Genium INET^{SM}

OMnet Message Reference

ASX Futures

Version: 2.0.0601

Document ID:	OMnet_MessRef_32
Documentation Release:	GENIUM_Product_a1018
Release Date:	2013-12-20
Publication Date:	2013-12-20
Confidentiality:	Non-confidential
API Kit:	48689

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.

Table of Contents

List of Tables

List of Figures

1 Summary of Changes

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

Changes between (41494) and (48689) for SFE (a63/a80).

	Changed message	Changes	Comments
1	BD18	Changes in struct **directed_delivery:** Changes in struct **cl_delivery_api:** Changes in type **class_no_i:** Changes in value set: Added value: 13 Added value: 11 Added value: 12 Added value: 21	
2	BU120	Textual changes in message description: section: Usage and Conditions: example #2: titled-block #3: list #1: listitem #2: new titled-block #4: list #1: listitem #2: new	
3	CD31	Textual changes in message description: section: Return Codes: new	
4	CQ31	Changes in answer **CA31:** Changes in struct **answer_delivery:** Changes in array **item:** Changes in type **class_no_i:** Changes in value set: Added value: 13 Added value: 11 Added value: 12 Added value: 21	
5	CQ52	Changes in answer **CA52:** Changes in struct **answer_missing_delivery:** Changes in array **item:** Changes in struct **cl_delivery_api:** Changes in type **class_no_i:** Changes in value set: Added value: 13 Added value: 11	

	Changed message	Changes	Comments
		Added value: 12 Added value: 21	
6	CQ53	Changes in answer **CA53:** Changes in struct **answer_api_delivery:** Changes in array **item:** Changes in struct **cl_delivery_api:** Changes in type **class_no_i:** Changes in value set: Added value: 13 Added value: 11 Added value: 12 Added value: 21	
7	LQ3	Changes in answer **LA3:** Changes in struct **answer_list_ver:** Changes in type **text_buffer_s:** Changed description	

2 Document Information

2.1 References

Here is a list of OMnet related documents:

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

2.2 Reader's Roadmap

This message reference contains the following chapters:

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

2.2.1 The OMnet Messages Chapter

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

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

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

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

<table>
<tr><th>Section</th><th>Description</th></tr>
<tr><td>Fingerprint</td><td>Each message has a Fingerprint section containing the following information:
<table>
<tr><th>Heading</th><th>Description</th></tr>
<tr><td>Transaction type</td><td>Transaction type is the identification of the transaction; broadcast, query or answer.
For more information on how the Transaction type is designed, refer to <i>OMnet Message Reference Manual, Introduction</i>.</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 <i>OMnet Application Programmer's Interface Manual</i>.</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 <i>OMnet Application Programmer's Interface Manual</i>.</td></tr>
<tr><td>Segmented</td><td>Specifies if an answer or broadcast is segmented or not (true/false).
For details, refer to <i>OMnet Message Reference Manual, Introduction</i>.</td></tr>
<tr><td>Partitioned</td><td>Specifies if a transaction or query is partitioned or not (true/false).
For more information, refer to <i>OMnet Message Reference Manual, Introduction</i>.</td></tr>
<tr><td>Facility</td><td>Transactions are sent on paths through the system called facilities. The system is only able to rout a transaction correctly if it is sent on the correct facility.</td></tr>
</table>
</td></tr>
</table>

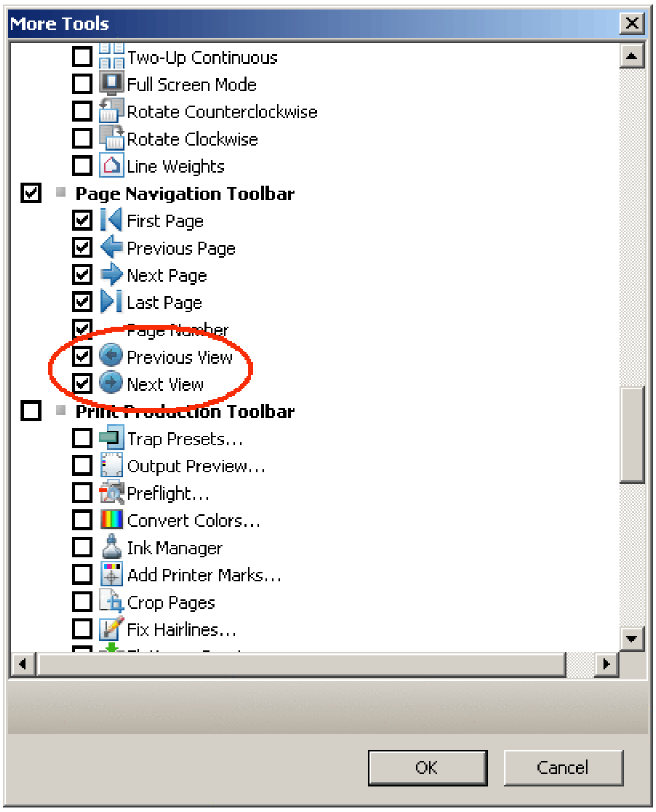
<table>
<tr><th>Section</th><th>Description</th></tr>
<tr><td></td><td><table><tr><th>Heading</th><th>Description</th></tr><tr><td></td><td>Refer to <i>OMnet Application Programmer’s Interface Manual</i>.</td></tr><tr><td>Virtual Underlying</td><td>Virtual Underlying is a grouping concept that makes 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 <i>OMnet Message Reference Manual, Introduction</i>.</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.



☐ Two-Up Continuous
☐ Full Screen Mode
☐ Rotate Counterclockwise
☐ Rotate Clockwise
☐ Line Weights
☑ Page Navigation Toolbar
☑ First Page
☑ Previous Page
☑ Next Page
☑ Last Page
☑ Page Number
☑ Previous View
☑ Next View
☐ Print Production Toolbar
☐ Trap Presets...
☐ Output Preview...
☐ Preflight...
☐ Convert Colors...
☐ Ink Manager
☐ Add Printer Marks...
☐ Crop Pages
☐ Fix Hairlines...

Figure 1: More Tools Dialog

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

Note:

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

3 OMnet Messages

3.1 Reference Data

3.1.1 BU2 [Series Update BROADCAST]

3.1.1.1 Fingerprint

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

3.1.1.2 Related Messages

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

3.1.1.3 Purpose

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

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

3.1.1.4 Structure

The BU2 BROADCAST has the following structure:

```
struct series_update_bu2 {
   struct broadcast_type
   UINT16_T chg_type_n  // Change Type
   char[2] filler_2_s  // Filler
   struct da2 {
      struct series  // Named struct no: 50000
      struct upper_level_series
      INT32_T contract_size_i  // Contract Size
      INT32_T price_quot_factor_i  // Price, Quotation Factor
      UINT32_T series_sequence_number_u  // Series, Sequence Number
      UINT16_T state_number_n  // Trading State Number
      UINT16_T step_size_multiple_n  // Tick Size, Multiple
      char[32] ins_id_s  // Series, Identity
      char[12] isin_code_s  // ISIN Code
      UINT8_T suspended_c  // Suspended
      char[8] date_last_trading_s  // Date, Last Trading
      char[6] time_last_trading_s  // Time, Last Trading
      char[8] settlement_date_s  // Date, Settlement
      char[8] start_date_s  // Date, Start
      char[8] end_date_s  // Date, End
      char[8] date_delivery_start_s  // Date, Delivery Start
      char[8] date_delivery_stop_s  // Date, Delivery Stop
      UINT8_T series_status_c  // Series, Status
      char[32] long_ins_id_s  // Series Name, Long
      char[8] date_first_trading_s  // Date, First Trading
      char[6] time_first_trading_s  // Time, First Trading
      UINT8_T traded_in_click_c  // Traded in GENIUM
      char[8] abbr_name_s  // Abbreviated Name
      char[6] stock_code_s  // Stock Code
      UINT8_T ext_info_source_c  // External Information Source
      char[8] effective_exp_date_s  // Effective Expiration Date
      char[2] filler_2_s  // Filler
   }
}
```

3.1.1.5 Usage and Conditions

Change Type

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

Trading State Number

contains the immediate ISS.

3.1.2 BU4 [Underlying Update BROADCAST]

3.1.2.1 Fingerprint

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

3.1.2.2 Related Messages

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

3.1.2.3 Purpose

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

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

3.1.2.4 Structure

The BU4 BROADCAST has the following structure:

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

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

3.1.2.5 Usage and Conditions

Change Type

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

Trading State Number

will contain the immediate ISS.

3.1.3 BU9 [Series Backoffice Update BROADCAST]

3.1.3.1 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

3.1.3.2 Related Messages

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

3.1.3.3 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).

3.1.3.4 Structure

The BU9 BROADCAST has the following structure:

```
 struct series_bo_update_bu9 {
    struct broadcast_type
```

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

3.1.3.5 Usage and Conditions

Trading State Number

will contain the immediate ISS.

3.1.4 BU10 [Instrument Class Update BROADCAST]

3.1.4.1 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

3.1.4.2 Related Messages

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

3.1.4.3 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).

3.1.4.4 Structure

The BU10 BROADCAST has the following structure:

```
struct inst_class_update_bu10_bu20 {
   struct broadcast_type
   UINT16_T chg_type_n  // Change Type
   char[2] filler_2_s  // Filler
   struct da10_da20 {
      struct series  // Named struct no: 50000
      struct upper_level_series
      INT32_T price_quot_factor_i  // Price, Quotation Factor
      INT32_T contract_size_i  // Contract Size
      INT32_T exerc_limit_i  // Exercise, Limit
      INT32_T redemption_value_i  // Redemption Value
      INT32_T min_qty_increment_i  // Minimum Quantity Increment
      UINT16_T derivate_level_n  // Derivate Level
      UINT16_T dec_in_strike_price_n  // Decimals, Strike Price
      UINT16_T dec_in_contr_size_n  // Decimals, Contract Size
      UINT16_T rnt_id_n  // Ranking Type
      UINT16_T dec_in_premium_n  // Decimals, Premium
      UINT16_T items_n  // Items
      Array ITEM [max no: 12] {
         struct tick_size
      }
      UINT16_T dec_in_deliv_n  // Decimals, Delivery
      UINT16_T items_block_n  // Item, Block
      Array BLOCK_SIZE [max no: 4] {
         INT64_T maximum_size_u  // Block Size, Maximum Volume
         UINT32_T minimum_size_n  // Block Size, Minimum Volume
         UINT32_T block_n  // Block Size
         UINT8_T lot_type_c  // Lot, Type
         char[3] filler_3_s  // Filler
      }
      UINT16_T cleared_dec_in_qty_n  // Decimals, Quantity
      UINT16_T virt_commodity_n  // Virtual Underlying
      UINT16_T dec_in_fixing_n  // Decimals, Fixing
      char[3] base_cur_s  // Currency, Trading
      UINT8_T traded_c  // Traded
      UINT8_T exerc_limit_unit_c  // Exercise, Limit Unit
      char[14] inc_id_s  // Instrument Class, Identity
      char[10] trc_id_s  // Trade Report Class
      char[32] name_s  // Name
      CHAR is_fractions_c  // Fraction, Premium
      UINT8_T price_format_c  // Premium/Price Format
      UINT8_T strike_price_format_c  // Strike Price, Format
      UINT8_T cabinet_format_c  // Cabinet Format
      UINT8_T price_unit_premium_c  // Price Unit, Premium
      UINT8_T price_unit_strike_c  // Price Unit, Strike
      char[32] settl_cur_id_s  // Currency, Settlement
      char[3] credit_class_s  // Credit Class
      char[12] csd_id_s  // CSD, Identity
      UINT8_T trd_cur_unit_c  // Traded Currency Unit
      UINT8_T collateral_type_c  // Collateral types
      UINT8_T fixing_req_c  // FIXING_REQ_C
      CHAR[2] mbs_id_s  // Minimum Bid Schedule
      char[12] valuation_group_id_s  // VAG, Identity ; Of type: VAG_ID_S
      char[3] filler_3_s  // Filler
   }
}
```

3.1.4.5 Usage and Conditions

Change Type

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

3.1.5 BU12 [Account Type Update BROADCAST]

3.1.5.1 Fingerprint

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

3.1.5.2 Related Messages

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

3.1.5.3 Purpose

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

3.1.5.4 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
   }
}
```

3.1.5.5 Usage and Conditions

Change Type

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

3.1.6 BU13 [Account Fee Type Update BROADCAST]

3.1.6.1 Fingerprint

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

3.1.6.2 Related Messages

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

3.1.6.3 Purpose

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

3.1.6.4 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
   }
}
```

3.1.6.5 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 BU18 [Non-Trading Days Update BROADCAST]

3.1.7.1 Fingerprint

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

3.1.7.2 Related Messages

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

3.1.7.3 Purpose

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

3.1.7.4 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
      }
   }
```

3.1.7.5 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 BU19 [Underlying Backoffice Update BROADCAST]

3.1.8.1 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

3.1.8.2 Related Messages

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

3.1.8.3 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).

3.1.8.4 Structure

The BU19 BROADCAST has the following structure:

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

3.1.8.5 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.

3.1.9 BU20 [Instrument Class Backoffice Update BROADCAST]

3.1.9.1 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

3.1.9.2 Related Messages

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

3.1.9.3 Purpose

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

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

3.1.9.4 Structure

The BU20 BROADCAST has the following structure:

```
struct inst_class_update_bu10_bu20 {
   struct broadcast_type
   UINT16_T chg_type_n  // Change Type
   char[2] filler_2_s  // Filler
   struct da10_da20 {
      struct series  // Named struct no: 50000
      struct upper_level_series
      INT32_T price_quot_factor_i  // Price, Quotation Factor
      INT32_T contract_size_i  // Contract Size
      INT32_T exerc_limit_i  // Exercise, Limit
      INT32_T redemption_value_i  // Redemption Value
      INT32_T min_qty_increment_i  // Minimum Quantity Increment
      UINT16_T derivate_level_n  // Derivate Level
      UINT16_T dec_in_strike_price_n  // Decimals, Strike Price
      UINT16_T dec_in_contr_size_n  // Decimals, Contract Size
      UINT16_T rnt_id_n  // Ranking Type
      UINT16_T dec_in_premium_n  // Decimals, Premium
      UINT16_T items_n  // Items
      Array ITEM [max no: 12] {
         struct tick_size
      }
      UINT16_T dec_in_deliv_n  // Decimals, Delivery
      UINT16_T items_block_n  // Item, Block
      Array BLOCK_SIZE [max no: 4] {
         INT64_T maximum_size_u  // Block Size, Maximum Volume
         UINT32_T minimum_size_n  // Block Size, Minimum Volume
         UINT32_T block_n  // Block Size
         UINT8_T lot_type_c  // Lot, Type
         char[3] filler_3_s  // Filler
      }
      UINT16_T cleared_dec_in_qty_n  // Decimals, Quantity
      UINT16_T virt_commodity_n  // Virtual Underlying
      UINT16_T dec_in_fixing_n  // Decimals, Fixing
      char[3] base_cur_s  // Currency, Trading
      UINT8_T traded_c  // Traded
      UINT8_T exerc_limit_unit_c  // Exercise, Limit Unit
      char[14] inc_id_s  // Instrument Class, Identity
      char[10] trc_id_s  // Trade Report Class
      char[32] name_s  // Name
      CHAR is_fractions_c  // Fraction, Premium
      UINT8_T price_format_c  // Premium/Price Format
      UINT8_T strike_price_format_c  // Strike Price, Format
      UINT8_T cabinet_format_c  // Cabinet Format
      UINT8_T price_unit_premium_c  // Price Unit, Premium
      UINT8_T price_unit_strike_c  // Price Unit, Strike
      char[32] settl_cur_id_s  // Currency, Settlement
      char[3] credit_class_s  // Credit Class
      char[12] csd_id_s  // CSD, Identity
      UINT8_T trd_cur_unit_c  // Traded Currency Unit
      UINT8_T collateral_type_c  // Collateral types
      UINT8_T fixing_req_c  // FIXING_REQ_C
      CHAR[2] mbs_id_s  // Minimum Bid Schedule
      char[12] valuation_group_id_s  // VAG, Identity ; Of type: VAG_ID_S
      char[3] filler_3_s  // Filler
   }
}
```

3.1.9.5 Usage and Conditions

Change Type

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

3.1.10 BU44 [Legal Account Instrument Update BROADCAST]

3.1.10.1 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

3.1.10.2 Related Messages

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

3.1.10.3 Purpose

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

3.1.10.4 Structure

The BU44 BROADCAST has the following structure:

```
struct legal_account_instrument_update_bu44 {
   struct broadcast_type
   UINT16_T chg_type_n  // Change Type
   char[2] filler_2_s  // Filler
   struct da44 {
      struct series  // Named struct no: 50000
      char[12] acc_type_s  // Account Type
   }
}
```

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 BU120 [Delta Underlying Update VIB]

3.1.11.1 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

3.1.11.2 Related Messages

DQ120

3.1.11.3 Purpose

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

3.1.11.4 Structure

The BU120 VIB has the following structure:

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

3.1.11.5 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.

3.1.12 BU121 [Delta Underlying Update for Back Office VIB]

3.1.12.1 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

3.1.12.2 Related Messages

DQ121

3.1.12.3 Purpose

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

3.1.12.4 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
                 }
              }
           }
```

3.1.12.5 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**.

