Group, Inc. • 2013

Datatype	UINT8_T
Description	The day type for the Reset Days. The business day convention is always following for the reset days.
Value Set	**name** / **value** Trading Days / 1 Calendar Days / 2
residual_i (Residual)	
Datatype	INT32_T
Description	Residual due to rounding in average price trade.
resp_fulfilled_n (Required fulfilled resp. in % with 0 decimals)	
Datatype	UINT16_T
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.
right_type_c (Right type)	
Datatype	UINT8_T
Description	The rights per participant.
Value Set	**name** / **value** Trade on Behalf / 1 Trade Report on Behalf / 2
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

© The NASDAQ OMX Group, Inc. • 2013

value	description
	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_deliv_q (Risk Margin Delivery)	
Datatype	INT64_T
Description	Margin component, risk margin delivery.
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_open_q (Risk Margin Open)	
Datatype	INT64_T
Description	Margin component, risk margin open.
risk_margin_q (Margining Requirements, Risk)	
Datatype	INT64_T
Description	Margin requirement after currency conversion.
risk_margin_scaling_factor_n (Risk margin scaling factor)	
Datatype	INT16_T
Description	Risk margin scaling factor (%) without decimals
risk_scale_s (Risk scale)	
Datatype	char[12]
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

© The NASDAQ OMX Group, Inc. • 2013

value	description
2	Rule 2 1. Inverted Price 2. Time
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

rollover_day_c (Rollover Day)	
Datatype	UINT8_T
Min	1

© The NASDAQ OMX Group, Inc. • 2013

Max	31
Description	The next end date will be the nearest settlement date to this day in month.
rollover_period_c (Rollover Period)	
Datatype	UINT8_T
Description	Length of the rollover period
Value Set	**name** / **value** None / 0 One_Month / 1 Three_Month / 3 Six_Month / 6 Twelve_Month / 12 One_Week / 21
rounding_before_index_c (Rounding before index)	
Datatype	UINT8_T
Description	Specifies if the rounding of the price is done before the index value is multiplied with the price.
Value Set	**name** / **value** Yes / 1 No / 2
run_type_c (Run Type)	
Datatype	UINT8_T
Description	Type of calculation run
Value Set	**name** / **value** All / 0 None / 1 EndOfDay / 2 Intraday / 3 Call / 4 Preliminary / 5
scenario_number_n (Scenario Number)	
Datatype	INT32_T
Description	Define an unsigned sequence number.
secondary_ccc_id_s (Secondary Curve Correlation Cube)	
Datatype	char[12]
Description	Name of Curve Correlation Cube applicable for secondary curve

© The NASDAQ OMX Group, Inc. • 2013

secondary_crv_id_s (Secondary Curve Id)	
Datatype	char[12]
Description	Curve Stressing objects (see struct STRESS_CRV_ID_S)
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
second_dvp_account_s (SECOND_DVP_ACCOUNT_S)	
Datatype	char[24]
second_holiday_id_s (Second State Holiday ID)	
Datatype	char[5]
Description	Second State holiday ID.
second_isin_code_s (SECOND_ISIN_CODE_S)	
Datatype	char[12]
second_quantity_q (Quantity, Second)	
Datatype	INT64_T
sector_code_s (Sector Code)	
Datatype	char[4]
Description	The sector code that the underlying is connected to.
security_account_s (Account, Security)	
Datatype	char[24]
Description	A Security Account (Sub Account) is unique within a Member. Allowed characters are (A-Z), (a-z), (0-9), space and hyphen.
security_type_c (Security Type)	
Datatype	UINT8_T
Description	Not applicable.
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]

© The NASDAQ OMX Group, Inc. • 2013

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_amount_q (Sell Amount)	
Datatype	INT64_T
Description	Defines the sell amount.
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.
sell_si_s (Sell Settlement Instruction)	
Datatype	char[120]
Description	Specifies the sell settlement instruction.
sell_ssi_s (Sell SSI)	
Datatype	char[120]
Description	Sell settlement instruction
sell_use_ssi_c (Sell use ssi)	
Datatype	UINT8_T
Description	If sell trade use SSI, valid values 1,2
sender_alias_s (Sender Alias)	
Datatype	char[50]
Description	Sender Alias specifies a user friendly name of the sender. May be blank.
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)	

© The NASDAQ OMX Group, Inc. • 2013

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_first_u (Sequence First)	
Datatype	UINT32_T
Description	First sequential number in a range.
sequence_last_i (Number, Last Sequential)	
Datatype	INT32_T
Description	Last number in a sequence.
sequence_last_u (Sequence Last)	
Datatype	UINT32_T
Description	Last sequential number in a range.
sequence_no_i (Number, Sequence)	
Datatype	INT32_T
Description	Enumeration of physical deliveries within a synthetic delivery.
sequence_number_i (Sequence Number)	
Datatype	INT32_T
Description	Define a sequence number.
sequence_number_n (Sequence Number)	
Datatype	INT32_T
Description	Define an unsigned sequence number.
sequence_number_u (Sequence Number)	
Datatype	UINT32_T
Description	Define 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

© The NASDAQ OMX Group, Inc. • 2013

Description	Defines how series expiring today should be handled in margin simulation.
Value Set	**name** / **value** / **description** Not included / 0 / Not included Evening mode / 1 / Evening mode This means included only if also included in EndOfDay calculations of today. Intra day mode / 2 / Intra day 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 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.