3.1.13 BU122 [Delta Instrument Class Update VIB]

3.1.13.1 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

3.1.13.2 Related Messages

DQ122

3.1.13.3 Purpose

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

3.1.13.4 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
            }
         }
      }
```

3.1.13.5 Usage and Conditions

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

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

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

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

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

3.1.14.1 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

3.1.14.2 Related Messages

DQ123

3.1.14.3 Purpose

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

3.1.14.4 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_leg_calc_rule  // Named struct no: 37115
         struct ns_price_tick_corr  // Named struct no: 37113
      }
   }
}
```

3.1.14.5 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**.

3.1.15 BU124 [Delta Instrument Series Update VIB]

3.1.15.1 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

3.1.15.2 Related Messages

DQ124

3.1.15.3 Purpose

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

3.1.15.4 Structure

The BU124 VIB has the following structure:

```
struct broadcast_segment_hdr
struct item_hdr
struct sub_item_hdr
struct ns_delta_header  // Named struct no: 37001
Sequence {
   struct item_hdr
   Sequence {
      struct sub_item_hdr
      Choice {
         struct ns_remove  // Named struct no: 37002
         struct ns_inst_series_basic  // Named struct no: 37301
         struct ns_inst_series_basic_single  // Named struct no: 37302
         struct ns_inst_series_power  // Named struct no: 37303
         struct ns_inst_series_repo  // Named struct no: 37304
         struct ns_inst_series_leg_flow  // Named struct no: 37309
      }
   }
}
```

3.1.15.5 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**.

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

3.1.16.1 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

3.1.16.2 Related Messages

DQ125

3.1.16.3 Purpose

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

3.1.16.4 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
      }
   }
}
```

3.1.16.5 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**.

3.1.17 DQ2 [Series QUERY]

3.1.17.1 Fingerprint

QUERY properties	
transaction type	DQ2
calling sequence	omniapi_query_ex
struct name	query_series

QUERY properties	
facility	EP0
partitioned	false
segmented	true
answers	DA2

ANSWER properties	
transaction type	DA2
struct name	answer_series
segmented	true

3.1.17.2 Related Messages

BU2

3.1.17.3 Purpose

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

3.1.17.4 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
}
```

3.1.17.5 Usage and Conditions

Series

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

3.1.17.6 Answer Structure

The DA2 ANSWER has the following structure:

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

3.1.17.7 Answer, comments

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

Series

is returned regardless of the setting of the field traded_in_click_c.

Valid standard combination series will be included in the answer.

Upper Level Series

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

Contract Size

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

Price Quotation Factor

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

Trading State Number

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

3.1.18 DQ3 [Instrument Type QUERY]

3.1.18.1 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

3.1.18.2 Purpose

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

3.1.18.3 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
}
```

3.1.18.4 Usage and conditions

Series

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

3.1.18.5 Answer Structure

The DA3 ANSWER has the following structure:

```
struct answer_instrument {
   struct transaction type
   UINT16 T segment number n  // Segment Number
   UINT16 T items n  // Items
   Array ITEM [max no: 100] {
      struct series  // Named struct no: 50000
      UINT32_T min_show_vol_u  // Order, Min Show Volume
      UINT16 T hidden vol meth n  // Method, Hidden Volume
      UINT16 T pub inf id n  // Public Order Info
      char[8] int id s  // Instrument, Identity
      char[32] name s  // Name
      UINT8 T maintain positions c  // Maintain Positions
      UINT8 T traded c  // Traded
      UINT8 T post trade proc c  // Post Trade processed
      UINT8_T pos_handling_c  // Position handling
      UINT8 T directed trade information c  // Directed Trade Information
      UINT8 T public deal information c  // Public Deal Information
      char[2] filler 2 s  // Filler
   }
}
```

3.1.18.6 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.

3.1.19 DQ4 [Underlying QUERY]

3.1.19.1 Fingerprint

QUERY properties	
transaction type	DQ4
calling sequence	omniapi_query_ex
struct name	query_underlying
facility	EP0
partitioned	false

QUERY properties	
segmented	true
answers	DA4

ANSWER properties	
transaction type	DA4
struct name	answer_underlying
segmented	true

3.1.19.2 Related Messages

BU4

3.1.19.3 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).

3.1.19.4 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
}
```

3.1.19.5 Usage and conditions

Series

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

3.1.19.6 Answer Structure

The DA4 ANSWER has the following structure:

```
struct answer_underlying {
   struct transaction_type
   UINT16_T segment_number_n  // Segment Number
   UINT16_T items_n  // Items
   Array ITEM [max no: 50] {
      INT32_T subscription_price_i  // Subscription, Price
      INT32_T interest_rate_i  // Interest Rate
      UINT16_T commodity_n  // Commodity Code
      char[6] com_id_s  // Underlying Identity
      char[12] isin_code_s  // ISIN Code
      UINT16_T dec_in_price_n  // Decimals, Price
      char[8] date_release_s  // Date, Issue
      char[8] date_termination_s  // Date, Maturity
      char[8] date_dated_s  // Date, Dated
      char[32] name_s  // Name
      char[3] base_cur_s  // Currency, Trading
      UINT8_T deliverable_c  // Deliverable
      UINT16_T coupon_frequency_n  // Coupon Frequency
      INT64_T nominal_value_q  // Nominal Value
      UINT16_T day_count_n  // Day Count
      UINT16_T days_in_interest_year_n  // Days In Interest Year
      UINT32_T coupon_interest_i  // Coupon Interest
      UINT16_T coupon_settlement_days_n  // Coupon Settlement Days
      UINT8_T underlying_type_c  // Type, Underlying
      UINT8_T price_unit_c  // Price Unit, Underlying
      UINT16_T dec_in_nominal_n  // Decimals, Nominal
      UINT16_T state_number_n  // Trading State Number
      UINT16_T linked_commodity_n  // Linked Commodity Code
      UINT8_T fixed_income_type_c  // Fixed Income Type
      UINT8_T underlying_status_c  // Underlying Status
      char[6] underlying_issuer_s  // Underlying Issuer
      char[6] time_delivery_start_s  // Time, Delivery Start
      char[6] time_delivery_stop_s  // Time, Delivery Stop
      char[4] sector_code_s  // Sector Code
      UINT16_T items_n  // Items
      Array COUPON [max no: 80] {
         char[8] date_coupdiv_s  // Coupon/Dividend Date
         UINT32_T dividend_i  // Dividend
      }
      UINT8_T virtual_c  // Virtual
      char[4] member_circ_numb_s  // Member, Circular Number
      CHAR inv_scheme_c  // Investment Scheme
      char[8] date_closing_s  // Date, Closing
      char[8] date_last_s  // Date, Last
      char[2] country_id_s  // Name, Country
      UINT8_T cur_unit_c  // Currency Unit
      char[3] filler_3_s  // Filler
   }
}
```

3.1.19.7 Answer, comments

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

Trading State Number

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

Decimals, Price

are used to interpret the Price Information for the Underlying.

3.1.20 DQ6 [Broker Signatures QUERY]

3.1.20.1 Fingerprint

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

ANSWER properties	
transaction type	DA6
struct name	answer_broker
segmented	true

3.1.20.2 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.

3.1.20.3 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
}
```

3.1.20.4 Usage and Conditions

Series

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

3.1.20.5 Answer Structure

The DA6 ANSWER has the following structure:

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

3.1.20.6 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.

3.1.21 DQ7 [Market QUERY]

3.1.21.1 Fingerprint

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

ANSWER properties	
transaction type	DA7
struct name	answer_market
segmented	true

3.1.21.2 Purpose

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

3.1.21.3 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
}
```

3.1.21.4 Usage and Conditions

Series

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

3.1.21.5 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
   }
}
```

3.1.21.6 Answer, comments

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

3.1.22 DQ8 [Instrument Group QUERY]

3.1.22.1 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

3.1.22.2 Purpose

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

3.1.22.3 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
}
```

3.1.22.4 Usage and Conditions

Series

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

3.1.22.5 Answer Structure

The DA8 ANSWER has the following structure:

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

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

3.1.22.6 Answer, comments

The answer received contains a list of instrument groups.

3.1.23 DQ9 [Series Backoffice QUERY]

3.1.23.1 Fingerprint

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

ANSWER properties	
transaction type	DA9
struct name	answer_series_bo
segmented	true

3.1.23.2 Related Messages

BU9

3.1.23.3 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).

3.1.23.4 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
}
```

3.1.23.5 Usage and conditions

Series

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

3.1.23.6 Answer Structure

The DA9 ANSWER has the following structure:

```
struct answer_series_bo {
   struct transaction type
   char[8] date trading s  // Date, Trading
   UINT16 T segment number n  // Segment Number
   UINT16_T items n  // Items
   Array ITEM [max no: 330] {
      struct series  // Named struct no: 50000
      struct upper level series
      INT32 T contract size i  // Contract Size
      INT32 T price quot factor i  // Price, Quotation Factor
      UINT16 T state number n  // Trading State Number
      char[32] ins id s  // Series, Identity
      char[12] isin code s  // ISIN Code
      UINT8 T stopped by issue c  // Stopped By Issue
      char[12] isin code old s  // ISIN Code, Old Series
      char[8] date notation s  // Date, Notation
      char[8] date last trading s  // Date, Last Trading
      char[6] time last trading s  // Time, Last Trading
      char[8] date delivery start s  // Date, Delivery Start
      char[8] date delivery stop s  // Date, Delivery Stop
      UINT8 T deliverable c  // Deliverable
      UINT8 T suspended c  // Suspended
      UINT8_T series status c  // Series, Status
      UINT8 T tm template c  // Template Series
      UINT8 T tm series c  // Tailor Made Series
      char[8] settlement date s  // Date, Settlement
      char[8] start date s  // Date, Start
      char[8] end date s  // Date, End
      UINT8 T accept collateral c  // Accepted as Collateral
      char[8] date first trading s  // Date, First Trading
```

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

3.1.23.7 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.

3.1.24 DQ10 [Instrument Class QUERY]

3.1.24.1 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

3.1.24.2 Related Messages

BU10

3.1.24.3 Purpose

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

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

3.1.24.4 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
}
```

3.1.24.5 Usage and conditions

Series

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

3.1.24.6 Answer Structure

The DA10 ANSWER has the following structure:

```
struct answer_instrument_class {
   struct transaction type
   UINT16_T segment_number_n  // Segment Number
   UINT16_T items_n  // Items
   Array ITEM [max no: 145] {
      struct series  // Named struct no: 50000
      struct upper_level_series
      INT32_T price_quot_factor_i  // Price, Quotation Factor
      INT32_T contract_size_i  // Contract Size
      INT32_T exerc_limit_i  // Exercise, Limit
      INT32_T redemption_value_i  // Redemption Value
      INT32_T min_qty_increment_i  // Minimum Quantity Increment
      UINT16_T derivate_level_n  // Derivate Level
      UINT16_T dec_in_strike_price_n  // Decimals, Strike Price
      UINT16_T dec_in_contr_size_n  // Decimals, Contract Size
      UINT16_T rnt_id_n  // Ranking Type
      UINT16_T dec_in_premium_n  // Decimals, Premium
      UINT16_T items_n  // Items
      Array ITEM [max no: 12] {
         struct tick_size
      }
      UINT16_T dec_in_deliv_n  // Decimals, Delivery
      UINT16_T items_block_n  // Item, Block
      Array BLOCK_SIZE [max no: 4] {
         INT64_T maximum_size_u  // Block Size, Maximum Volume
         UINT32_T minimum_size_n  // Block Size, Minimum Volume
         UINT32_T block_n  // Block Size
         UINT8_T lot_type_c  // Lot, Type
         char[3] filler_3_s  // Filler
      }
      UINT16_T cleared_dec_in_qty_n  // Decimals, Quantity
      UINT16_T virt_commodity_n  // Virtual Underlying
      UINT16_T dec_in_fixing_n  // Decimals, Fixing
      char[3] base_cur_s  // Currency, Trading
      UINT8_T traded_c  // Traded
      UINT8_T exerc_limit_unit_c  // Exercise, Limit Unit
      char[14] inc_id_s  // Instrument Class, Identity
      char[10] trc_id_s  // Trade Report Class
      char[32] name_s  // Name
      CHAR is_fractions_c  // Fraction, Premium
      UINT8_T price_format_c  // Premium/Price Format
      UINT8_T strike_price_format_c  // Strike Price, Format
      UINT8_T cabinet_format_c  // Cabinet Format
      UINT8_T price_unit_premium_c  // Price Unit, Premium
      UINT8_T price_unit_strike_c  // Price Unit, Strike
      char[32] settl_cur_id_s  // Currency, Settlement
      char[3] credit_class_s  // Credit Class
      char[12] csd_id_s  // CSD, Identity
      UINT8_T trd_cur_unit_c  // Traded Currency Unit
      UINT8_T collateral_type_c  // Collateral types
      UINT8_T fixing_req_c  // FIXING_REQ_C
      CHAR[2] mbs_id_s  // Minimum Bid Schedule
      char[12] valuation_group_id_s  // VAG, Identity ; Of type: VAG_ID_S
      char[3] filler_3_s  // Filler
   }
}
```

3.1.24.7 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**.

3.1.25 DQ12 [Account Type QUERY]

3.1.25.1 Fingerprint

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

ANSWER properties	
transaction type	DA12
struct name	answer_account_type
segmented	true

3.1.25.2 Related Messages

BU12

3.1.25.3 Purpose

This query retrieves all existing account types in the system.

3.1.25.4 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
}
```

3.1.25.5 Answer Structure

The DA12 ANSWER has the following structure:

```
struct answer_account_type {
   struct transaction type
   UINT16 T segment number n  // Segment Number
   UINT16 T items n  // Items
   Array ITEM [max no: 100] {
      char[12] acc type s  // Account Type
      char[40] description s  // Description
      UINT8 T open close c  // Open or Closed
      UINT8 T transitory c  // Transitory
      UINT8 T market_maker_c  // Market Maker
      UINT8 T own inventory c  // Own Inventory
      UINT8_T exclusive_opening_sell_c  // Exclusive Opening Sell
      UINT8 T positions allowed c  // Positions, Allowed
      UINT8 T trades allowed c  // Trades, Allowed
      char[12] atr id s  // Account Type Rule
      CHAR origin c  // Origin, Account Type
   }
}
```

3.1.26 DQ13 [Account Fee Type QUERY]

3.1.26.1 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

3.1.26.2 Related Messages

BU13

3.1.26.3 Purpose

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

3.1.26.4 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
}
```

3.1.26.5 Answer Structure

The DA13 ANSWER has the following structure:

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

3.1.27 DQ14 [Underlying Adjustment QUERY]

3.1.27.1 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

3.1.27.2 Purpose

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

3.1.27.3 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
}
```

3.1.27.4 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.

3.1.27.5 Answer Structure

The DA14 ANSWER has the following structure:

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

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

3.1.27.6 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.

3.1.28 DQ15 [Converted Series QUERY]

3.1.28.1 Fingerprint

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

QUERY properties	
partitioned	false
segmented	true
answers	DA15

ANSWER properties	
transaction type	DA15
struct name	answer_converted_series
segmented	true

3.1.28.2 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.

3.1.28.3 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
}
```

3.1.28.4 Usage and Conditions

Adjustment Identifier

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

3.1.28.5 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
```

```
    }
}
```

3.1.28.6 Answer, comments

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

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

Series, Old

is the series before adjustment.

Series, New

is the series after adjustment.

Contract Size

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

3.1.29 DQ18 [Non-Trading Days QUERY]

3.1.29.1 Fingerprint

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

ANSWER properties	
transaction type	DA18
struct name	answer_non_trading_days
segmented	true

3.1.29.2 Related Messages

BU18

3.1.29.3 Purpose

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

3.1.29.4 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
}
```

3.1.29.5 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**.

3.1.29.6 Answer Structure

The DA18 ANSWER has the following structure:

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

3.1.30 DQ19 [Underlying Backoffice QUERY]

3.1.30.1 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

3.1.30.2 Related Messages

BU19

3.1.30.3 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).

3.1.30.4 Structure

The DQ19 QUERY has the following structure:

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

3.1.30.5 Usage and conditions

Series

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

3.1.30.6 Answer Structure

The DA19 ANSWER has the following structure:

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

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

3.1.30.7 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.

3.1.31 DQ20 [Instrument Class Backoffice QUERY]

3.1.31.1 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

3.1.31.2 Related Messages

BU20

3.1.31.3 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).