© The NASDAQ OMX Group, Inc. • 2013

Value Set	

value	description
O	Order
Q	Query
D	Deal
A	Answer (only from the Central System)
I	Information

session_order_n (Session Order)	
Datatype	UINT16_T
Description	Sessions are sorted in time order
settlement_date_q (Date, Settlement)	
Datatype	INT64_T
settlement_date_s (Date, Settlement; Settlement date in CSD)	
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_instruction_s (Settlement instruction)	
Datatype	char[120]
Description	Settlement instruction for trade report
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	

name	value	description
sp_type_query_on_all	1	Apply to all types. For query use only
sp_type_normal	2	Normal
sp_type_fixing	8	Fixing

settlement_product_s (Settlement product)	
Datatype	char[15]
Description	Settlement product configured on Instrument type

© The NASDAQ OMX Group, Inc. • 2013

settlement_requirement_q (Settlement Requirement)	
Datatype	INT64_T
Description	The settlement amount used in a collateral valuation run. The number of decimals equals decimals in premium price of currency.

settlement_type_c (Settlement, Type)	
Datatype	UINT8_T
Description	Specifies if the contract is settled physically, in cash or optional according to EMIR (European Markets Infrastructure Regulation).

Value Set

name	value
Not set	0
Optional	1
Cash	2
Physical	3

settle_class_c (Class Number)	
Datatype	UINT8_T
Description	Defines the class number.

Value Set

name	value	description
Marketplace_fee	1	Marketplace fixed fee
Clearing_fee	2	Clearing variable fee
Tax	3	Tax
Rebate	4	Rebate
Settlement	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
Information	9	Information
Variation_margin	10	Variation margin
Commission	11	Commission
Settlement_intraday_collect	12	Intraday settlement collect
Accrued_interest	13	The interest accrued on cash instruments.
Settlement_dvp_cvr	16	Quantity of underlying used as cover to be delivered
Settlement_odvp_cvr	18	Payment for delivery of cover quantity
Rounding	20	Rounding

© The NASDAQ OMX Group, Inc. • 2013

name	value	description
Balance_adjustment	21	Balance adjustment
Fee3	23	Fee 3
Fee4	24	Fee 4
Fee5	25	Fee 5
Fee6	26	Fee 6
Fee7	27	Fee 7
Fee8	28	Fee 8
Fee9	29	Fee 9
FairValue	30	Fair value
Market_Value_Margin	31	Market_Value_Margin Market Value Margin
Market_Value_Interest	32	Market_Value_Interest Market Value Interest

settle_domestic_currency_c (Settlement Domestic Currency)	
Datatype	UINT8_T
Description	Defines the domestic currency options.
Value Set	

name	value
Yes	1
No	2

settle_foreign_currency_c (Settlement Foreign Currency)	
Datatype	UINT8_T
Description	Defines the foreign currency options.
Value Set	

name	value
Yes	1
No	2

settle_price_i (Price, Settlement)	
Datatype	INT32_T
Description	The daily settlement price for the Series.
settle_status_c (Settlement Status)	
Datatype	UINT8_T
Description	This enumeration is used to assign a DvP instruction a state representing its status towards the CSD.

© The NASDAQ OMX Group, Inc. • 2013

Value Set	**name** / **value** Pending (awaiting acceptance from the CSD) / 1 Holding (will be sent to the CSD in the future) / 2 Completed (accepted and executed by the CSD) / 3 Failed (rejected by the CSD) / 4 Accepted (and not yet executed by the CSD) / 5 Recalled (cancellation of accepted instruction) / 6 Cancelled / 7
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.
set_end_consid_c (Set End Consideration)	
Datatype	UINT8_T
Description	End Consideration
Value Set	**name** / **value** Yes / 1 No / 2
set_start_consid_c (Calculate Settlement Amount)	
Datatype	UINT8_T
Description	Specifies if settlement amount should be calculated in the post trade message.
Value Set	**name** / **value** Yes / 1 No / 2

© The NASDAQ OMX Group, Inc. • 2013

short_code (SHORT_CODE)	
Datatype	CHAR
Description	Intermediate field.
short_high_i (Short, High)	
Datatype	UINT32_T
Description	Margin value for a short position at a given valuation point at high volatility.
short_low_i (Short, Low)	
Datatype	UINT32_T
Description	Margin value for a short position at a given valuation point at low volatility.
short_middle_i (Short, Middle)	
Datatype	UINT32_T
Description	Margin value for a short position at a given valuation point at middle volatility.
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

© The NASDAQ OMX Group, Inc. • 2013

name	value	description
Prices and valuation intervals used in the calculations	7	Prices and valuation intervals used in the calculations 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.

sort_item_n (Sort item)	
Datatype	UINT16_T
Description	Defines the sorting number of the list headings specified in the turnover list.

sort_type_c (Sort Criteria)	
Datatype	UINT8_T
Description	Not applicable.

Value Set

value	description
0	Default

source_id_c (Source for paynote data)	
Datatype	UINT8_T

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

© The NASDAQ OMX Group, Inc. • 2013

	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

© The NASDAQ OMX Group, Inc. • 2013

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

split_rule_c (Split rule)		
Datatype	UINT8_T	
Description	Specifies how the traded quantity is splitted.	
Value Set	**name** / **value** / **description**	
	Even / 1 / The quantity is equally splitted between the month contracts.	
	Preserve / 2 / The quantity given for the strip applies to each individual month contract.	

sponsored_client_country_id_s (Sponsored Client, Country)	
Datatype	char[2]
Description	The exchange identity that together with Sponsored Client, Customer represents the Sponsored Client.