3.1.31.4 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
}
```

3.1.31.5 Usage and conditions

Series

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

3.1.31.6 Answer Structure

The DA20 ANSWER has the following structure:

```
struct answer_instrument_class {
   struct transaction type
   UINT16_T segment_number_n  // Segment Number
   UINT16_T items_n  // Items
   Array ITEM [max no: 145] {
      struct series  // Named struct no: 50000
      struct upper level series
      INT32_T price_quot_factor_i  // Price, Quotation Factor
      INT32_T contract_size_i  // Contract Size
      INT32_T exerc_limit_i  // Exercise, Limit
      INT32_T redemption_value_i  // Redemption Value
      INT32_T min_qty_increment_i  // Minimum Quantity Increment
      UINT16_T derivate_level_n  // Derivate Level
      UINT16_T dec_in_strike_price_n  // Decimals, Strike Price
      UINT16_T dec_in_contr_size_n  // Decimals, Contract Size
      UINT16_T rnt_id_n  // Ranking Type
      UINT16_T dec_in_premium_n  // Decimals, Premium
      UINT16_T items_n  // Items
      Array ITEM [max no: 12] {
         struct tick size
      }
      UINT16_T dec_in_deliv_n  // Decimals, Delivery
      UINT16_T items_block_n  // Item, Block
      Array BLOCK_SIZE [max no: 4] {
         INT64_T maximum_size_u  // Block Size, Maximum Volume
         UINT32_T minimum_size_n  // Block Size, Minimum Volume
         UINT32_T block_n  // Block Size
         UINT8_T lot_type_c  // Lot, Type
         char[3] filler_3_s  // Filler
      }
      UINT16_T cleared_dec_in_qty_n  // Decimals, Quantity
      UINT16_T virt_commodity_n  // Virtual Underlying
      UINT16_T dec_in_fixing_n  // Decimals, Fixing
      char[3] base_cur_s  // Currency, Trading
      UINT8_T traded_c  // Traded
      UINT8_T exerc_limit_unit_c  // Exercise, Limit Unit
      char[14] inc_id_s  // Instrument Class, Identity
      char[10] trc_id_s  // Trade Report Class
      char[32] name_s  // Name
      CHAR is_fractions_c  // Fraction, Premium
      UINT8_T price_format_c  // Premium/Price Format
      UINT8_T strike_price_format_c  // Strike Price, Format
      UINT8_T cabinet_format_c  // Cabinet Format
      UINT8_T price_unit_premium_c  // Price Unit, Premium
      UINT8_T price_unit_strike_c  // Price Unit, Strike
      char[32] settl_cur_id_s  // Currency, Settlement
      char[3] credit_class_s  // Credit Class
      char[12] csd_id_s  // CSD, Identity
      UINT8_T trd_cur_unit_c  // Traded Currency Unit
      UINT8_T collateral_type_c  // Collateral types
      UINT8_T fixing_req_c  // FIXING_REQ_C
      CHAR[2] mbs_id_s  // Minimum Bid Schedule
      char[12] valuation_group_id_s  // VAG, Identity ; Of type: VAG_ID_S
      char[3] filler_3_s  // Filler
   }
}
```

3.1.31.7 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**.

3.1.32 DQ22 [Instrument Type Backoffice QUERY]

3.1.32.1 Fingerprint

QUERY properties	
transaction type	DQ22
calling sequence	omniapi_query_ex

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

ANSWER properties	
transaction type	DA22
struct name	answer_instrument
segmented	true

3.1.32.2 Purpose

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

3.1.32.3 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
}
```

3.1.32.4 Usage and conditions

Series

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

3.1.32.5 Answer Structure

The DA22 ANSWER has the following structure:

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

3.1.32.6 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.

3.1.33 DQ23 [Market Backoffice QUERY]

3.1.33.1 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

3.1.33.2 Purpose

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

3.1.33.3 Structure

The DQ23 QUERY has the following structure:

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

3.1.33.4 Usage and Conditions

Series

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

3.1.33.5 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
   }
}
```

3.1.33.6 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.

3.1.34 DQ24 [Exchange QUERY]

3.1.34.1 Fingerprint

QUERY properties	
transaction type	DQ24
calling sequence	omniapi_query_ex

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

ANSWER properties	
transaction type	DA24
struct name	answer_exchange_da24
segmented	true

3.1.34.2 Purpose

This query provides information on all exchanges in the system.

3.1.34.3 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
}
```

3.1.34.4 Usage and conditions

Series

must be zeroed.

3.1.34.5 Answer Structure

The DA24 ANSWER has the following structure:

```
struct answer_exchange_da24 {
   struct transaction type
   UINT16_T segment_number_n  // Segment Number
   UINT16_T items_n  // Items
   Array ITEM [max no: 100] {
      struct da24 {
         UINT8_T country_c  // Country Number
         CHAR opra_indicator_c  // OPRA Indicator
         char[32] name_s  // Name
         char[4] exchange_short_s  // Exchange, Short Name
         char[2] country_id_s  // Name, Country
         char[40] tz_exchange_s  // Time Zone, Exchange
         char[12] master_clh_id_s  // Master CLH, Identity
         char[2] country_s  // Country
         char[8] date_implementation_s  // Date, Implementation
         char[2] filler_2_s  // Filler
      }
   }
}
```

3.1.34.6 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.

3.1.35 DQ44 [Legal Account Instrument QUERY]

3.1.35.1 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	copy_list_gen
segmented	false

3.1.35.2 Purpose

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

3.1.35.3 Structure

The DQ44 QUERY has the following structure:

```
struct query_legal_account_instrument {
    struct transaction type
    struct series  // Named struct no: 50000
    UINT16 T segment number n  // Segment Number
    char[2] filler 2 s  // Filler
}
```

3.1.35.4 Answer Structure

The DA44 ANSWER has the following structure:

```
struct copy_list_gen {
    struct transaction type
    UINT16 T items n  // Items
    char[2] filler 2 s  // Filler
    Array ITEM [max no: 50] {
        struct ucl_gen {
            char[2] country id s  // Name, Country
            char[5] ex customer s  // Customer, Identity
            CHAR filler 1 s  // Filler
            UINT16 T copies n  // COPIES N
            Array RECIPIENT [max no: 100] {
                struct user code
            }
        }
    }
}
```

3.1.36 DQ45 [Trade Report Type QUERY]

3.1.36.1 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

3.1.36.2 Purpose

This query is used to retrieve all trade report types.

3.1.36.3 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
}
```

3.1.36.4 Usage and conditions

Series

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

3.1.36.5 Answer Structure

The DA45 ANSWER has the following structure:

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

3.1.36.6 Answer, comments

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

3.1.37 DQ46 [Deal Source QUERY]

3.1.37.1 Fingerprint

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

ANSWER properties	
transaction type	DA46
struct name	answer_deal_source
segmented	true

3.1.37.2 Purpose

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

3.1.37.3 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
}
```

3.1.37.4 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
    }
}
```

3.1.37.5 Answer, comments

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

3.1.38 DQ78 [Exception Days QUERY]

3.1.38.1 Fingerprint

QUERY properties	
transaction type	DQ78
calling sequence	omniapi_query_ex
struct name	query_exception_days
facility	EP0
partitioned	false
segmented	true
answers	DA78

ANSWER properties	
transaction type	DA78
struct name	answer_exception_days
segmented	true

3.1.38.2 Related Messages

BU78

3.1.38.3 Purpose

The purpose of this query is to retrieve the exception days defined on Market, Instrument Type and Instrument Class level.

3.1.38.4 Structure

The DQ78 QUERY has the following structure:

```
struct query_exception_days {
    struct transaction_type
```

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

3.1.38.5 Usage and conditions

An exception day is a day when an alternative Trading Session is used instead of the normal trading session. An exception day can also define that the market is open on a weekday that normally is closed.

Series

Series should be null-filled to retrieve all exception days for all markets, instrument types and instrument classes.

Series should be filled with Country and Market to retrieve all exception days for all markets, instrument types and instrument classes that are connected to the specified market.

3.1.38.6 Answer Structure

The DA78 ANSWER has the following structure:

```
struct answer_exception_days {
   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[8] date_exception_s  // Date, Exception
   }
}
```

3.1.38.7 Answer, comments

The answer received contains a list of exception days and series where series can represent a market, an instrument type or an instrument class.

One item is sent for each combination of series and exception day.

Each response is prefaced with the Transaction Type (DA78) and an item field specifying the number of records contained in the response.

3.1.39 DQ120 [Delta Underlying QUERY]

3.1.39.1 Fingerprint

QUERY properties	
transaction type	DQ120
calling sequence	omniapi_query_ex
struct name	query_delta

QUERY properties	
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

3.1.39.2 Related Messages

BU120

3.1.39.3 Purpose

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

3.1.39.4 Concept of Delta Queries and Broadcasts

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

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

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

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

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

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

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

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

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

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

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

Example

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

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

3.1.39.5 Structure

The DQ120 QUERY has the following structure:

```
struct query_delta
```

3.1.39.6 Usage and Conditions

Full Answer Timestamp

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

Download Reference Number

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

3.1.39.7 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
      }
   }
}
```

3.1.39.8 Answer, comments

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

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

Full Answer Timestamp

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

Download Reference Number

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

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

3.1.40 DQ121 [Delta Underlying for Back Office QUERY]

3.1.40.1 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

3.1.40.2 Related Messages

BU121

3.1.40.3 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.

3.1.40.4 Structure

The DQ121 QUERY has the following structure:

```
struct query_delta
```

3.1.40.5 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**.

3.1.40.6 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
      }
   }
}
```

3.1.40.7 Answer, comments

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

3.1.41 DQ122 [Delta Instrument Class QUERY]

3.1.41.1 Fingerprint

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

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

3.1.41.2 Related Messages

BU122

3.1.41.3 Purpose

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

3.1.41.4 Structure

The DQ122 QUERY has the following structure:

```
struct query_delta
```

3.1.41.5 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**.

3.1.41.6 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
```

```
            }
        }
    }
```

3.1.41.7 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.

3.1.42 DQ123 [Delta Instrument Class for Back Office QUERY]

3.1.42.1 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

3.1.42.2 Related Messages

BU123

3.1.42.3 Purpose

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

3.1.42.4 Structure

The DQ123 QUERY has the following structure:

```
struct query delta
```

3.1.42.5 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**.

3.1.42.6 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 leg calc rule  // Named struct no: 37115
         struct ns price tick corr  // Named struct no: 37113
      }
   }
}
```

3.1.42.7 Answer, comments

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

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

3.1.43 DQ124 [Delta Instrument Series QUERY]

3.1.43.1 Fingerprint

QUERY properties	
transaction type	DQ124
calling sequence	omniapi_query_ex
struct name	query_delta

QUERY properties	
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

3.1.43.2 Related Messages

BU124

3.1.43.3 Purpose

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

3.1.43.4 Structure

The DQ124 QUERY has the following structure:

```
struct query_delta
```

3.1.43.5 Usage and Conditions

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

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

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

3.1.43.6 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
      }
   }
}
```

3.1.43.7 Answer, comments

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

3.1.44 DQ125 [Delta Instrument Series for Back Office QUERY]

3.1.44.1 Fingerprint

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

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

3.1.44.2 Related Messages

BU125

3.1.44.3 Purpose

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

3.1.44.4 Structure

The DQ125 QUERY has the following structure:

```
struct query delta
```

3.1.44.5 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.

3.1.44.6 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
         struct ns inst series leg flow  // Named struct no: 37309
      }
   }
}
```

3.1.44.7 Answer, comments

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

3.2 Market Status

3.2.1 BI1 [Resumption and Suspension of Trading BROADCAST]

3.2.1.1 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

3.2.1.2 Purpose

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

3.2.1.3 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
}
```

3.2.2 BI41 [Instrument Status Information BROADCAST]

3.2.2.1 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

3.2.2.2 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.

3.2.2.3 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
   }
}
```

3.2.2.4 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 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

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

Expiration Date
Strike Price

can in some cases be zero for a series.

Trading State Number

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

Level

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

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

Seconds to State Change

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

3.2.3 UQ15 [Instrument Status QUERY]

3.2.3.1 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

ANSWER properties	
struct name	answer_instrument_status
segmented	true

3.2.3.2 Purpose

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

3.2.3.3 Structure

The UQ15 QUERY has the following structure:

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

3.2.3.4 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

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

Level

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

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

3.2.3.5 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.

3.2.3.6 Answer Structure

The UA15 ANSWER has the following structure:

```
struct answer_instrument_status {
   struct transaction type
   UINT16 T segment number n  // Segment Number
   UINT16 T items n  // Items
   Array ITEM [max no: 1000] {
      struct series  // Named struct no: 50000
      UINT16 T state number n  // Trading State Number
      UINT16 T state level e  // Level
   }
}
```

3.2.3.7 Answer, comments

Series

Series, completed with one of the following:

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

Segment Number

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

3.3 Trade and Position Management

3.3.1 BD6 [Dedicated Trade Information VIB]

3.3.1.1 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

3.3.1.2 Related Messages

CQ10

CQ11

3.3.1.3 Purpose

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

3.3.1.4 Structure

The BD6 VIB has the following structure:

```
struct broadcast hdr
Sequence {
   struct sub item hdr
   Choice {
      struct cl trade base api  // Named struct no: 3
      struct cl trade secur part  // Named struct no: 20
   }
}
```

3.3.1.5 Usage and Conditions

This is a variable broadcast.

The first structure after the header part is always cl_trade_base_api. In addition to that, none or several structures can follow; each preceded by a header.

On systems using BD6 the queries CQ10 and CQ11 are used in conjunction to recover trades.

When retrieving trades disseminated with BD6, the actual data structure is a sequence starting with:

- cl_trade_base_api (named struct no = 3)

3.3.1.6 Structure Contents

Exchange Info

is equivalent to the Passthrough Information field in cl_trade_api.

Date, As of and Time, As of

fields contain information about when the deal was closed or the original trade was registered (in case of rectify or overtaking trade). It is the same data as Time Stamp, last change, but in “business time” format.

Time Stamp, last change

contains date and time the deal was closed, propagated from the MP subsystem (VMS format).

Sequence Number

is assigned each broadcast to allow for a recipient to verify that no trade broadcasts are lost and to indicate the order in which they were sent. The sequence number is unique per participant and instrument type, meaning that the same trade has different sequence numbers for different recipients.

3.3.2 BD18 [Dedicated Delivery BROADCAST]

3.3.2.1 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

3.3.2.2 Related Messages

CQ52, CQ53

3.3.2.3 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.

3.3.2.4 Structure

The BD18 BROADCAST has the following structure:

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

3.3.2.5 Usage and Conditions

All recipients are handled within their organisation, which means that all deliveries to a customer that belongs to an organisation is sent to the customer that is defined centrally to be the organisation owner.

To interpret the information correctly it is important to remember some clearing system fundamentals:

- Every entity that in some respect can change ownership involves a series, be it money or an ordinary financial product.
- The change of ownership itself is called a delivery.
- Everything that happens to a series during its lifetime is defined through product

events.

- Product events are always released through a stimulus (often regarded as being the same thing as the event itself).

Sequence Number

The Sequence Number is sequential for each customer, instrument type and clearing date. This number can be used by the customer to discover missed dedicated delivery information. To recover a missed dedicated delivery broadcast, use the Delivery query.

Date

contains the date on which this delivery is created, that is the current business date.

Series

contains the binary series from which this delivery emanates. If, for example, this delivery is due to an exercise of a stock option. The series field contains the stock option series.

Original Delivery Number
Original Key Number

are only defined when Delivery type is either rollback or overtaking. In these cases these fields together with series, points out the delivery that this delivery either rolls back or overtakes. These fields are zero when Delivery Type is Normal.

Delivery Type

defines the types Normal, Rollback and Overtaking.

Originator Type

is set to Reversing if this delivery is created from a trade and the trade type on this trade is reversing. Otherwise this field is Normal.

Delivery State

defines if this delivery is active or rectified. When the delivery is sent as a broadcast it is always Normal.

Customer Account

is the Customer and Account for the Clearing Entity, Trade or Position, that this delivery is created from.

Delivery Account

is the account that handles the delivery for the Customer. This information is defined on Account level in the central system and is either Settlement Propagation or Delivery Propagation. If no propagation is set for the account, this field has the same value as **Customer Account**.

Delivery Account will for a DVP hold the account configured to handle deliveries for the clearing account. For other items, it will hold the configured settlement account.

Clearing Account

is the account that holds the position account. For a BD18 originating from a trade, **Clearing Account** will have the account set from Position Propagation on the trading account. If no propagation is set for the account, this field has the same value as **Customer Account**.

For a BD18 originating from a Position, **Clearing Account** has the same value as **Customer Account**.

Quantity, Delivery Base

defines the calculated quantity for the delivery. The sign is set from the clearing house's point of view (i.e. is delivered from the clearing house). The number of decimals used is specified by the decimals in premium in the DQ4/DQ123 query, for the class of the series defined in the Delivery Base.

Delivery Number, Key Number

gives together with country, market and instrument group in the Series field a unique combination for this delivery.

Origin, Delivery Number

defines the origin for this delivery. When the field value is different from Delivery Number it defines a trade number from which this delivery is calculated. The trade is then identified with this field and country, market, and instrument group from the Series field.

Settlement Date

defines the date when this delivery is to be settled.

Quantity, Delivery

defines the quantity for which this delivery is calculated from. It can be a trade quantity or a position amount.

Delivery, Base

is a series that defines what is delivered. The quantity for this is defined in the **Quantity, Delivery Base**.

Class Number

is a number indicating type of settlement for a delivery item.

3.3.3 BD29 [Directed Give Up BROADCAST]

3.3.3.1 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

3.3.3.2 Related Messages

CQ61, CQ76

3.3.3.3 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.

3.3.3.4 Structure

The BD29 BROADCAST has the following structure:

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

3.3.3.5 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.”
- 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.

3.3.4 BD39 [Dedicated Trade Change Information BROADCAST]

3.3.4.1 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

3.3.4.2 Related Messages

Dedicated Trade Information Broadcast and CQ39

3.3.4.3 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.

3.3.4.4 Structure

The BD39 BROADCAST has the following structure:

```
struct directed_trade_change {
   struct broadcast_type
   struct cl_trade_change_api
}
```

3.3.4.5 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.

3.3.5 BD40 [Dedicated auxiliary position info update information BROADCAST]

3.3.5.1 Fingerprint

BROADCAST properties	
transaction type	BD40
calling sequence	omniapi_read_event_ext_ex or omniapi_read_event_block
struct name	directed_pos_info_update
info type	dedicated

3.3.5.2 Related Messages

CQ40

3.3.5.3 Purpose

This broadcast is disseminated when any of the auxiliary information associated with a position is updated.

3.3.5.4 Structure

The BD40 BROADCAST has the following structure:

```
struct directed_pos_info_update {
    struct broadcast_type
    struct pos_info_update_api
}
```

3.3.5.5 Usage and conditions

The auxiliary information consists of :

- quantity to be exempted from automatic/general exercise (deny exercise)
- quantity closed-out current clearing date
- quantity of underlying used as cover for short position (exchange specific)

The time for the most recent update of auxiliary information on the position is provided as modified time and date.

The time and date from the most recently received BD40 is intended to be used as input to a CQ40 query transaction in order to retrieve the information distributed in BD40 broadcasts while the API-client is disconnected.

3.3.6 CC11 [Cancel Holding Rectify Trade TRANSACTION]

3.3.6.1 Fingerprint

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

3.3.6.2 Related Messages

CQ14, CQ15

3.3.6.3 Purpose

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

3.3.6.4 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
}
```

3.3.6.5 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.

3.3.7 CC12 [Cancel Holding Rectify Deal TRANSACTION]

3.3.7.1 Fingerprint

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

3.3.7.2 Related Messages

CQ16, CQ17

3.3.7.3 Purpose

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

3.3.7.4 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
}
```

3.3.7.5 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.

3.3.8 CC13 [Exercise Request TRANSACTION]

3.3.8.1 Fingerprint

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

3.3.8.2 Purpose

The purpose of this transaction is to request an exercise.

3.3.8.3 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
}
```

3.3.8.4 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.

3.3.9 CC14 [Deny Exercise Request TRANSACTION]

3.3.9.1 Fingerprint

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

3.3.9.2 Purpose

The purpose of this transaction is to inform the Central System that a certain quantity for an account should not participate in an automatic exercise. If this quantity exceeds the held position, the whole position is excluded from automatic exercise.

3.3.9.3 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
}
```

3.3.10 CC15 [Cancel Exercise Request TRANSACTION]

3.3.10.1 Fingerprint

TRANSACTION properties	
transaction type	CC15
calling sequence	omniapi_tx_ex

TRANSACTION properties	
struct name	annul_exercise_req
facility	EP3
partitioned	false

3.3.10.2 Related Messages

CQ21

3.3.10.3 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**.

3.3.10.4 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
}
```

3.3.10.5 Usage and conditions

Series

must be set to the Series of the exercise request to be cancelled.

Exercise Request Number

must be set to the exercise request number identifying the exercise request to be cancelled.

3.3.11 CC38 [Confirm Give up Request TRANSACTION]

3.3.11.1 Fingerprint

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

3.3.11.2 Related Messages

CQ61

3.3.11.3 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.

3.3.11.4 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
   }
}
```

3.3.11.5 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.

3.3.12 CC40 [Reject Give up Request TRANSACTION]

3.3.12.1 Fingerprint

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

3.3.12.2 Related Messages

CQ61

3.3.12.3 Purpose

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

3.3.12.4 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
}
```

3.3.12.5 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.

3.3.13 CC41 [Modify Commission Table TRANSACTION]

3.3.13.1 Fingerprint

TRANSACTION properties	
transaction type	CC41
calling sequence	omniapi_tx_ex
struct name	modify_commission
facility	EP5
partitioned	false

3.3.13.2 Related Messages

CQ64, BI71

3.3.13.3 Purpose

This transaction is used to modify data in the commission table.

3.3.13.4 Structure

The CC41 TRANSACTION has the following structure:

```
struct modify_commission {
   struct transaction_type
   struct series  // Named struct no: 50000
   struct party
   UINT8_T send_or_receive_c  // Send or Receive
   CHAR filler_1_s  // Filler
   UINT16_T items_n  // Items
   Array ITEM [max no: 800] {
      struct series  // Named struct no: 50000
      struct party
      char[10] account_id_s  // Account, Identity
      char[15] customer_info_s  // Customer, Information
      char[8] created_date_s  // Date, Created
      char[6] created_time_s  // Time, Created
      char[12] user_code_s  // User Code
      CHAR filler_1_s  // Filler
      INT32_T commission_i  // Commission
   }
}
```

3.3.13.5 Usage and conditions

Party

identifies the member that sends or receives a give-up. Party must contain the country and customer identity.

Each commission item contains the same type of item information as the answer to the Query Commission Table CQ64 transaction does.

3.3.14 CD5 [Transitory Account Trades TRANSACTION]

3.3.14.1 Fingerprint

TRANSACTION properties	
transaction type	CD5
calling sequence	omniapi_tx_ex
struct name	cl_reregistration_bo
facility	EP3
partitioned	false

3.3.14.2 Purpose

This transaction is used to transfer trades from the daily account to the client account. It is used by the Back Office (application) and identifies a trade by using the unique Trade Number.

3.3.14.3 Structure

The CD5 TRANSACTION has the following structure:

```
struct cl_reregistration_bo {
   struct transaction_type
   struct series  // Named struct no: 50000
   UINT8_T items_c  // Item
   char[3] filler_3_s  // Filler
   Array ITEM [max no: 100] {
      struct account
      INT32_T trade_number_i  // Trade Number
      INT64_T deal_quantity_i  // Quantity, Deal
      char[15] customer_info_s  // Customer, Information
      char[2] reserved_2_s  // Reserved
      CHAR reserved_1_c  // Reserved
      UINT8_T open_close_req_c  // Open_ Close Request
      CHAR filler_1_s  // Filler
   }
}
```

3.3.14.4 Usage and conditions

Series

must be completely specified.

This function is related only to Client Clearing and thus not valid for Member Clearing. In a client clearing model, the Exchange provides the clearing service on anonymous client identities for the customers.

A certain trade can be transferred to one or several client accounts. It is possible to request how the positions should be updated. This transaction, a synchronous transaction, will allow the choices open, close, and normal.

If a close order cannot be executed for CD5, an error message will be returned.

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.

A Daily Account Trades transaction can only be canceled on the same business day as it is created.

3.3.15 CD28 [Rectify Trade TRANSACTION]

3.3.15.1 Fingerprint

TRANSACTION properties	
transaction type	CD28
calling sequence	omniapi_tx_ex
struct name	rectify_trade
facility	EP3
partitioned	false

3.3.15.2 Related Messages

CD27

3.3.15.3 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.

3.3.15.4 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
   }
}
```

3.3.15.5 Usage and conditions

Series
Trade number

identify the trade to be rectified.

Item

the number of overtaking trades to be created by the rectification.

Account

the desired destination account of an overtaking trade.

Open Close Request

the desired Open Close Request of the overtaking trade.

Customer Information

the desired Customer Information of the overtaking trade.

Quantity, Trade

the desired quantity of a overtaking trade. The sum of the quantities of the overtaking trades must equal the quantity of the trade to be rectified.

3.3.16 CD31 [Rectify Deal TRANSACTION]

3.3.16.1 Fingerprint

TRANSACTION properties	
transaction type	CD31

TRANSACTION properties	
calling sequence	omniapi_tx_ex
struct name	rectify_deal
facility	EP3
partitioned	false

3.3.16.2 Purpose

A deal rectification transaction is used for changing a whole deal or to cancel it.

3.3.16.3 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
   }
}
```

3.3.16.4 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
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

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.

3.3.16.5 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.

3.3.17 CD32 [Average Price Trade TRANSACTION]

3.3.17.1 Fingerprint

TRANSACTION properties	
transaction type	CD32
calling sequence	omniapi_tx_ex
struct name	average_price_trade
facility	EP3
partitioned	false

3.3.17.2 Related Messages

CQ16, CQ17, CC12

3.3.17.3 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.

3.3.17.4 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
   }
}
```

3.3.17.5 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**.

3.3.17.6 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.

3.3.18 CD35 [Give up Request TRANSACTION]

3.3.18.1 Fingerprint

TRANSACTION properties	
transaction type	CD35
calling sequence	omniapi_tx_ex
struct name	give_up_request
facility	EP3
partitioned	false

3.3.18.2 Purpose

This transaction is used to give up a trade to another member.

3.3.18.3 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
}
```

3.3.18.4 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.

3.3.19 CD38 [Long Position Adjustment TRANSACTION]

3.3.19.1 Fingerprint

TRANSACTION properties	
transaction type	CD38
calling sequence	omniapi_tx_ex
struct name	long_position_adj
facility	EP3
partitioned	false

3.3.19.2 Purpose

The purpose of this transaction is to net a position by closing an equal amount of long and short contracts respectively.

3.3.19.3 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
        }
    }
```

3.3.19.4 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.

3.3.20 CD54 [Position Closeout QUERY]

3.3.20.1 Fingerprint

QUERY properties	
transaction type	CD54
calling sequence	omniapi_query_ex
struct name	position_closeout
facility	EP3
partitioned	true
answers	CA54

ANSWER properties	
transaction type	CA54
struct name	position_closeout_status
segmented	false

3.3.20.2 Related Messages

CQ122, CQ123, CD55

3.3.20.3 Purpose

The purpose of this transaction is to allow closeout of a collection of positions.

3.3.20.4 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
   }
}
```

3.3.20.5 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.
- 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.

3.3.20.6 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
   }
}
```

3.3.21 CD55 [Restore Position TRANSACTION]

3.3.21.1 Fingerprint

TRANSACTION properties	
transaction type	CD55
calling sequence	omniapi_tx_ex
struct name	restore_position
facility	EP3
partitioned	true

3.3.21.2 Related Messages

CQ122, CQ123, CD54

3.3.21.3 Purpose

The purpose of this transaction is to allow reinstatement of a previously closed-out position.

3.3.21.4 Structure

The CD55 TRANSACTION has the following structure:

```
struct restore_position {
   struct transaction_type
   struct series  // Named struct no: 50000
   struct account
   INT64_T closeout_qty_i  // Quantity, Close out
   char[8] date_s  // Date
}
```

3.3.21.5 Usage and conditions

Series

- Series identifies together with account the position.
- Series must be completely filled and identify an existing series.

Account

- Account identifies together with Series the position.
- Account must identify a specific account, wildcards is not supported.

Closeout Quantity

is the quantity to be reinstated.

Date

- is the Clearing date for which the position should be reinstated.
- must be current Clearing date or a prior Clearing date.

3.3.22 CQ3 [Position QUERY]

3.3.22.1 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

3.3.22.2 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.

3.3.22.3 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
}
```

3.3.22.4 Usage and conditions

Series

must be complete up to **Country number**, **Market code** and **Instrument group**.

Segment Number

is one for the first query and then incremented.

Search Series
Account

identifies the positions to be returned in the answer.

Date

must be valid and have one of the following values:

- Previous calendar date: The overnight (O/N) position is returned. These positions are static during the day.
- Today's business date. The current position for the current clearing date (provided it exists for the instrument) is returned.
- Next calendar date. The current position for the next clearing date is returned; trades as of next clearing date are added to the current clearing date position.

Note that the previous and next calendar date is in relation to current business date in the system. For example, the previous calendar date will refer to a Sunday when current business date is a Monday.

3.3.22.5 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
         }
      }
```

3.3.22.6 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).

3.3.23 CQ8 [Fixing Values QUERY]

3.3.23.1 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

3.3.23.2 Purpose

This transaction retrieves fixing value for cash settled products (on a daily basis, when they are exercised or when they are closed).

3.3.23.3 Structure

The CQ8 QUERY has the following structure:

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

3.3.23.4 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.

3.3.23.5 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
   }
}
```

3.3.23.6 Answer, comments

If the maximum number of items for one transaction is returned, the query should be repeated with incremented segment number.

3.3.24 CQ10 [Query missing trade QUERY]

3.3.24.1 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

3.3.24.2 Related Messages

BD6 (Dedicated Trade Information VIB)

CQ11 (Query Missing Trade, Historical Query).

3.3.24.3 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.

3.3.24.4 Structure

The CQ10 QUERY has the following structure:

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

```
    INT32_T sequence_first_i  // Number, First Sequential
    INT32_T sequence_last_i  // Number, Last Sequential
    char[8] date_s  // Date
}
```

3.3.24.5 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.

3.3.24.6 Answer Structure

The CA10 VIA has the following structure:

```
struct answer_missing_trade_hdr {
   struct transaction_type
   char[2] filler_2_s  // Filler
   UINT16_T items_n  // Items
}
Sequence {
   struct item_hdr
   Sequence {
      struct sub_item_hdr
      Choice {
         struct cl_trade_base_api  // Named struct no: 3
         struct cl_trade_secur_part  // Named struct no: 20
      }
   }
}
```

3.3.24.7 Answer, comments

The answer is built up with variable trade structures. Each trade is built with several sub-structures to form a flow of data in which each trade can consist of one or several structures. A trade consists at least of the structure cl_trade_base_api. Each sub-structure is prefaced with a header.

3.3.25 CQ11 [Query missing trade, historical QUERY]

3.3.25.1 Fingerprint

QUERY properties	
transaction type	CQ11
calling sequence	omniapi_query_ex
struct name	query_api_trade
facility	EP5
partitioned	false
answers	CA11

VIA properties	
transaction type	CA11
struct name	The message complies with the VIM concept and has no top struct. The sequence of possible structs is described in the Structure section.
segmented	false

3.3.25.2 Related Messages

BD6 (Dedicated Trade Information VIB) and CQ10 (Query Missing Trade Query).

3.3.25.3 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.

3.3.25.4 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
}
```

3.3.25.5 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.

3.3.25.6 Answer Structure

The CA11 VIA has the following structure:

```
struct answer_api_trade_hdr {
   struct transaction_type
   struct series  // Named struct no: 50000
   char[8] from_date_s  // Date, From
   INT32_T sequence_first_i  // Number, First Sequential
   UINT16_T items_n  // Items
   char[2] filler_2_s  // Filler
}
Sequence {
   struct item_hdr
   Sequence {
      struct sub_item_hdr
      Choice {
         struct cl_trade_base_api  // Named struct no: 3
         struct cl_trade_secur_part  // Named struct no: 20
      }
   }
}
```

3.3.25.7 Answer, comments

See CQ10.

3.3.26 CQ14 [Holding Rectify Trade QUERY]

3.3.26.1 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

3.3.26.2 Related Messages

CQ15, CC11

3.3.26.3 Purpose

This query is used for retrieving information on requests to rectify trades. The query will only return information on requests that initially were placed in a holding state awaiting confirmation by the exchange or clearinghouse. Whether a request to rectify a trade requires confirmation or not depends on the exchange/clearinghouse policy.

Use CQ15 to get detailed information regarding a holding rectify trade.

Use CC11 to withdraw ("reject") a request to rectify a trade.

3.3.26.4 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
}
```

3.3.26.5 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.