© The NASDAQ OMX Group, Inc. • 2013

sponsored_client_ex_customer_s (Sponsored Client, Customer)	
Datatype	char[5]
Description	This field together with Sponsored, Country, identifies the member/participant that represents the Sponsored Client.
spons_user_name_s (Sponsoring User)	
Datatype	char[32]
Description	Defines the sponsoring user that will be monitored by the system.
spot_i (Spot)	
Datatype	UINT32_T
Description	Margin vector field. The spot price corresponding to the margin requirement.
spot_val_margin_q (Spot Value Margin)	
Datatype	INT64_T
Description	Margin component, spot value margin.
spread_i (Spread)	
Datatype	INT32_T
Description	Specified the spread.
spread_id_s (Max spread id)	
Datatype	char[5]
Description	Max spread id.
spread_unit_c (Spread Unit)	
Datatype	CHAR
Description	Defines the unit of the spread for Max Spread.
Value Set	**name** / **value** / **description** spread unit percent / P / Percentage spread unit absolute / A / Absolute value spread unit ticks / T / Ticks
start_date_s (Date, Start)	
Datatype	char[8]
Description	Start date. Format: YYYYMMDD.
start_time (START_TIME)	
Datatype	INT32_T
start_time_s (Time, Start)	
Datatype	char[6]
Description	Time in ASCII, internal use. Format: HHMMSS
state_c (State)	
Datatype	UINT8_T

© The NASDAQ OMX Group, Inc. • 2013

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.
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.
pending_collateral_check	15	Pending Collateral Check Collateral checks are ongoing.
rejected_collateral_check	16	Rejected Collateral Check Operation rejected due to missing collaterals.

state_i (State, Product)

© The NASDAQ OMX Group, Inc. • 2013

Datatype	INT32_T
Description	Defines the system state of the product.
Value Set	**value** / **description** 0 / None 1 / Business 2 / Close of Business 3 / After Business 4 / Next Business Day 5 / Deleted 6 / Repair
state_item_c (STATE_ITEM_C)	
Datatype	UINT8_T
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 6 / Linked_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.

© The NASDAQ OMX Group, Inc. • 2013

	A higher priority has a higher number.
state_type_name_s (State Type Name)	
Datatype	char[20]
Description	ASCII representation of the State Type.
state_type_number_n (State Type Number)	
Datatype	UINT16_T
Description	Numeric identification of the State Type.
status_description_s (Status Description)	
Datatype	char[100]
Description	Text associated to the message code for System Status.
stat_description_s (Validation Description)	
Datatype	char[80]
Description	Description of validation failure of account field.
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

stress_crv_id_s (Stress Curve Id)	
Datatype	char[12]
Description	Id for Curve Stressing objects STRESS_FACTORS_FOR_YIELD_CURVE
stress_level_pc1_down_q (Stress Level, PC1 down)	
Datatype	INT64_T
Description	Stress Level for stressing PC1 for yield curve down.
stress_level_pc1_up_q (Stress Level, PC1 up)	
Datatype	INT64_T
Description	Stress Level for stressing PC1 for yield curve up.
stress_level_pc2_down_q (Stress Level, PC2 down)	
Datatype	INT64_T
Description	Stress Level for stressing PC2 for yield curve down.
stress_level_pc2_up_q (Stress Level, PC2 up)	
Datatype	INT64_T
Description	Stress Level for stressing PC2 for yield curve up.
stress_level_pc3_down_q (Stress Level, PC3 down)	
Datatype	INT64_T
Description	Stress Level for stressing PC3 for yield curve down.
stress_level_pc3_up_q (Stress Level, PC3 up)	
Datatype	INT64_T
Description	Stress Level for stressing PC3 for yield curve up.
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)	

© The NASDAQ OMX Group, Inc. • 2013

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.
strip_range_c (Strip range)	
Datatype	UINT8_T
Description	Specifies the period of strip instruments.
Value Set	**name** / **value** Annual / 1 Semi Annual / 2 Quarterly / 3
subscription_price_i (Subscription, Price)	
Datatype	INT32_T
Description	Not applicable.
sub_fix_income_type_s (Sub Fixed Income Type)	
Datatype	char[32]
Description	Defines any additional categorization of the Underlying, e.g. Callable or Putable.
sub_settle_status_c (Settlement Sub-status)	
Datatype	UINT8_T
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

© The NASDAQ OMX Group, Inc. • 2013

value	description
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_condition_s (Swap condition)	
Datatype	char[256]
Description	Condition for swap in trade report

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

synthetic_type_c (Type, Synthetic)	
Datatype	UINT8_T
Description	Not Applicable.
Value Set	

value	description
0	Not applicable

© The NASDAQ OMX Group, Inc. • 2013

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
termination_agree_date_s (Termination Agree Date)	
Datatype	char[8]
Description	Date when the termination takes place
termination_info_s (Termination Info)	
Datatype	char[80]
Description	Specified information about this termination.
termination_number_u (Termination Number)	
Datatype	UINT32_T
Description	Specified the number for this termination.
termination_operation_c (Termination Operation)	
Datatype	UINT8_T
Description	Specified the swap termination operation.
Value Set	**name** / **value** Enter / 1 Rectify / 2 Reject / 3 Cancel / 4
termination_search_c (Termination search option)	
Datatype	UINT8_T
Description	Defines if this is a termination history search or a normal search for terminations.
Value Set	**name** / **value** Normal / 1 Termination History / 2
termination_state_c (Termination State)	
Datatype	UINT8_T

© The NASDAQ OMX Group, Inc. • 2013

Description	Enumeration for the different SWAP termination states
Value Set	**name** / **value** Not terminated / 1 Partially terminated / 2 Fully terminated / 3
term_code_s (TERM_CODE_S)	
Datatype	char[12]
Description	Term Code desc. for REPO instruments
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]

© The NASDAQ OMX Group, Inc. • 2013

Description	intermediate field.
timestamp_best_ask (TIMESTAMP_BEST_ASK)	
Datatype	INT32_T
timestamp_best_bid (TIMESTAMP_BEST_BID)	
Datatype	INT32_T
timestamp_comp_s (Time, Computation)	
Datatype	char[5]
Description	A time stamp, "HH.MM".
timestamp_date_s (Timestamp, Date)	
Datatype	char[8]
Description	Timestamp in YYYYMMDD format
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.
timestamp_time_s (Timestamp, Time)	
Datatype	char[6]
Description	Timestamp in Format: HHMMSS.
time_delay_i (Time Delay)	
Datatype	INT32_T
Description	Specifies the time delay in minutes to delay the publishing of the trade statistics broadcast (BD2) and the trade ticker broadcast(BD70).
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.

© The NASDAQ OMX Group, Inc. • 2013

	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_to_maturity_u (Time to maturity)	
Datatype	UINT32_T
Description	Specify Time To Maturity (in months) up to which the given Value after Hair Cut should apply.

© The NASDAQ OMX Group, Inc. • 2013

time_validity_n (Validity Time)	
Datatype	UINT16_T
Description	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.
Value Set	**value** / **description** MSB set to 0 / 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. MSB set to 1 / 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. MSB set to 2 / 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. MSB set to 5 / Days The order will be stored in the order book for the number of days specified in LSB. MSB set to 6 / 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. MSB set to 32 / Good Till Session The order will be stored in the order book until end of the session state type specified in LSB.
tm_series_c (Tailor Made Series)	
Datatype	UINT8_T

Description	Defines if this this is a tailor made series
Value Set	**name** / **value** Yes / 1 No / 2
tm_template_c (Template Series)	
Datatype	UINT8_T
Description	Defines if this a template series used for Tailor Made Series.
Value Set	**name** / **value** Yes / 1 No / 2
today_opt_premium_q (Todays Option Premium)	
Datatype	INT64_T
Description	Margin component, todays option premium.
total_amount_q (Total amount)	
Datatype	INT64_T
total_buy_q (Total Buy)	
Datatype	INT64_T
Description	Specifies the maximum allowed total summary of the quantity of buy orders in the market.
total_collateral_value_q (Total Collateral Value)	
Datatype	INT64_T
Description	The sum of Guarantees, Cash collateral and Securites. The number of decimals equals decimals in premium price of currency.
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_margin_req_q (TOTAL_MARGIN_REQ_Q)	
Datatype	INT64_T
Description	This is the total of all margin requirements, stemming from cleared trades.
total_net_buy_q (Total Net Buy)	
Datatype	INT64_T
Description	Specifies the maximum allowed total net for buy orders which is Traded Net+Open Buy orders
total_net_sell_q (Total Net Sell)	
Datatype	INT64_T
Description	Specifies the maximum allowed total net for sell orders which is Traded Net+Open Sell orders
total_no_of_ask_orders_u (Ask Orders, Total Number)	

© The NASDAQ OMX Group, Inc. • 2013

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_req_balance_account_q (Balance, Total Required)	
Datatype	INT64_T
Description	Total Required Balance On Account. The number of decimals equals decimals in premium price of currency.
total_sell_q (Total Sell)	
Datatype	INT64_T
Description	Specifies the maximum allowed total summary of the quantity of sell orders in the market.
total_surplus_deficit_base_cur_after_fx_haircut_q (Total surplus deficit in base currency)	
Datatype	INT64_T
Description	Total surplus or deficit in base currency after fx haircut. The number of decimals equals decimals in premium price of currency.
total_surplus_deficit_base_cur_q (Total surplus deficit in base currency)	
Datatype	INT64_T
Description	Total surplus or deficit in base currency. The number of decimals equals decimals in premium price of currency.
total_surplus_deficit_q (Total surplus deficit)	
Datatype	INT64_T
Description	Total surplus or deficit from collateral evaluation. The number of decimals equals decimals in premium price of currency.
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)	

© The NASDAQ OMX Group, Inc. • 2013

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
to_settlement_date_s (To Settlement Date)	
Datatype	char[8]
Description	Specifies to settlement date.
to_termination_agree_date_s (To Termination Agree Date)	
Datatype	char[8]
Description	The answer to the query should return records to this termination date
to_time_s (Time, To)	
Datatype	char[6]
Description	Defines the to time. Format: HHMMSS.
traded_bought_q (Traded Bought)	
Datatype	INT64_T
Description	Specifies the maximum allowed quantity of bought positions in the market.
traded_c (Traded)	
Datatype	UINT8_T
Description	Defines if the instrument is a tradable instrument or not.
Value Set	**name** / **value**: Yes / 1; No / 2
traded_in_click_c (Traded in GENIUM)	
Datatype	UINT8_T
Description	Specifies whether the series is traded in the system or not.
Value Set	**value** / **description**: 1 / Yes; 2 / No
traded_net_q (Traded Net)	
Datatype	INT64_T
Description	Specifies the maximum allowed quantity of traded bought-traded sold positions in the market.