3.3.26.6 Answer Structure

The CA14 ANSWER has the following structure:

```
struct answer_rectify_t {
   struct transaction_type
   UINT16_T segment_number_n  // Segment Number
   char[2] reserved_2_s  // Reserved
   struct partition_low
   struct partition_high
   UINT16_T items_n  // Items
   UINT8_T instance_next_c  // Next Instance Number
   CHAR filler_1_s  // Filler
   Array ITEM [max no: 400] {
      struct ans_rect_t_item {
         char[8] created_date_s  // Date, Created
         char[6] created_time_s  // Time, Created
         char[8] asof_date_s  // Date, As Of
         char[6] asof_time_s  // Time, As Of
         char[8] clearing_date_s  // Clearing Date
         char[8] orig_clearing_date_s  // Clearing Date, Original
         struct trading_code
         struct user_code
         struct series  // Named struct no: 50000
         INT32_T trade_number_i  // Trade Number
         INT32_T rectify_trade_number_i  // Rectify Trade Number
         INT32_T ext_seq_nbr_i  // External Clearinghouse, Sequence Number
         UINT8_T state_c  // State
         UINT8_T bought_or_sold_c  // Bought or Sold
         UINT8_T reserved_prop_c  // Reserved Properties
         CHAR filler_1_s  // Filler
         struct new_account {
            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
                INT64_T trade_quantity_i  // Quantity, Trade
                INT32_T deal_price_i  // Price, Deal
            }
        }
    }
```

3.3.26.7 Answer, comments

If the maximum number of items for one transaction is returned, the query should be repeated with incremented segment number.

Date, Created
Time, Created

Creation date and time for rectify trade request.

Date, As Of
Time, As Of

Match date and time for trade subject to rectify.

Clearing Date

Clearing date for processing of rectify transaction.

Clearing Date, Original

Original Clearing date for processing of trade subject to rectify.

TRADING_CODE

Identifies user submitting the rectify trade request.

USER

Identifies user confirming or rejecting the rectify trade request.

Series

Instrument in trade subject to rectify trade request.

Trade Number

Together with instrument type of traded seres, Trade Number identifies the trade subject to rectify trade request.

Rectify Trade Number

Together with instrument type of traded seres, Rectify Trade Number identifies the rectify trade request.

External Clearing House, Sequence Number

sequence number provided by external exchange system, optional.

State

returns current state of reqeust: Holding, Active or Rejected.

Bought or Sold

indicates whether trade corresponds to bought or sold contracts.

Reserved Properties

Not applicable.

NEW_ACCOUNT

New account for trade - set to "*" if trade is moved to several accounts.

ACCOUNT

account into which trade is allocated prior to rectify operation.

Quantity, Trade

quantity in trade subject to rectify.

Price, Deal

price in trade subject to rectify.

3.3.27 CQ15 [Detailed Holding Rectify Trade QUERY]

3.3.27.1 Fingerprint

QUERY properties	
transaction type	CQ15
calling sequence	omniapi_query_ex
struct name	query_rectify_t_cont
facility	EP3
partitioned	false
answers	CA15

ANSWER properties	
transaction type	CA15
struct name	answer_rectify_ext_cont
segmented	false

3.3.27.2 Related Messages

CQ14, CC11

3.3.27.3 Purpose

This query is used for receiving detailed information about a holding rectify trade.

3.3.27.4 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
}
```

3.3.27.5 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**.

3.3.27.6 Answer Structure

The CA15 ANSWER has the following structure:

```
struct answer_rectify_ext_cont {
   struct transaction_type
   UINT16_T items_n  // Items
   char[2] filler_2_s  // Filler
   Array ITEM [max no: 100] {
      struct account
      INT64_T trade_quantity_i  // Quantity, Trade
      UINT8_T open_close_req_c  // Open  Close Request
      char[15] customer_info_s  // Customer, Information
   }
}
```

3.3.28 CQ16 [Holding Rectify Deal QUERY]

3.3.28.1 Fingerprint

QUERY properties	
transaction type	CQ16
calling sequence	omniapi_query_ex

QUERY properties	
struct name	query_rectify_d
facility	EP3
partitioned	false
answers	CA16

ANSWER properties	
transaction type	CA16
struct name	answer_rectify_d
segmented	false

3.3.28.2 Related Messages

CQ17, CC12

3.3.28.3 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.

3.3.28.4 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
}
```

3.3.28.5 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.

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.

3.3.28.6 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
      }
   }
}
```

3.3.28.7 Answer, comments

If the maximum number of items for one transaction is returned, the query should be repeated with incremented segment number.

3.3.29 CQ17 [Detailed Rectify Deal QUERY]

3.3.29.1 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

3.3.29.2 Related Messages

CQ16, CC12

3.3.29.3 Purpose

This transaction gives detailed information of the trades included in a specified rectified deal in state holding.

3.3.29.4 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
}
```

3.3.29.5 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. 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**.

3.3.29.6 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
   }
}
```

3.3.30 CQ19 [Account Propagation QUERY]

3.3.30.1 Fingerprint

QUERY properties	
transaction type	CQ19
calling sequence	omniapi_query_ex
struct name	query_account_prop
facility	EP5
partitioned	false
answers	CA19

ANSWER properties	
transaction type	CA19
struct name	answer_propagate
segmented	false

3.3.30.2 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.

3.3.30.3 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
}
```

3.3.30.4 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

3.3.30.5 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
   }
}
```

3.3.30.6 Answer, comments

If the maximum number of items for one transaction is returned, the query should be repeated with incremented segment number.

3.3.31 CQ20 [Open Interest QUERY]

3.3.31.1 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

3.3.31.2 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.

3.3.31.3 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
}
```

3.3.31.4 Usage and conditions

Series

must be completed with **Country Number**, **Market Code** and **Instrument Group**.

Segment Number

is one for the first query and then incremented.

Search Series

identifies the series for which data is to be returned in the answer.

Date

must be filled with current business date.

3.3.31.5 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
   }
}
```

3.3.31.6 Answer, comments

If the maximum number of items for one transaction is returned, the query should be repeated with incremented segment number.

3.3.32 CQ21 [Pending Exercise Request QUERY]

3.3.32.1 Fingerprint

QUERY properties	
transaction type	CQ21
calling sequence	omniapi_query_ex
struct name	query_exercise_req
facility	EP3
partitioned	true
answers	CA21

ANSWER properties	
transaction type	CA21

ANSWER properties	
struct name	answer_exercise_req
segmented	false

3.3.32.2 Related Messages

CC15

3.3.32.3 Purpose

The purpose of this query is to retrieve all pending exercise requests. Use CC15 to either confirm or reject the pending exercise request.

3.3.32.4 Structure

The CQ21 QUERY has the following structure:

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

3.3.32.5 Usage and conditions

Series

must be completed with **Country Number**, **Market Code** and **Instrument Group**.

Segment Number

is one for the first query and then incremented.

Search Series
Account

identify the pending exercise requests for which data is to be returned in the answer.

3.3.32.6 Answer Structure

The CA21 ANSWER has the following structure:

```
struct answer_exercise_req {
   struct transaction_type
   struct partition_low
   struct partition_high
   UINT16_T segment_number_n  // Segment Number
   UINT16_T items_n  // Items
   Array ITEM [max no: 250] {
      struct series  // Named struct no: 50000
      struct account
      CHAR reserved_1_c  // Reserved
      char[2] reserved_2_s  // Reserved
      CHAR filler_1_s  // Filler
      struct trading_code
      struct ex_user_code
      char[8] modified_date_s  // Date, Modified
      char[6] modified_time_s  // Time, Modified
      char[8] asof_date_s  // Date, As Of
      char[6] asof_time_s  // Time, As Of
      INT64_T quantity_i  // Quantity
      INT32_T trade_number_i  // Trade Number
      INT32_T exercise_number_i  // Exercise, Request Number
      UINT8_T state_c  // State
      char[3] filler_3_s  // Filler
   }
}
```

3.3.32.7 Answer, comments

If the maximum number of items for one transaction is returned, the query should be repeated with incremented segment number.

3.3.33 CQ22 [Error Message QUERY]

3.3.33.1 Fingerprint

QUERY properties	
transaction type	CQ22
calling sequence	omniapi_query_ex
struct name	query_error_msg
facility	EP5
partitioned	false
answers	CA22

ANSWER properties	
transaction type	CA22
struct name	answer_error_msg
segmented	true

3.3.33.2 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.

3.3.33.3 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
}
```

3.3.33.4 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.

3.3.33.5 Answer Structure

The CA22 ANSWER has the following structure:

```
struct answer_error_msg {
   struct transaction_type
   struct partition_low
   struct partition_high
   UINT16_T segment_number_n  // Segment Number
   UINT16_T items_n  // Items
   Array ITEM [max no: 100] {
      struct trading_code
      struct series  // Named struct no: 50000
      struct account
      char[8] created_date_s  // Date, Created
      char[6] created_time_s  // Time, Created
      char[10] error_operation_s  // Error, Operation
      UINT32_T error_id_u  // Error Identity
      char[40] error_problem_s  // Error, Problem
```

```
    }
}
```

3.3.33.6 Answer, comments

If the maximum number of items for one transaction is returned, the query should be repeated with incremented segment number.

3.3.34 CQ31 [Simulate Fee QUERY]

3.3.34.1 Fingerprint

QUERY properties	
transaction type	CQ31
calling sequence	omniapi_query_ex
struct name	query_simulate_fee
facility	EP3
partitioned	false
answers	CA31

ANSWER properties	
transaction type	CA31
struct name	answer_delivery
segmented	false

3.3.34.2 Purpose

This query calculates the fees for a particular trade. The fees are returned as delivery information (see Answer below).

3.3.34.3 Structure

The CQ31 QUERY has the following structure:

```
struct query_simulate_fee {
    struct transaction_type
    struct series  // Named struct no: 50000
    INT32_T deal_price_i  // Price, Deal
    INT64_T deal_quantity_i  // Quantity, Deal
    struct account
    UINT8_T bid_or_ask_c  // Bid or Ask
    UINT8_T open_close_req_c  // Open  Close Request
    char[2] filler_2_s  // Filler
}
```

3.3.34.4 Usage and conditions

Series
Price, Deal
Quantity, Deal
Account
Bid or Ask
Open Close Request

define the characteristics of the trade and must be specified in order for the central system to be able to calculate the fee data

3.3.34.5 Answer Structure

The CA31 ANSWER has the following structure:

```
struct answer_delivery {
   struct transaction type
   struct partition low
   struct partition high
   UINT16 T segment number n  // Segment Number
   UINT16 T items n  // Items
   Array ITEM [max no: 100] {
      char[8] date_s  // Date
      INT32_T event_type_i  // Stimuli Event
      struct series  // Named struct no: 50000
      struct account
      INT32 T class no i  // Class Number
      INT64 T deliv base quantity q  // Quantity, Delivery Base
      char[8] settlement date s  // Date, Settlement
      INT64 T delivery quantity q  // Quantity, Delivery
      struct deliv base
   }
}
```

3.3.34.6 Answer, comments

Quantity, Delivery Base

identifies the number of **Delivery Base** to deliver/receive. The sign is set from the clearinghouse’s point of view (i.e. is delivered from the clearinghouse). The number of decimals used in the Quantity, Delivery Base is specified by the decimals in price in the Query Underlying Transaction, see**DQ4** (referring to the **Delivery Base**).

Delivery Base

identifies what to deliver.

In the answer Quantity, Delivery Base and Quantity, Delivery is summarized per Date; Event Type; Series; Customer; Account; Class Number; Date, Settlement; and Delivery Base.

3.3.35 CQ32 [Deal Capture Missing Exercise By Exeption QUERY]

3.3.35.1 Fingerprint

QUERY properties	
transaction type	CQ32
calling sequence	omniapi_query_ex
struct name	ced_query_missing_exbyex_proxy
facility	EP5
partitioned	false
answers	CA32

ANSWER properties	
transaction type	CA32
struct name	ced_answer_missing_exbyex_proxy
segmented	false

3.3.35.2 Purpose

This query is used to recover and get synchronized again after a missing sequence number is detected in the Dedicated Exercise Information flow.

3.3.35.3 Structure

The CQ32 QUERY has the following structure:

```
struct ced_query_missing_exbyex_proxy {
   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] yyyymmdd s  // Date
   char[2] country id s  // Name, Country
   char[5] ex customer s  // Customer, Identity
   CHAR filler 1 s  // Filler
}
```

3.3.35.4 Usage and conditions

A member can query for own Exercise Information only. A GCM member can, however, do queries on behalf of their NCMs.

Series

should be zeroed.

Sequence first

specifies the first missing sequence number. **Sequence first** must be > 0. A value of 0 in **Sequence last** means all sequence numbers starting with **Sequence first** should be returned.

Sequence last

specifies the last missing sequence number. A value of 0 in **Sequence last** means all sequence numbers starting with **Sequence first** should be returned.

Date

specifies the date for which information is requested. It could be today's date or a previous date. How many previous dates that are available for recovery is defined by the exchange.

Country id and External customer

GCM members can specify an on-behalf member by filling in the Country id and External customer fields with a valid NCM. If not used, the Country id should be filled with 2 character lines, completed with trailing spaces, and External customer should be filled with 5 character lines, completed with trailing spaces.

3.3.35.5 Return Codes

A CQ32 transaction may also be aborted by the system, in which case only the reason for the transaction being aborted is returned to the sender.

Cstatus	Txstat
Successful	Normal
Transaction aborted	CL_EX_UNKNOWNNCMMEM - NCM member specified in transaction is unknown
Transaction aborted	CL_EX_ILLEGNCMMEM - NCM member is illegal for this GCM member
Transaction aborted	CL_EX_SEQCROSSED - Last sequence number is lower than first sequence number
Transaction aborted	CL_EX_SEQISZERO - Sequence number zero is illegal
Transaction aborted	...

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

3.3.35.6 Answer Structure

The CA32 ANSWER has the following structure:

```
struct ced_answer_missing_exbyex_proxy {
   struct transaction_type
   UINT16_T items_n   // Items
   char[2] filler_2_s   // Filler
   Array ITEM [max no: 400] {
      struct ced_exercise_info {
         struct trading_code
         struct series  // Named struct no: 50000
         INT32_T rqst_type_i  // RQST_TYPE_I
         INT32_T sequence_number_i  // Sequence Number
         INT64_T quantity_i  // Quantity
         INT32_T ext_status_i  // Return Status
         UINT8_T confirm_reject_c  // Confirm or Reject
         char[8] yyyymmdd_s  // Date
         char[6] hhmmss_s  // Time, External
         char[10] account_id_s  // Account, Identity
         char[15] customer_info_s  // Customer, Information
         INT32_T total_ex_day_i  // TOTAL_EX_DAY_I
         INT32_T history_ex_i  // HISTORY_EX_I
      }
   }
}
```

3.3.35.7 Answer, comments

After a successful CQ32 transaction, a list of missing Exercise Information is returned to the sender. Each response is prefaced with a header. Thereafter each record contains information that corresponds to the contents of a Dedicated Exercise Information Broadcast (BD12).

3.3.36 CQ36 [Average Price Trade QUERY]

3.3.36.1 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

3.3.36.2 Purpose

This query returns the trade number of the trades that are part of an average price trade.

3.3.36.3 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
}
```

3.3.36.4 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.

3.3.36.5 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
   }
}
```

3.3.36.6 Answer, comments

If the maximum number of items for one transaction is returned, the query should be repeated with incremented segment number.

3.3.37 CQ38 [Account QUERY]

3.3.37.1 Fingerprint

QUERY properties	
transaction type	CQ38
calling sequence	omniapi_query_ex
struct name	query_account
facility	EP5
partitioned	false
answers	CA38

ANSWER properties	
transaction type	CA38
struct name	answer_account_ext
segmented	true

3.3.37.2 Purpose

The purpose of this query is to retrieve account information for own accounts.

3.3.37.3 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
}
```

3.3.37.4 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.

3.3.37.5 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
   }
}
```

3.3.37.6 Answer, comments

If the maximum number of items for one transaction is returned, the query should be repeated with incremented segment number.

3.3.38 CQ39 [Trade Change QUERY QUERY]

3.3.38.1 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

ANSWER properties	
segmented	false

3.3.38.2 Related Messages

CQ10, BD39

3.3.38.3 Purpose

The purpose of this query is to retrieve missing trade change broadcasts.

3.3.38.4 Structure

The CQ39 QUERY has the following structure:

```
struct query_missing_trade_change {
   struct transaction_type
   struct series  // Named struct no: 50000
   UINT8_T instance_c  // Instance, Number
   char[3] filler_3_s  // Filler
   INT32_T sequence_first_i  // Number, First Sequential
   INT32_T sequence_last_i  // Number, Last Sequential
   char[8] date_s  // Date
}
```

3.3.38.5 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).

3.3.38.6 Answer Structure

The CA39 ANSWER has the following structure:

```
struct answer_missing_trade_change {
   struct transaction_type
   char[2] filler_2_s  // Filler
   UINT16_T items_n  // Items
   Array ITEM [max no: 1000] {
      struct cl_trade_change_api
   }
}
```

3.3.39 CQ40 [Auxiliary position info updated QUERY]

3.3.39.1 Fingerprint

QUERY properties	
transaction type	CQ40
calling sequence	omniapi_query_ex
struct name	query_updated_pos_info
facility	EP3
partitioned	true
answers	CA40

ANSWER properties	
transaction type	CA40
struct name	answer_updated_pos_info
segmented	true

3.3.39.2 Related Messages

BD40, CQ3

3.3.39.3 Purpose

This query is used for retrieving auxiliary information associated with positions that have been updated since a specified date and time.