© The NASDAQ OMX Group, Inc. • 2013

traded_quantity_q (Traded Quantity)	
Datatype	INT64_T
Description	Specifies from which quantity the delay time is valid for.
traded_sold_q (Traded Sold)	
Datatype	INT64_T
Description	Specifies the maximum allowed quantity of sold positions in the market.
tradenumber (TRADENUMBER)	
Datatype	INT32_T
Description	intermediate field.
trader_authorization_c (Trader, Authorization)	
Datatype	UINT8_T
Description	Defines if the user is allowed to act on firm orders.
Value Set	**name** / **value** Allow delete/alter firm orders / 1 Disallow delete/alter firm orders / 2
trades_allowed_c (Trades, Allowed)	
Datatype	UINT8_T
Description	Is it allowed to store trades on the account
Value Set	**name** / **value** Yes / 1 No / 2
trade_condition_n (Trade Condition)	
Datatype	UINT16_T
Description	The condition in which a trade was executed.
Value Set	**name** / **value** / **description** trade_cnd_no_cnd / 0 / No condition trade_cnd_late_trade / 1 / Late Trade trade_cnd_internal_trade / 2 / Internal Trade/Crossing trade_cnd_bulletin_board / 4 / Bulletin Board Trade trade_cnd_buy_write / 8 / Buy Write trade_off_market / 16 / Off Market
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_number_q (Trade number)	
Datatype	INT64_T
Description	Trade number
trade_operation_c (Trade Operation)	
Datatype	UINT8_T
Description	Defines the type trade operation.
Value Set	**name** / **value** None / 0 New Deal / 1 Rectify / 2 Cancel / 3 Terminate / 4 Retry / 5 Retry Auto / 6
trade_operation_number_q (TRADE_OPERATION_NUMBER_Q)	
Datatype	INT64_T
Description	Unique Trade Operation Number
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_reject_sec_u (Trade Reject, Seconds)	
Datatype	UINT32_T
Description	Defines the time in seconds during which it is possible to reject the trade.
trade_reporting_only_c (Only trade reports allowed)	
Datatype	UINT8_T
Description	Specifies whether the series only allows trade reporting.
Value Set	**value** / **description** 1 / Yes

© The NASDAQ OMX Group, Inc. • 2013

value	description
2	No

trade_report_category_c (Trade Report Category)	
Datatype	UINT8_T
Value Set	

name	value
Normal	1
Generate SWIFT confirmation	2
Confirm (generated from confirm trade report tx).	3

trade_report_nbr_q (Trade report number)	
Datatype	UINT64_T
Description	Unique number for trade report.

trade_report_number_q (TRADE REPORT NUMBER)	
Datatype	UINT64_T
Description	An increasing sequence number assigned to each trade. Trade number is unique within Instrument type

trade_report_reason_c (Trade report reason)	
Datatype	UINT8_T
Description	Enumeration describing the reason for state and sub states of a Trade Report, or action to a Trade Report.
Value Set	

name	value
None	0
Counterparty has cancelled	1
Pending Counterparty cancel	2
Counterparty has terminated	3
Pending Counterparty termination	4
Party Clearing Member	5
Counterparty Clearing Member	6
Party lacks collateral	7
Counterparty lacks collateral	8
Old account lacks collateral	9
New account lacks collateral	10
Both Accounts are lacking collateral	11
Manually Confirmed by Clearinghouse	12
Manually Rejected by Clearinghouse	13
Automatic End of Day Cleanup	14

© The NASDAQ OMX Group, Inc. • 2013

name	value
Rejected by Counterparty	15
Exposure exceeded	16
Exposure exceeded and lacking collateral	17
Record update	18
Confirmation Due on Termination Date	19
Configuration Error	20
Party Exposure Limit Exceeded	21
Counterparty Exceeded Exposure Limit	22

trade_report_state_c (Trade Report State)	
Datatype	UINT8_T
Description	Enumeration for the various states of a Trade Report, or action to a Trade Report.

Value Set

name	value
none	0
Unmatched	1
Matched	3
Cancelled	4
Rejected by Clearinghouse	5
Novated	6
Terminated	7
Deleted	8

trade_report_sub_state_c (Trade Report Substate)	
Datatype	UINT8_T
Description	Enumeration for the various sub states of a Trade Report, or action to a Trade Report.

Value Set

name	value
none	0
Pending cancel	1
Pending Termination	3
Netted to Zero	6
Pending Clearing Member Acceptance	13
Rejected by Clearing Member	14
Pending Clearinghouse Confirmation	15
Pending Clearinghouse Auto Confirm	16
Rejected by Clearinghouse	17
Cancelled by Counterpart	18

© The NASDAQ OMX Group, Inc. • 2013

name	value
Ongoing Clearinghouse Check	19

trade_report_type_i (Trade Report Type)	
Datatype	UINT32_T
Description	Enumeration for the various types of trade reports

Value Set

name	value
none	0
Standard	1
Tailormade	2
Fixed income	3
Discount security	4
FRA	5
IR Swap	6
Fx	7
Cash	8
Repo	9
Agreement	10
SSI	11
Equity	12
Xcur Swap	13

trade_report_version_n (Trade report version)	
Datatype	UINT16_T
Description	Version of a trade report.