3.3.39.4 Structure

The CQ40 QUERY has the following structure:

```
struct query_updated_pos_info {
   struct transaction_type
   struct series  // Named struct no: 50000
   struct search_series
   struct account
   UINT16_T segment_number_n  // Segment Number
   char[8] modified_date_s  // Date, Modified
   char[6] modified_time_s  // Time, Modified
}
```

3.3.39.5 Usage and conditions

The query is intended to be used after login for recovering from any missed BD40 broadcasts while the API-client was disconnected.

The auxiliary information consists of :

- quantity to be exempted from automatic/general exercise (deny exercise)
- quantity closed-out current clearing date
- quantity of underlying used as cover for short position (exchange specific)

Series

must be complete up to Country Number, Market Code and Instrument Group.

Segment Number

is one for the first query and then incremented.

Search Series Account

identifies the positions for which auxiliary information is to be returned in the answer.

3.3.39.6 Answer Structure

The CA40 ANSWER has the following structure:

```
struct answer_updated_pos_info {
   struct transaction_type
   struct partition_low
   struct partition_high
   UINT16_T segment_number_n  // Segment Number
   UINT16_T items_n  // Items
   Array ITEM [max no: 900] {
      struct pos_info_update_api
   }
}
```

3.3.39.7 Answer, comments

If the maximum number of items for one transaction is returned, the query should be repeated with incremented segment number.

3.3.40 CQ52 [Delivery QUERY]

3.3.40.1 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

3.3.40.2 Related Messages

BD18, CQ53

3.3.40.3 Purpose

This query retrieves deliveries. For example, if a missing sequence number is detected for the Delivery Dedicated broadcast (BD18), this query is used to get synchronized with the broadcast flow again.

3.3.40.4 Structure

The CQ52 QUERY has the following structure:

```
struct query_missing_delivery {
   struct transaction_type
   struct series  // Named struct no: 50000
   INT32_T sequence_first_i  // Number, First Sequential
   INT32_T sequence_last_i  // Number, Last Sequential
   char[8] date_s  // Date
}
```

3.3.40.5 Usage and conditions

This transaction retrieves deliveries for the current business day, to query for historical deliveries, use CQ53.

Series

must be completed with **Country Number**, **Market Code** and **Instrument Group**.

Number, first sequential

is the first missing one.

Number, last sequential

is the last missing one. If the Number, last sequential is equal to zero, all available deliveries are sent in sequence.

Date

must hold the current Clearing date for the Instrument Type in question.

Class Number

is a number indicating type of settlement for a delivery item.

3.3.40.6 Answer Structure

The CA52 ANSWER has the following structure:

```
struct answer_missing_delivery {
   struct transaction_type
   char[2] filler_2_s   // Filler
   UINT16_T items_n   // Items
   Array ITEM [max no: 280] {
      struct cl_delivery_api
   }
}
```

3.3.40.7 Answer, comments

If the maximum number of items for one transaction is returned, the query should be repeated with Number, First sequential set to the next missing sequence number after the Sequence Number of the last received item.

Apart from the header each record in the response contains the same information as the **directed_delivery** struct in the Delivery Dedicated broadcast (BD18).

Date

must hold the current business date.

3.3.41 CQ53 [Delivery History QUERY]

3.3.41.1 Fingerprint

QUERY properties	
transaction type	CQ53
calling sequence	omniapi_query_ex
struct name	query_api_delivery
facility	EP5
partitioned	true
answers	CA53

ANSWER properties	
transaction type	CA53
struct name	answer_api_delivery
segmented	false

3.3.41.2 Related Messages

BD18, CQ52

3.3.41.3 Purpose

This query retrieves historical deliveries. The information is available to the trading member and the clearing member the next trading day. To retrieve deliveries for the current trading day, use CQ52.

3.3.41.4 Structure

The CQ53 QUERY has the following structure:

```
struct query_api_delivery {
   struct transaction type
   struct series  // Named struct no: 50000
   char[8] from date s  // Date, From
   INT32 T sequence first i  // Number, First Sequential
   char[8] to date s  // Date, To
   INT32 T sequence last i  // Number, Last Sequential
}
```

3.3.41.5 Usage and conditions

The historical delivery information is available to the members the next business day and is queried per instrument type.

Series

must be completed with **Country Number**, **Market Code** and **Instrument Group**.

Date, From
Date, To

must be Clearing Dates that are historical dates compared to current Clearing date. **Date, From** must be less or equal to **Date, To**.

Number, first sequential

is the first item to get for **Date, From**. Zero or one means the first item for that date.

Number, last sequential

is the last item to get for **Date, To**. Zero means the last item for that date.

Class Number

is a number indicating type of settlement for a delivery item.

3.3.41.6 Answer Structure

The CA53 ANSWER has the following structure:

```
struct answer_api_delivery {
   struct transaction type
   struct series  // Named struct no: 50000
   char[8] from date s  // Date, From
   INT32 T sequence first i  // Number, First Sequential
   UINT16 T items n  // Items
   char[2] filler 2 s  // Filler
   Array ITEM [max no: 280] {
      struct cl delivery api
   }
}
```

3.3.41.7 Answer, comments

Date

contains the date on which this delivery was created.

Apart from the header each record in the response contains the same information as the **directed_delivery** struct in the Delivery Dedicated broadcast (BD18).

If all deliveries that reside centrally are to be fetched, the following sequence must be performed:

Loop for all instrument types defined, except for country = 255, market = 255 and instrument group = 255.

For each instrument type, do a CQ53 query until CA53 signals that no more deliveries exist.

The first CQ53 is filled with the following parameters:

- Series, filled with current instrument type.
- Date, From. Set to ‘19000101’.
- Sequence Number 1. Set to 1.
- Date, To. Set to yesterday’s date.
- Sequence Number 2. Set to 0.

If Number, first sequential in CA53 is greater than zero, more CQ53 queries must be done to retrieve data. CQ53 must be filled with the following parameters:

- Series, filled with series in CA53.
- Date, From. Filled with Date, From in CA53.
- Sequence Number 1. Filled with Sequence Number 1 in CA53.
- Date, To. Set to yesterday’s date.
- Sequence Number 2. Set to 0.

3.3.42 CQ61 [Holding Give Up Request QUERY]

3.3.42.1 Fingerprint

QUERY properties	
transaction type	CQ61

QUERY properties	
calling sequence	omniapi_query_ex
struct name	query_give_up_request
facility	EP3
partitioned	false
answers	CA61

ANSWER properties	
transaction type	CA61
struct name	answer_give_up_request
segmented	true

3.3.42.2 Related Messages

CC38

CC40

BD29

CQ76

CQ77

3.3.42.3 Purpose

The query returns Give-up requests in a holding state, but may also return Give-up requests in other states depending on the query criteria (see below). The answer contains information to facilitate the tracking of give-ups and their origins.

3.3.42.4 Structure

The CQ61 QUERY has the following structure:

```
struct query_give_up_request {
   struct transaction type
   struct series  // Named struct no: 50000
   struct party
   UINT32_T ext_trade_number_u  // Trade Number, External
   UINT16_T segment_number_n  // Segment Number
   UINT8_T state_c  // State
   CHAR buy_or_sell_c  // Buy or Sell
   UINT8_T send_or_receive_c  // Send or Receive
   char[8] created_date_s  // Date, Created
   char[32] series_id_s  // Series, Identity
   char[2] country_id_s  // Name, Country
   char[5] ex_customer_s  // Customer, Identity
   char[30] give_up_text_s  // Give Up, Free Text
   char[2] filler_2_s  // Filler
}
```

3.3.42.5 Usage and conditions

Note: It is recommended to use BD29/CQ76 instead of CQ61.

Facility EP3 should be used for current date and facility EP5 for historic dates.

The query is only partitioned when used on facility EP3.

Use CC38 to confirm or reject a Give-up request.

Series

must be complete up to **Country Number**, **Market Code** and **Instrument Group**. Determines clearing partition when querying for current business date on facility EP3.

Date, Created

must be filled with the business date when the Give-up request was created.

Segment Number

should be set to 1 for retrieving the first answer segment from a partition and then incremented for retrieval of subsequent answer segments.

State

has the following impact on the returned give-up requests in the answer:

0	all give-ups are returned regardless of state
1	Holding
5	Completed
6	Rejected

Series Id

should contain an explicit series name or a series wildcard string.

Send or Receive

defines the interpretation of the member (**Name, Country** and **Customer, Identity**) and **Party** field.

When set to ‘1’ (send), the member field is used for filtering of the participant initiating the **Give-Up** and the **Party** fields are used for filtering the receiving/destination member for the give-up.

If set to ‘2’ (receive), the member field is used for filtering of the participant receiving **Give-Up** and the **Party** fields are used used for filtering the member initiating the give-up.

Country, Name and Customer Identity

specifies give-up/take-up member (participant id) for filtering give-up.

Wildcard search/filtering can be used. Must be filled with “*”, “*” when doing a wildcard search

Party

specifies take-up/give-up member (participant id) for filtering give-up.

Wildcard search/filtering can be used. Must be filled with “*”, “*” when doing a wildcard search.

Buy or Sell

allows for filtering on give-ups on buy (1) or sell (2) trades.

Filtering will not be applied if set to 0.

Give Up, Free Text

allows searching for give-up(s) with specified “Free text”.

Wildcard search/filtering can be used. Must be set to “*” when doing a wildcard search.

Trade Number, External

allows searching for give-up(s) on trade(s) with specified external trade number.

External trade number on trades is not used by all exchanges.

Must be set to 0 when doing a wildcard search.

3.3.42.6 Answer Structure

The CA61 ANSWER has the following structure:

```
struct answer_give_up_request {
   struct transaction type
   struct partition low
   struct partition high
   UINT16 T segment number n  // Segment Number
   UINT16 T items n  // Items
   Array ITEM [max no: 420] {
      struct series  // Named struct no: 50000
      struct account
      struct party
      INT32 T give up number i  // Give Up, Number
      INT64 T trade quantity i  // Quantity, Trade
      INT32 T deal price i  // Price, Deal
      INT32 T trade number i  // Trade Number
      INT32 T commission i  // Commission
      UINT8 T bought or sold c  // Bought or Sold
      UINT8 T state c  // State
      char[8] created date s  // Date, Created
      char[6] created time s  // Time, Created
      char[30] give up text s  // Give Up, Free Text
      char[8] asof date s  // Date, As Of
      char[6] asof time s  // Time, As Of
      char[8] orig clearing date s  // Clearing Date, Original
      UINT8 T old trade c  // Old Trade Indicator
      CHAR ext trade fee type c  // External Trade, Fee Type
      UINT8 T deal source c  // Deal Source
      UINT8 T reserved prop c  // Reserved Properties
      char[8] clearing date s  // Clearing Date
      UINT32 T ext trade number u  // Trade Number, External
      UINT32 T orig ext trade number u  // Trade Number, Original External
   }
}
```

3.3.42.7 Answer, comments

Account

describes the destination member in the giveup. The 10 last characters may be left blank, thus only defining the member, or set to point out a specific account.

Party

identifies the customer that gives up the trade.

Deal source

data refer to the original trade’s deal source. Please refer to the detailed field descriptions for further information.

The following fields describe the trade that is subject to the giveup:

- Series
- Party
- Bought or Sold
- Quantity, Trade
- Price, Deal
- Trade Number
- Date, Created
- Time, Created
- Date, As Of
- Time, As Of
- Original Clearing Date
- Old Trade Indicator
- Deal Source
- External Trade Fee Type
- Trade Number, External
- Original Trade Number, External

The Quantity, Trade field specifies the give-up portion of the trade.

Of these, Date, As Of; Time, As Of; Original Clearing Date; Old Trade Indicator; Deal Source; External Trade Fee Type only contain significant data for give-up requests made the current business day and whose states are either holding or completed.

Give-Up Number; State; Account; Give-Up Free Text, Clearing Date are fields that describe the giveup. Clearing Date is the clearing date of the giveup itself.

If the maximum number of items for one transaction is returned, the query should be repeated with incremented segment number.

3.3.43 CQ62 [Confirm Give Up Request QUERY]

3.3.43.1 Fingerprint

QUERY properties	
transaction type	CQ62
calling sequence	omniapi_query_ex
struct name	query_conf_give_up_req_items
facility	EP5
partitioned	false
answers	CA62

ANSWER properties	
transaction type	CA62
struct name	answer_conf_give_up_req_items
segmented	false

3.3.43.2 Related Messages

CC38, CQ61

3.3.43.3 Purpose

This query returns the give-up items sent when a giveup was confirmed. This query can only be sent for a confirmed giveup.

3.3.43.4 Structure

The CQ62 QUERY has the following structure:

```
struct query_conf_give_up_req_items {
   struct transaction type
   struct series  // Named struct no: 50000
   INT32_T give_up_number_i  // Give Up, Number
}
```

3.3.43.5 Usage and conditions

Use CQ61 to query for Give-up requests in holding state.

Use CC38 to reject or confirm holding Give-up requests.

Series

must contain the whole series for the giveup.

Give up number

identifies the give-up.

3.3.43.6 Answer Structure

The CA62 ANSWER has the following structure:

```
struct answer_conf_give_up_req_items {
   struct transaction type
   UINT16 T items n   // Items
   char[2] filler 2 s   // Filler
   Array ITEM [max no: 50] {
      struct account
      INT64 T trade quantity i   // Quantity, Trade
      UINT8 T open close req c   // Open  Close Request
      char[15] customer info s   // Customer, Information
   }
}
```

3.3.43.7 Answer, comments

This information is the same information as sent in the Confirm Give-Up Trade Transaction, see **CC38**.

3.3.44 CQ64 [Commission Table QUERY]

3.3.44.1 Fingerprint

QUERY properties	
transaction type	CQ64
calling sequence	omniapi_query_ex
struct name	query_commission
facility	EP5
partitioned	false
answers	CA64

ANSWER properties	
transaction type	CA64
struct name	answer_commission
segmented	false

3.3.44.2 Related Messages

CC41, BI71

3.3.44.3 Purpose

This query returns commission rules from the commission table for the specified member.

3.3.44.4 Structure

The CQ64 QUERY has the following structure:

```
struct query_commission {
   struct transaction_type
   struct series  // Named struct no: 50000
   struct party
   UINT8_T send_or_receive_c  // Send or Receive
   char[3] filler_3_s  // Filler
}
```

3.3.44.5 Usage and conditions

Series

select records from the table and may optionally contain *Country Number*, *Market Code*, *Instrument Group*, *Commodity* or the full series.

Party

identifies the member that sends or receives a give-up. Party must contain the country and customer identity.

3.3.44.6 Answer Structure

The CA64 ANSWER has the following structure:

```
struct answer_commission {
   struct transaction_type
   struct party
   UINT8_T send_or_receive_c  // Send or Receive
   CHAR filler_1_s  // Filler
   UINT16_T items_n  // Items
   Array ITEM [max no: 800] {
      struct series  // Named struct no: 50000
      struct party
      char[10] account_id_s  // Account, Identity
      char[15] customer_info_s  // Customer, Information
      char[8] created_date_s  // Date, Created
      char[6] created_time_s  // Time, Created
      char[12] user_code_s  // User Code
      CHAR filler_1_s  // Filler
      INT32_T commission_i  // Commission
```

```
        }
    }
```

3.3.44.7 Answer, comments

Party, Send or Receive

contain the same information as in the query.

Series, Party, Account, Customer Information

describe a commission rule. These fields do not necessarily contain any data.

Commission

is the default commission to the receiver of a give-up or the commission expected by the receiver of a give-up.

3.3.45 CQ65 [Level Position QUERY]

3.3.45.1 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

3.3.45.2 Related Messages

CQ3

3.3.45.3 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.

3.3.45.4 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
}
```

3.3.45.5 Usage and conditions

Account

If the field Account contains any wildcards, the **Summary** field must be set to1 (yes); the query transaction will otherwise be aborted with an error-status.

Account Type

When filled must either be a valid account type name or a valid wildcard representation of an Account Type name. If Account Type is not blank, only positions on accounts with an Account Type matching the argument is returned in the answer.

Level Type

specifies the account level of interest; origin or margin.

Segment Number

is one for the first query and then incremented.

Series Id

should contain an explicit series name or a series wildcard string.

Summary

specifies whether to return the aggregated positions on the specified account level or if the individual position items are to be returned.

Summary =2 (no) is only applicable if the field **Customer Account**does not contain any wildcards, i.e. it identifies a single account. In that case, one may retrieve all the individual ‘position items’ making up the aggregated (and “propagated”) position on a margin or origin account.

Date

must be valid and have one of the following values:

- Previous calendar date: The overnight (O/N) position is returned. These positions are static during the day.
- Today’s business date. The current position for the current clearing date (provided it exists for the instrument) is returned.
- Next calendar date. The current position for the next clearing date is returned; trades as of next clearing date are added to the current clearing date position.

Note that the previous and next calendar date is in relation to current business date in the system. For example, the previous calendar date will refer to a Sunday when current business date is a Monday.

This query is used when the account structure makes it relevant to ask for Origin Level and Margin Level accounts. Use Position Information Transaction, see **CQ3**, for an ordinary account level query.

3.3.45.6 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
   }
}
```

3.3.45.7 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.

3.3.46 CQ68 [Clearing Date QUERY]

3.3.46.1 Fingerprint

QUERY properties	
transaction type	CQ68
calling sequence	omniapi_query_ex
struct name	query_clearing_date
facility	EP5
partitioned	false
answers	CA68

ANSWER properties	
transaction type	CA68
struct name	answer_clearing_date
segmented	false

3.3.46.2 Purpose

The purpose of this query is to retrieve information on the current and the next clearing date for instrument types.

3.3.46.3 Structure

The CQ68 QUERY has the following structure:

```
struct query_clearing_date {
   struct transaction_type
   struct series  // Named struct no: 50000
   struct search_series
}
```

3.3.46.4 Usage and conditions

Series, Search

may be zeroed to retrieve clearing date information on all instrument types handled by a particular clearing server.

3.3.46.5 Answer Structure

The CA68 ANSWER has the following structure:

```
struct answer_clearing_date {
   struct transaction_type
   struct partition_low
   struct partition_high
   char[16] omex_version_s  // OMEX Version
   char[8] business_date_s  // Date, Business
   UINT16_T items_n  // Items
   char[2] filler_2_s  // Filler
   Array ITEM [max no: 1000] {
      struct series  // Named struct no: 50000
      char[8] clearing_date_s  // Clearing Date
      char[8] next_clearing_date_s  // Clearing Date, Next
      char[8] prev_clearing_date_s  // Clearing Date, Previous
      CHAR tra_cl_next_day_c  // Cleared Next Day
      char[3] filler_3_s  // Filler
   }
}
```

3.3.46.6 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.

3.3.47 CQ72 [Net Open Interest QUERY]

3.3.47.1 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

3.3.47.2 Related Messages

CQ20 – Open Interest

3.3.47.3 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.

3.3.47.4 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
}
```

3.3.47.5 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.

3.3.47.6 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
      }
   }
```

3.3.47.7 Answer, comments

If the maximum number of items for one transaction is returned, the query should be repeated with incremented segment number.

3.3.48 CQ76 [Give Up QUERY]

3.3.48.1 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

3.3.48.2 Related Messages

BD29

3.3.48.3 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.

3.3.48.4 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
}
```

3.3.48.5 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.

3.3.48.6 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
   }
}
```

3.3.48.7 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.

3.3.49 CQ77 [Give Up History QUERY]

3.3.49.1 Fingerprint

QUERY properties	
transaction type	CQ77
calling sequence	omniapi_query_ex
struct name	query_api_give_up
facility	EP5
partitioned	true
answers	CA77

ANSWER properties	
transaction type	CA77
struct name	answer_api_give_up
segmented	false

3.3.49.2 Related Messages

CQ76

3.3.49.3 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.

3.3.49.4 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
}
```

3.3.49.5 Usage and conditions

Series

must be completed with **Country Number**, **Market Code** and **Instrument Group**.

Date, From
Date, To

must be Clearing Dates that are historical dates compared to current Clearing date. Clearing Date, From must be less or equal to Clearing Date, To.

Sequence Number 1

is the first item to get for **Clearing Date, From**. Zero or one means the first item for that date.

Sequence Number 2

is the last item to get for **Clearing Date, To**. Zero means the last item for that date.

3.3.49.6 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
   }
}
```

3.3.49.7 Answer, comments

Apart from the header each record in response contains the same information as directed_give_up_t.

If all giveups that reside centrally are to be fetched, the following sequence must be performed:

Loop for all instrument types defined, except for country = 255, market = 255 and instrument group = 255.

For each instrument type, do a CQ77 query until CA77 signals that no more give ups exist.

The first CQ77 is filled with the following parameters:

- Series, filled with current instrument type.
- Clearing Date, From. Set to ‘19000101’.
- Sequence Number 1. Set to 1.
- Clearing Date, To. Set to yesterday’s date.
- Sequence Number 2. Set to 0.

If Sequence Number 1 in CA77 is greater than zero, more CQ77 queries must be done to retrieve data. CQ77 must be filled with the following parameters:

- Series, filled with series in CA77.
- Clearing Date, From. Filled with Clearing Date, From in CA77.

- Sequence Number 1. Filled with Sequence Number 1 in CA77.
- Clearing Date, To. Set to yesterday’s date.
- Sequence Number 2. Set to 0.

3.3.50 CQ122 [Position History QUERY]

3.3.50.1 Fingerprint

QUERY properties	
transaction type	CQ122
calling sequence	omniapi_query_ex
struct name	query_position_history
facility	EP5
partitioned	false
answers	CA122

ANSWER properties	
transaction type	CA122
struct name	answer_position_history
segmented	true

3.3.50.2 Related Messages

CQ123, CD54, CD55

3.3.50.3 Purpose

This query retrieves historical information and closed-out quantities on a particular position identified by the position account and instrument series.

3.3.50.4 Structure

The CQ122 QUERY has the following structure:

```
struct query_position_history {
   struct transaction_type
   struct series  // Named struct no: 50000
   struct account
   char[8] date_s  // Date
}
```

3.3.50.5 Usage and conditions

Series

- Series together with account identifies the position.
- Series must be completely filled and identify an existing series.

Account

- Account together with series identifies the position.
- Account must identify a specific account.
- Wildcards in account is not supported.

Date

must be prior to current Clearing date.

3.3.50.6 Answer Structure

The CA122 ANSWER has the following structure:

```
struct answer_position_history {
   struct transaction type
   struct partition low
   struct partition high
   struct series  // Named struct no: 50000
   struct account
   INT64 T nbr held q  // Held
   INT64 T nbr written q  // Written
   INT64 T qty closed out q  // Quantity, Closed out
   char[8] date s  // Date
}
```

3.3.51 CQ123 [Position Closeout Log QUERY]

3.3.51.1 Fingerprint

QUERY properties	
transaction type	CQ123
calling sequence	omniapi_query_ex
struct name	query_closeout_log
facility	EP5
partitioned	false
answers	CA123

ANSWER properties	
transaction type	CA123
struct name	answer_closeout_log
segmented	true

3.3.51.2 Related Messages

CQ122, CD54, CD55

3.3.51.3 Purpose

This query retrieves information on position closeout and position reinstatement requests.

3.3.51.4 Structure

The CQ123 QUERY has the following structure:

```
struct query_closeout_log {
   struct transaction_type
   struct series  // Named struct no: 50000
   UINT16_T segment_number_n  // Segment Number
   char[2] filler_2_s  // Filler
   struct account
   char[32] series_id_s  // Series, Identity
   char[8] from_date_s  // Date, From
   char[6] from_time_s  // Time, From
   char[8] to_date_s  // Date, To
   char[6] to_time_s  // Time, To
}
```

3.3.51.5 Usage and conditions

Series

should be filled with Country number, Market code and Instrument group.

Segment number

is one for the first query and then incremented.

Series, Identity

should contain an explicit series name or a series wildcard string.

Account

must identify a specific member. Wildcards is only supported in “Account identity” part pf Account

Date, From

is the time range start date for which position cloesout information is requested.

Date, To

is the time range stop date for which position cloesout information is requested.

Time, From

is the time range start time for which position cloesout information is requested.

Time, To

is the time range stop time for which position cloesout information is requested.

The range in time (and date) is expressed in the system time, which normally is identical to the local time at the exchange.

3.3.51.6 Answer Structure

The CA123 ANSWER has the following structure:

```
struct answer_closeout_log {
   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: 825] {
      struct trading_code
      struct series  // Named struct no: 50000
      struct account
      INT64_T closeout_qty_i  // Quantity, Close out
      char[8] date_s  // Date
      char[8] created_date_s  // Date, Created
      char[6] created_time_s  // Time, Created
      UINT8_T open_close_c  // Open or Closed
      UINT8_T state_c  // State
   }
}
```

3.4 Reports

3.4.1 LQ3 [List with Version QUERY]

3.4.1.1 Fingerprint

QUERY properties	
transaction type	LQ3
calling sequence	omniapi_query_ex
struct name	query_list_ver

QUERY properties	
facility	EP4
partitioned	false
answers	LA3

ANSWER properties	
transaction type	LA3
struct name	answer_list_ver
segmented	true

3.4.1.2 Purpose

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

3.4.1.3 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
}
```

3.4.1.4 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.

3.4.1.5 Answer Structure

The LA3 ANSWER has the following structure:

```
struct answer_list_ver {
   struct transaction_type
   struct series  // Named struct no: 50000
   INT32_T info_type_i  // Information Type
   UINT16_T segment_number_n  // Segment Number
   char[40] list_name_s  // Name, List
   char[3] report_version_s  // Report Version
   CHAR filler_1_s  // Filler
```

```
    char[8] file_type_s  // File Type
    UINT16_T items_n  // Items
    char[50000] text_buffer_s  // Text, Buffer
}
```

3.4.1.6 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.

3.4.2 LQ4 [Available Reports with Version QUERY]

3.4.2.1 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

3.4.2.2 Purpose

This transaction is used for querying for available report versions.

3.4.2.3 Structure

The LQ4 QUERY has the following structure:

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

3.4.2.4 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 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.

3.4.2.5 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
    }
}
```

3.4.2.6 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.

3.5 Miscellaneous

3.5.1 BI7 [Signal Information Ready BROADCAST]

3.5.1.1 Fingerprint

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

3.5.1.2 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.

3.5.1.3 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
}
```

3.5.1.4 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 type signals might also have been sent, e.g. BI7, type 2. After the BI7 type 3 signal has been sent, new trades via Dedicated Trade Information Broadcast and new deliveries via BD18 is sent followed by a BI7 type 1 signal and possibly other BI7 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 Num-

Information type	Interpretation	Comment
		ber, 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.
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. Information Type identifies the report. Series contains in this case Country Number and Market Code.

Information type	Interpretation	Comment
		Signals sent to indicate when specific reports are available depend on Exchange policy.

3.5.2 BI27 [Clearing message BROADCAST]

3.5.2.1 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

3.5.2.2 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.

3.5.2.3 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
   }
}
```

3.5.2.4 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.

3.5.3 BI71 [Set Commission Table BROADCAST]

3.5.3.1 Fingerprint

BROADCAST properties	
transaction type	BI71
calling sequence	omniapi_read_event_ext_ex or omniapi_read_event_block
struct name	set_commission_table
info type	general

3.5.3.2 Purpose

This broadcast makes it possible for applications to detect when the data in the commission table is changed. It is sent out every time the commission table is modified.

3.5.3.3 Structure

The BI71 BROADCAST has the following structure:

```
struct set_commission_table {
   struct broadcast_type
   struct party
}
```

3.5.3.4 Usage and Conditions

Party

indicates for which member the database table has been modified.

3.5.4 UQ9 [BI7 Signals Sent QUERY]

3.5.4.1 Fingerprint

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

QUERY properties	
answers	UA9

ANSWER properties	
transaction type	UA9
struct name	answer_bi7_signals_sent
segmented	true

3.5.4.2 Purpose

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

3.5.4.3 Structure

The UQ9 QUERY has the following structure:

```
struct query_bi7_signals_sent {
   struct transaction type
   struct search series
   UINT16_T segment_number_n  // Segment Number
   char[8] business date s  // Date, Business
   UINT16 T seq num srm n  // Sequence number for SRM
}
```

3.5.4.4 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
   }
}
```

3.5.5 UQ12 [Business Date QUERY]

3.5.5.1 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

3.5.5.2 Purpose

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

3.5.5.3 Structure

The UQ12 QUERY has the following structure:

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

3.5.5.4 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.

3.5.5.5 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
}
```

3.5.5.6 Answer, comments

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

3.5.6 UQ13 [BI27 Broadcasts Sent QUERY]

3.5.6.1 Fingerprint

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

ANSWER properties	
transaction type	UA13
struct name	answer_bi27_broadcasts_sent
segmented	true

3.5.6.2 Purpose

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

3.5.6.3 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
}
```

3.5.6.4 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
      }
   }
}
```

3.5.6.5 Answer, comments

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

3.6 Risk Management

3.6.1 RQ44 [Margin Underlying Real Time Price QUERY]

3.6.1.1 Fingerprint

QUERY properties	
transaction type	RQ44
calling sequence	omniapi_query_ex
struct name	query_realtime_ulg_price
facility	EP4
partitioned	false
answers	RA44

ANSWER properties	
transaction type	RA44
struct name	answer_realtime_ulg_price
segmented	true

3.6.1.2 Purpose

This query contains real time underlying prices.

3.6.1.3 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
}
```

3.6.1.4 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.

3.6.1.5 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
   }
}
```

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 {
      char[2] country_id_s  // Name, Country
      char[5] ex_customer_s  // Customer, Identity
      CHAR filler_1_s  // Filler
   }
   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
   }
   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
   }
   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
   }
   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
   }
   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
   }
   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
   }
   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
   }
   char[3] risk_currency_s  // Currency, Risk
   INT32_T rank_class_i  // Risk Ranking Class
   char[8] modified_date_s  // Date, Modified
   char[6] modified_time_s  // Time, Modified
   char[8] created_date_s  // Date, Created
   char[6] created_time_s  // Time, Created
   char[4] investor_type_s  // Investor Type
   char[4] nationality_s  // Nationality
   char[20] account_text_s  // Account Text
   char[34] ext_acc_id_s  // External Account ID
   char[15] ext_acc_controller_s  // External Account Controller
   char[12] ext_acc_registrar_s  // External Account Registrar
   char[16] org_number_s  // Organization number
   char[32] account_alias_s  // Account alias
   char[15] diary_number_s  // Diary Number
   char[12] acc_type_s  // Account Type
   char[12] fee_type_s  // Account Fee Type
   char[12] cust_bank_id_s  // Custodian Bank
   UINT8_T acc_state_c  // Account State
   UINT8_T read_access_c  // Read Access
   UINT8_T auto_net_c  // Auto Netting
   UINT8_T risk_cur_conv_c  // Risk, Currency Conversion
   UINT8_T risk_margin_net_c  // Risk, Margin Net
   UINT8_T acc_allow_nov_c  // Novation Allowed
   char[2] filler_2_s  // Filler
}
```

4.3 ANSWER_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
}
```

4.4 BROADCAST_HDR

```
struct broadcast_hdr {
   struct broadcast_type
   UINT16_T items_n  // Items
   UINT16_T size_n  // Size
}
```

4.5 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
}
```

4.6 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
}
```

4.7 CL_DELIVERY_API

```
struct cl_delivery_api {
   struct account
   struct delivery_account {
      char[2] country_id_s  // Name, Country
      char[5] ex_customer_s  // Customer, Identity
      char[10] account_id_s  // Account, Identity
      char[3] filler_3_s  // Filler
   }
   struct series
   struct deliv_base
   INT64_T deliv_base_quantity_q  // Quantity, Delivery Base
   INT64_T delivery_quantity_q  // Quantity, Delivery
   INT32_T delivery_number_i  // Delivery, Number
   INT32_T key_number_i  // Key Number
   INT32_T delivery_origin_i  // Delivery Origin
   INT32_T class_no_i  // Class Number
   INT32_T sequence_number_i  // Sequence Number
   INT32_T event_type_i  // Stimuli Event
   INT32_T original_delivery_number_i  // Original, Delivery Number
   INT32_T original_key_number_i  // Original, Key Number
   UINT32_T delivery_unit_u  // Delivery Unit
   UINT32_T delivery_properties_u  // Delivery Properties
   UINT32_T propagation_u  // Propagation
   char[8] settlement_date_s  // Date, Settlement
   char[8] date_s  // Date
   char[24] dvp_account_s  // DVP Account
   char[8] original_date_s  // Original Date
   char[32] passthrough_s  // Passthrough Information
   UINT8_T delivery_type_c  // Delivery, Type
   UINT8_T originator_type_c  // Originator Type
   UINT8_T delivery_state_c  // Delivery, State
   UINT8_T bought_or_sold_c  // Bought or Sold
   CHAR ext_trade_fee_type_c  // External Trade, Fee Type
   CHAR filler_1_s  // Filler
   char[2] giving_up_exchange_s  // Giving Up Exchange
   char[8] settlement_instr_date_s  // Date, Settlement instruction
}
```