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

value	description
1	Active. The trade is active.
2	Rectified. The trade has been rectified.
3	Deleted. The trade has been deleted.
4	Transferred. The trade has been transferred.

© The NASDAQ OMX Group, Inc. • 2013

<table>
<tr><td colspan="2">trade_type_c (Type, Trade)</td></tr>
<tr><td>Datatype</td><td>UINT8_T</td></tr>
<tr><td>Description</td><td>What type of trade is it?</td></tr>
<tr><td>Value Set</td><td>
<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>
<tr><td>15</td><td>Superseding</td></tr>
<tr><td>16</td><td>State_change</td></tr>
<tr><td>17</td><td>Give_up</td></tr>
<tr><td>18</td><td>Take_up</td></tr>
</table>
</td></tr>
<tr><td colspan="2">trade_venue_c (Trade venue)</td></tr>
</table>

© The NASDAQ OMX Group, Inc. • 2013

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
trading_suspend_resume_c (Trading, Suspend/Resume)	
Datatype	CHAR
Description	Defines if the participant is Suspended/Resumed.
Value Set	**value** / **description** 1 / Resume 2 / Suspend
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
transfer_cash_account_s (Transfer Account, Cash)	
Datatype	char[24]

© The NASDAQ OMX Group, Inc. • 2013

Description	The counterparty account within a cash record. A Cash Account (Cash Record) is unique within a Member. Allowed characters are (A-Z), (a-z), (0-9) and space and hyphen.
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

© The NASDAQ OMX Group, Inc. • 2013

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	**value** / **description** Y / Yes N / No
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	**name** / **value** Primary Unit / 1 Secondary Unit / 2 Tertiary Unit / 3
trend_indicator_c (Trend Indicator)	
Datatype	CHAR
Description	Trend indicator for the latest price compared to the previous one.
Value Set	**name** / **value** / **description** Up / + / Price is higher price than previously. Down / - / Price is lower price than previously. Same / = / Price is unchanged. None / / No trend available, it might for example be the first price of the day. The value is blank (space).
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.

© The NASDAQ OMX Group, Inc. • 2013

turnaround_yesterday_u (Turnover, Yesterday)	
Datatype	INT64_T
Description	The total traded amount yesterday.
turnover_list_name_s (Turnover List Name)	
Datatype	char[32]
Description	Defines the name of the turnover list.
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.
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).
tx_status_i (TX_STATUS_I)	
Datatype	INT32_T
Description	Transaction status
type_of_date_c (Type of Date)	
Datatype	UINT8_T
Description	Identifies the type of date, e.g. termination date, flow date.
Value Set	**name** / **value** none / 0 Termination date / 1 Swap Flow Start date / 2 Swap Flow End date / 3 Rate Set date / 4 Payment date / 5 Principal Exchange date / 6 Unwind Settlement date / 7 First Rollover date / 8
tz_exchange_s (Time Zone, Exchange)	
Datatype	char[40]
Description	The time zone environment variable for the exchange. (POSIX standard)

© The NASDAQ OMX Group, Inc. • 2013

	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_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

© The NASDAQ OMX Group, Inc. • 2013

value	description
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

undisclosed_min_ord_val_i (Minimum Order Value, Undisclosed Quantity)	
Datatype	INT32_T
Description	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.

und_price_modifier_c (Modifier, Underlying 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

und_price_mod_factor_i (Modifier Factor, Underlying 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

© The NASDAQ OMX Group, Inc. • 2013

unwind_consideration_q (UNWIND_CONSIDERATION_Q)	
Datatype	INT64_T
unwind_settlement_date_s (Unwind Settlement Date)	
Datatype	char[8]
Description	The date when the REPO terminates
update_status_note_c (Status Note, Update)	
Datatype	UINT8_T
Description	Create notification code in CDB, is exchange specific.
Value Set	**value** / **description** 1 / Yes 2 / No
upper_ccc_id_s (Upper Curve Correlation Cube)	
Datatype	char[12]
Description	Name of Upper Curve Correlation Cube
upper_limit_i (Premium/Price, High Limit)	
Datatype	INT32_T
Description	The upper limit in the price interval.
up_int_i (Valuation Interval, Up)	
Datatype	INT32_T
Description	Defines the valuation interval up in margin calculations. Expressed in percent of underlying price. Represented with 4 implicit decimals.
urgent_c (Urgent)	
Datatype	UINT8_T
Description	Indicates whether the message should be treated as urgent or not. Urgent = 1, Not Urgent = 2
url_link_s (Link, URL)	
Datatype	CHAR[255]
Description	The Link, URL field hold the full URL for a link elsewhere on the Web, typically a document.
user_code_s (User Code)	
Datatype	char[12]
Description	Defines a unique user in the system.
user_id_s (User)	
Datatype	char[5]
Description	Defines the user signature.
use_agreement_c (Use agreement)	
Datatype	UINT8_T

© The NASDAQ OMX Group, Inc. • 2013

<table>
<tr><td>Description</td><td>If agreement is used</td></tr>
<tr><td colspan="2">use_ssi_c (Use SSI)</td></tr>
<tr><td>Datatype</td><td>UINT8_T</td></tr>
<tr><td>Description</td><td>Specifies whether SSI (Standard Settlement Instruction) should be used 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">usr_id_n (User, Number)</td></tr>
<tr><td>Datatype</td><td>UINT16_T</td></tr>
<tr><td>Description</td><td>A unique number that identified the user, used when subscribing for directed broadcast information.</td></tr>
<tr><td colspan="2">ust_id_s (User Type, Identity)</td></tr>
<tr><td>Datatype</td><td>char[5]</td></tr>
<tr><td>Description</td><td>The name of the user type.</td></tr>
<tr><td colspan="2">utc_date_s (UTC, Date)</td></tr>
<tr><td>Datatype</td><td>char[8]</td></tr>
<tr><td>Description</td><td>UTC date, format: YYYYMMDD.</td></tr>
<tr><td colspan="2">utc_offset_i (UTC, Offset)</td></tr>
<tr><td>Datatype</td><td>INT32_T</td></tr>
<tr><td>Description</td><td>Current offset between UTC and the local time specified in the TZ-variable.</td></tr>
<tr><td colspan="2">utc_time_s (UTC, Time)</td></tr>
<tr><td>Datatype</td><td>char[6]</td></tr>
<tr><td>Description</td><td>UTC time, format: HHMMSS.</td></tr>
<tr><td colspan="2">vag_id_s (Valuation Group Identity)</td></tr>
<tr><td>Datatype</td><td>char[12]</td></tr>
<tr><td>Description</td><td>Collateral valuation group identity.</td></tr>
<tr><td colspan="2">vag_limit_i (Valuation Group Limit (%))</td></tr>
<tr><td>Datatype</td><td>INT32_T</td></tr>
<tr><td>Description</td><td>The upper limit of how much of the intial margins that can be covered by collaterals belonging to this Valuation Group.Expressed in percent. No implicit decimals.</td></tr>
<tr><td colspan="2">valid_from_date_s (Valid From Date)</td></tr>
<tr><td>Datatype</td><td>char[8]</td></tr>
<tr><td>Description</td><td>The date from when the item is active from in format YYYYMMDD.</td></tr>
<tr><td colspan="2">valuation_date_s (Valuation Date)</td></tr>
<tr><td>Datatype</td><td>char[8]</td></tr>
<tr><td>Description</td><td>The date of a collateral valuation run. Format is YYYYMMDD.</td></tr>
<tr><td colspan="2">value_high_i (Value, high)</td></tr>
</table>

© The NASDAQ OMX Group, Inc. • 2013

Datatype	INT32_T
Description	Margin value calculated with high volatiltiy, 2 implicit decimals.
value_low_i (Value, low)	
Datatype	INT32_T
Description	Margin value calculated with low volatiltiy, 2 implicit decimals.
value_middle_i (Value, middle)	
Datatype	INT32_T
Description	Margin value calculated with middle volatiltiy, 2 implicit decimals.
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.
variation_margin_req_q (Variation margin requirement.)	
Datatype	INT64_T
Description	Variation margin, i.e. daily settlement for futures.
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.
version_i (VERSION_I)	
Datatype	INT32_T
version_n (Version; Collateral position version)	
Datatype	UINT16_T
Description	Version of collateral position or bank/payment instruction.
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)	

© The NASDAQ OMX Group, Inc. • 2013

<table>
<tr><td>Datatype</td><td>UINT16_T</td></tr>
<tr><td>Description</td><td>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.</td></tr>
<tr><td colspan="2">volatility_corr_rm_c (Volatility correlation)</td></tr>
<tr><td>Datatype</td><td>UINT8_T</td></tr>
<tr><td>Description</td><td>If Yes then the volatility is correlated in the margin calculation</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">volatility_i (volatility)</td></tr>
<tr><td>Datatype</td><td>INT32_T</td></tr>
<tr><td>Description</td><td>Volatility</td></tr>
<tr><td colspan="2">volume_today_i (Volume, Today)</td></tr>
<tr><td>Datatype</td><td>INT64_T</td></tr>
<tr><td>Description</td><td>Today's volume.</td></tr>
<tr><td colspan="2">volume_u (Volume)</td></tr>
<tr><td>Datatype</td><td>INT64_T</td></tr>
<tr><td>Description</td><td>Order or trade volume.</td></tr>
<tr><td colspan="2">volume_yesterday_i (Volume, Yesterday)</td></tr>
<tr><td>Datatype</td><td>INT64_T</td></tr>
<tr><td>Description</td><td>Yesterday's volume.</td></tr>
<tr><td colspan="2">vol_ivl_held_high_i (Volatility Interval Held, High)</td></tr>
<tr><td>Datatype</td><td>INT32_T</td></tr>
<tr><td>Description</td><td>The high implied volatility used in margin calculations for held options. Expressed in percent. 4 implicit decimals</td></tr>
<tr><td colspan="2">vol_ivl_held_low_i (Volatility Interval Held, Low)</td></tr>
<tr><td>Datatype</td><td>INT32_T</td></tr>
<tr><td>Description</td><td>The low implied volatility used in margin calculations for held options. Expressed in percent. 4 implicit decimals</td></tr>
<tr><td colspan="2">vol_ivl_held_mid_i (Volatility Interval Held, Mid)</td></tr>
<tr><td>Datatype</td><td>INT32_T</td></tr>
<tr><td>Description</td><td>The mid implied volatility used in margin calculations for held options. Expressed in percent. 4 implicit decimals</td></tr>
<tr><td colspan="2">vol_ivl_long_high_i (Volatility Interval Long, High)</td></tr>
<tr><td>Datatype</td><td>INT32_T</td></tr>
</table>

© The NASDAQ OMX Group, Inc. • 2013

Description	The high implied volatility used in margin calculations for long options. Expressed in percent. 4 implicit decimals.
vol_ivl_long_low_i (Volatility Interval Long, Low)	
Datatype	INT32_T
Description	The low implied volatility used in margin calculations for long options. Expressed in percent. 4 implicit decimals.
vol_ivl_long_mid_i (Volatility Interval Long, Mid)	
Datatype	INT32_T
Description	The mid implied volatility used in margin calculations for long options. Expressed in percent. 4 implicit decimals.
vol_ivl_short_high_i (Volatility Interval Short, High)	
Datatype	INT32_T
Description	The high implied volatility used in margin calculations for short options. Expressed in percent. 4 implicit decimals.
vol_ivl_short_low_i (Volatility Interval Short, Low)	
Datatype	INT32_T
Description	The low implied volatility used in margin calculations for short options. Expressed in percent. 4 implicit decimals.
vol_ivl_short_mid_i (Volatility Interval Short, Mid)	
Datatype	INT32_T
Description	The mid implied volatility used in margin calculations for short 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_src_c (Volatility Source)	
Datatype	UINT8_T
Description	Defines how volatility is fetched for this series.

© The NASDAQ OMX Group, Inc. • 2013

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
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

warning_breach_lvl_n (Warning Breach Level)	
Datatype	INT16_T
Description	Specifies the percentage of the limits when warning emails can be sent.

warning_msg_s (Warning Message)	
Datatype	char[80]
Description	This is a warning message that will be shown at a trading state change.

warrant_c (Warrant)	
Datatype	UINT8_T
Description	If the instrument is a warrant:

Value Set

value	description
1	Yes
2	No

when_issued_c (When Issued)	
Datatype	UINT8_T

© The NASDAQ OMX Group, Inc. • 2013

Description	Not applicable.
Value Set	**value** / **description**: 2 / No
win_id_s (Window Class)	
Datatype	char[15]
Description	Window class used in window method in margin calculations.
written_high_i (Written, High)	
Datatype	UINT32_T
Description	Margin vector value for a written series at a high volatility, and at the corresponding spot price, 2 implicit decimals.
written_low_i (Written, Low)	
Datatype	UINT32_T
Description	Margin vector value for a written series at a low volatility, and at the corresponding spot price, 2 implicit decimals.
written_middle_i (Written, Middle)	
Datatype	UINT32_T
Description	Margin vector value for a written series at a medium volatility, and at the corresponding spot price, 2 implicit decimals.
writ_for_adj_i (Future Adjustment Written)	
Datatype	INT32_T
Description	Adjustment factor for margin calculation of written futures and forwards. Expressed in percent with 4 implicit decimals.
writ_marg_q (Marginables, Written)	
Datatype	INT64_T
Description	The number of written marginables in a position.
writ_vol_down_i (Volatility Written, Down)	
Datatype	INT32_T
Description	Volatility interval down for written options in margin calculations. Expressed in percent, 4 implicit decimals.
writ_vol_up_i (Volatility Written, Up)	
Datatype	INT32_T
Description	Volatility interval up for written options in margin calculations. Expressed in percent, 4 implicit decimals.
yield_conv_n (Yield Convention)	
Datatype	UINT16_T
Description	Yield Convention Number of month
yield_i (YIELD_I)	

© The NASDAQ OMX Group, Inc. • 2013

Datatype	INT32_T
yymmdd_s (Trading Date)	
Datatype	char[6]
Description	Date in ASCII. Format: YYMMDD.
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

© The NASDAQ OMX Group, Inc. • 2013

Index Register

A

© The NASDAQ OMX Group, Inc. • 2013

B

© The NASDAQ OMX Group, Inc. • 2013

C

© The NASDAQ OMX Group, Inc. • 2013

© The NASDAQ OMX Group, Inc. • 2013

D

© The NASDAQ OMX Group, Inc. • 2013

© The NASDAQ OMX Group, Inc. • 2013

© The NASDAQ OMX Group, Inc. • 2013

E

© The NASDAQ OMX Group, Inc. • 2013

F

G

H

I

© The NASDAQ OMX Group, Inc. • 2013

J

K

L

© The NASDAQ OMX Group, Inc. • 2013

M

© The NASDAQ OMX Group, Inc. • 2013

© The NASDAQ OMX Group, Inc. • 2013

N

O

© The NASDAQ OMX Group, Inc. • 2013

P

© The NASDAQ OMX Group, Inc. • 2013

Q

R

© The NASDAQ OMX Group, Inc. • 2013

S

© The NASDAQ OMX Group, Inc. • 2013

© The NASDAQ OMX Group, Inc. • 2013

T

© The NASDAQ OMX Group, Inc. • 2013

U

© The NASDAQ OMX Group, Inc. • 2013

V

W

© The NASDAQ OMX Group, Inc. • 2013

Y

© The NASDAQ OMX Group, Inc. • 2013