4.8 CL_GIVE_UP_API

```
struct cl_give_up_api {
   struct series
   struct account
   struct party
   INT32_T sequence_number_i  // Sequence Number
   INT32_T gup_reason_i  // Give Up, Broadcast Reason
   INT32_T give_up_number_i  // Give Up, Number
   INT64_T trade_quantity_i  // Quantity, Trade
   INT32_T deal_price_i  // Price, Deal
   INT32_T trade_number_i  // Trade Number
   INT32_T commission_i  // Commission
   UINT8_T bought_or_sold_c  // Bought or Sold
   UINT8_T state_c  // State
   char[8] created_date_s  // Date, Created
   char[6] created_time_s  // Time, Created
   char[30] give_up_text_s  // Give Up, Free Text
   char[8] asof_date_s  // Date, As Of
   char[6] asof_time_s  // Time, As Of
   char[8] orig_clearing_date_s  // Clearing Date, Original
   UINT8_T old_trade_c  // Old Trade Indicator
   CHAR ext_trade_fee_type_c  // External Trade, Fee Type
   UINT8_T deal_source_c  // Deal Source
   UINT8_T reserved_prop_c  // Reserved Properties
   char[8] clearing_date_s  // Clearing Date
   UINT32_T ext_trade_number_u  // Trade Number, External
   UINT32_T orig_ext_trade_number_u  // Trade Number, Original External
   UINT8_T trade_venue_c  // Trade venue
   char[3] filler_3_s  // Filler
}
```

4.9 CL_TRADE_CHANGE_API

```
struct cl_trade_change_api {
   struct series
   INT32_T trade_number_i  // Trade Number
   INT32_T sequence_number_i  // Sequence Number
   UINT8_T trade_state_c  // Trade, State
   UINT8_T le_state_c  // Type, Legal Event
   UINT8_T give_up_state_c  // Give Up, State
   UINT8_T instance_c  // Instance, Number
   INT64_T rem_quantity_i  // Quantity, Remaining
```

```
      char[8] modified_date_s  // Date, Modified
      char[6] modified_time_s  // Time, Modified
      char[2] filler_2_s  // Filler
      UINT32_T big_attention_u  // Big Attention
   }
```

4.10 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
   }
```

4.11 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
   }
```

4.12 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
   }
```

4.13 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
   }
```

4.14 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
}
```

4.15 ITEM_HDR

```
struct item_hdr {
   UINT16_T items_n  // Items
   UINT16_T size_n  // Size
}
```

4.16 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
}
```

4.17 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
}
```

4.18 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
}
```

4.19 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
}
```

4.20 PARTITION_HIGH

```
struct partition_high {
   UINT8_T country_c  // Country Number
   UINT8_T market_c  // Market Code
   UINT8_T instrument_group_c  // Instrument Group
   UINT8_T modifier_c  // Modifier
   UINT16_T commodity_n  // Commodity Code
   UINT16_T expiration_date_n  // Date, Expiration
   INT32_T strike_price_i  // Strike Price
}
```

4.21 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
}
```

4.22 PARTY

```
struct party {
   char[2] country_id_s  // Name, Country
   char[5] ex_customer_s  // Customer, Identity
   CHAR filler_1_s  // Filler
}
```

4.23 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
}
```

4.24 POS_INFO_UPDATE_API

```
struct pos_info_update_api {
   struct series
   struct account
   INT64_T deny_exercise_q  // Deny Exercise
   INT64_T qty_closed_out_q  // Quantity, Closed out
   UINT32_T quantity_cover_u  // Quantity Cover
   char[8] modified_date_s  // Date, Modified
   char[6] modified_time_s  // Time, Modified
   UINT8_T reserved_prop_c  // Reserved Properties
   CHAR filler_1_s  // Filler
}
```

4.25 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
}
```

4.26 SEARCH_SERIES

```
struct search_series {
   UINT8_T country_c  // Country Number
   UINT8_T market_c  // Market Code
   UINT8_T instrument_group_c  // Instrument Group
   UINT8_T modifier_c  // Modifier
   UINT16_T commodity_n  // Commodity Code
   UINT16_T expiration_date_n  // Date, Expiration
   INT32_T strike_price_i  // Strike Price
}
```

4.27 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
}
```

4.28 SUB_ITEM_HDR

```
struct sub_item_hdr {
   UINT16_T named_struct_n  // Named Struct, Number
   UINT16_T size_n  // Size
}
```

4.29 TICK_SIZE

```
struct tick_size {
   INT32_T step_size_i  // Tick Size
   INT32_T lower_limit_i  // Premium/Price, Low Limit
   INT32_T upper_limit_i  // Premium/Price, High Limit
}
```

4.30 TIME_SPEC

```
struct time_spec {
   UINT32_T tv_sec  // Time in seconds
   INT32_T tv_nsec  // Time in nanoseconds
}
```

4.31 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
}
```

4.32 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
}
```

4.33 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
}
```

4.34 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
}
```

5 Named Structs Involved in VIMs

Named structs used in the variable information messages (VIM) included in this message reference are listed here in numerical order.

5.1 CL_TRADE_BASE_API (3)

```
struct cl_trade_base_api {
   struct trading_code
   struct series  // Named struct no: 50000
   struct give_up_member  // Named struct no: 50002
   QUAD_WORD order_number_u  // Order Number
   INT32_T sequence_number_i  // Sequence Number
   INT32_T trade_number_i  // Trade Number
   INT32_T deal_price_i  // Price, Deal
   INT64_T trade_quantity_i  // Quantity, Trade
   struct account
   char[15] customer_info_s  // Customer, Information
   UINT8_T bought_or_sold_c  // Bought or Sold
   UINT8_T deal_source_c  // Deal Source
   UINT8_T open_close_req_c  // Open  Close Request
   UINT8_T trade_type_c  // Type, Trade
   UINT8_T le_state_c  // Type, Legal Event
   struct user_code
   char[8] created_date_s  // Date, Created
   char[6] created_time_s  // Time, Created
   char[8] asof_date_s  // Date, As Of
   char[6] asof_time_s  // Time, As Of
   char[8] modified_date_s  // Date, Modified
   char[6] modified_time_s  // Time, Modified
   UINT8_T trade_state_c  // Trade, State
   UINT8_T attention_c  // Attention
   INT32_T deal_number_i  // Deal Number
   UINT32_T global_deal_no_u  // Global Deal Number
   INT32_T orig_trade_number_i  // Trade Number, Original
   struct orig_series
   CHAR[32] exchange_info_s  // Exchange, Information
   UINT32_T big_attention_u  // Big Attention
   char[8] clearing_date_s  // Clearing Date
   struct execution_timestamp  // Of type: TIME_SPEC
   UINT8_T trade_venue_c  // Trade venue
   UINT8_T instance_c  // Instance, Number
   UINT16_T exch_order_type_n  // Order Type, Exchange
   struct party
   UINT16_T trade_rep_code_n  // Trade Report Code
   char[2] filler_2_s  // Filler
   struct match_id
}
```

5.2 CL_TRADE_SECUR_PART (20)

```
struct cl_trade_secur_part {
   struct countersign_code
   struct new_series
   struct party
   struct pos_account
   struct combo_series
   INT64_T nbr_held_q  // Held
   INT64_T nbr_written_q  // Written
   INT64_T total_held_q  // Held, Total
   INT64_T total_written_q  // Written Total
   INT32_T ext_seq_nbr_i  // External Clearinghouse, Sequence Number
   INT32_T ext_status_i  // Return Status
   INT64_T rem_quantity_i  // Quantity, Remaining
   INT64_T quantity_i  // Quantity
   UINT32_T ext_trade_number_u  // Trade Number, External
   UINT32_T orig_ext_trade_number_u  // Trade Number, Original External
   INT32_T residual_i  // Residual
   INT32_T give_up_number_i  // Give Up, Number
   INT32_T commission_i  // Commission
   INT32_T combo_deal_price_i  // Combo deal price
   char[8] clearing_date_s  // Clearing Date
   char[32] passthrough_s  // Passthrough Information
   char[10] ex_client_s  // Client
   CHAR ext_trade_fee_type_c  // External Trade, Fee Type
   UINT8_T give_up_state_c  // Give Up, State
   char[2] reserved_2_s  // Reserved
   UINT8_T orig_trade_type_c  // Trade Type, Original
   UINT8_T open_close_c  // Open or Closed
   CHAR reserved_1_c  // Reserved
   UINT8_T account_type_c  // Account Type
   UINT8_T instigant_c  // Instigant
   UINT8_T cab_price_ind_c  // Cabinet Price Indicator
}
```

5.3 NS_DELTA_HEADER (37001)

```
struct ns_delta_header {
   INT64_T download_ref_number_q  // Download Reference Number
   struct full_answer_timestamp  // Of type: TIME_SPEC
   UINT8_T full_answer_c  // Full Answer
   char[3] filler_3_s  // Filler
}
```

5.4 NS_REMOVE (37002)

```
struct ns_remove {
   struct series  // Named struct no: 50000
}
```

5.5 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
}
```

5.6 NS_PRICE_TICK (37102)

```
struct ns_price_tick {
   struct tick_size
   UINT16_T dec_in_premium_n  // Decimals, Premium
   CHAR is_fractions_c  // Fraction, Premium
   UINT8_T price_format_c  // Premium/Price Format
}
```

5.7 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
}
```

5.8 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
}
```

5.9 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
}
```

5.10 NS_PRICE_TICK_CORR (37113)

```
struct ns_price_tick_corr {
   struct tick_size
   UINT16_T dec_in_premium_n  // Decimals, Premium
   char[2] filler_2_s  // Filler
}
```

5.11 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
}
```

5.12 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
}
```

5.13 NS_FIXED_INCOME (37202)

```
struct ns_fixed_income {
   INT64_T nominal_value_q  // Nominal Value
   UINT32_T coupon_interest_i  // Coupon Interest
   UINT16_T dec_in_nominal_n  // Decimals, Nominal
   UINT16_T coupon_settlement_days_n  // Coupon Settlement Days
   UINT16_T coupon_frequency_n  // Coupon Frequency
   UINT16_T rate_determ_days_n  // Rate Determination Days
   char[8] date_release_s  // Date, Issue
   char[8] date_termination_s  // Date, Maturity
   char[8] date_dated_s  // Date, Dated
   char[8] date_proceed_s  // Date, Proceed
   UINT8_T fixed_income_type_c  // Fixed Income Type
   UINT8_T day_calc_rule_c  // Day Calculation Rule
   char[2] filler_2_s  // Filler
}
```

5.14 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
}
```

5.15 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
}
```

5.16 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
}
```

5.17 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
}
```

5.18 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
}
```

5.19 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
}
```

5.20 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
}
```

5.21 NS_CONVERTIBLES (37213)

```
struct ns_convertibles {
    char[8] date_convert_from_s  // Date, Convert From
    char[8] date_convert_through_s  // Date, Convert Through
}
```

5.22 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
}
```

5.23 NS_INST_SERIES_BASIC (37301)

```
struct ns_inst_series_basic {
   struct series  // Named struct no: 50000
   UINT16_T step_size_multiple_n  // Tick Size, Multiple
   char[32] ins_id_s  // Series, Identity
   char[32] long_ins_id_s  // Series Name, Long
   char[8] date_last_trading_s  // Date, Last Trading
   char[6] time_last_trading_s  // Time, Last Trading
   char[8] date_first_trading_s  // Date, First Trading
   char[6] time_first_trading_s  // Time, First Trading
   UINT8_T series_status_c  // Series, Status
   UINT8_T suspended_c  // Suspended
   UINT8_T traded_in_click_c  // Traded in GENIUM
   UINT8_T db_operation_c  // Operation
   UINT8_T trade_reporting_only_c  // Only trade reports allowed
   CHAR filler_1_s  // Filler
}
```

5.24 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
}
```

5.25 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
}
```

5.26 NS_INST_SERIES_REPO (37304)

```
struct ns_inst_series_repo {
   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
}
```

5.27 NS_INST_SERIES_BO (37306)

```
struct ns_inst_series_bo {
   char[12] isin_code_old_s  // ISIN Code, Old Series
   UINT8_T tm_template_c  // Template Series
   UINT8_T tm_series_c  // Tailor Made Series
   UINT8_T accept_collateral_c  // Accepted as Collateral
}
```

5.28 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
}
```

5.29 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
}
```

5.30 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
}
```

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
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
BD40	Dedicated auxiliary position info update information	Standard	dedicated	4
BI1	Resumption and Suspension of Trading	Standard	general	1
BI7	Signal Information Ready	Standard	general	1
BI27	Clearing message	Standard	general	1
BI41	Instrument Status Information	Standard	general	1
BI71	Set Commission Table	Standard	general	1
BU2	Series Update	Standard	general	1
BU4	Underlying Update	Standard	general	1
BU9	Series Backoffice Update	Standard	general	1
BU10	Instrument Class Update	Standard	general	1
BU12	Account Type Update	Standard	general	1
BU13	Account Fee Type Update	Standard	general	1
BU18	Non-Trading Days Update	Standard	general	1
BU19	Underlying Backoffice Update	Standard	general	1
BU20	Instrument Class Backoffice Update	Standard	general	1
BU44	Legal Account Instrument Update	Standard	general	1

Transaction Type	Name	Design	Information Type	Information Type Value
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

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.

<table>
<tr><th colspan="2">abbrev_name_s (Abbreviation Name)</th></tr>
<tr><td>Datatype</td><td>char[32]</td></tr>
<tr><td>Description</td><td>Specifies the abbreviation name for the underlying.</td></tr>
<tr><th colspan="2">abbr_name_s (Abbreviated Name)</th></tr>
<tr><td>Datatype</td><td>char[8]</td></tr>
<tr><td>Description</td><td>Abbreviated name</td></tr>
<tr><th colspan="2">accept_collateral_c (Accepted as Collateral)</th></tr>
<tr><td>Datatype</td><td>UINT8_T</td></tr>
<tr><td>Description</td><td>Accepted as collateral?.</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>
<tr><td>Default</td><td>0</td></tr>
</table></td></tr>
<tr><th colspan="2">account_alias_s (Account alias)</th></tr>
<tr><td>Datatype</td><td>char[32]</td></tr>
<tr><td>Description</td><td>Defines the account name alias for an account.</td></tr>
<tr><th colspan="2">account_id_s (Account, Identity)</th></tr>
<tr><td>Datatype</td><td>char[10]</td></tr>
<tr><td>Description</td><td>The account identification part of an ACCOUNT structure; the part after the member identification.</td></tr>
<tr><th colspan="2">account_text_s (Account Text)</th></tr>
<tr><td>Datatype</td><td>char[20]</td></tr>
<tr><td>Description</td><td>Free text, 20 characters</td></tr>
<tr><th colspan="2">account_type_c (Account Type)</th></tr>
<tr><td>Datatype</td><td>UINT8_T</td></tr>
<tr><td>Description</td><td>The account type for a trade.</td></tr>
<tr><td>Value Set</td><td><table>
<tr><th>name</th><th>value</th></tr>
<tr><td>Customer</td><td>1</td></tr>
<tr><td>Firm</td><td>2</td></tr>
<tr><td>Market Maker</td><td>3</td></tr>
</table></td></tr>
</table>

<table>
<tr><td colspan="2">account_type_s (Account Type)</td></tr>
<tr><td>Datatype</td><td>char[12]</td></tr>
<tr><td>Description</td><td>Tells what type of account it is.</td></tr>
<tr><td colspan="2">accr_intr_round_u (Accrued Interest Rounding)</td></tr>
<tr><td>Datatype</td><td>UINT32_T</td></tr>
<tr><td>Description</td><td>Accrued Interest Rounding</td></tr>
<tr><td colspan="2">accr_intr_ud_c (Accrued Interest Up or Down)</td></tr>
<tr><td>Datatype</td><td>UINT8_T</td></tr>
<tr><td>Description</td><td>Accrued Interest Up/Down</td></tr>
<tr><td>Value Set</td><td><table>
<tr><th>name</th><th>value</th></tr>
<tr><td>Up</td><td>1</td></tr>
<tr><td>Down</td><td>2</td></tr>
</table></td></tr>
<tr><td colspan="2">acc_allow_nov_c (Novation Allowed)</td></tr>
<tr><td>Datatype</td><td>UINT8_T</td></tr>
<tr><td>Description</td><td>Defines if novation is allowed on an account or not.None indicates that novation is not applicable on the account.</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>Yes</td><td>1</td></tr>
<tr><td>No</td><td>2</td></tr>
</table></td></tr>
<tr><td colspan="2">acc_state_c (Account State)</td></tr>
<tr><td>Datatype</td><td>UINT8_T</td></tr>
<tr><td>Description</td><td>Defines the state that the account is in.</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>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></td></tr>
<tr><td colspan="2">acc_type_s (Account Type)</td></tr>
</table>

<table>
<tr><td>Datatype</td><td>char[12]</td></tr>
<tr><td>Description</td><td>Tells what type of account it is</td></tr>
<tr><td colspan="2">actual_start_date_s (Actual Start Date)</td></tr>
<tr><td>Datatype</td><td>char[8]</td></tr>
<tr><td>Description</td><td>Defines actual start date. Distributed in UTC together with Actual Start Time. Format: YYYYMMDD.</td></tr>
<tr><td colspan="2">actual_start_time_s (Actual Start Time)</td></tr>
<tr><td>Datatype</td><td>char[6]</td></tr>
<tr><td>Description</td><td>Defines actual start time. Distributed in UTC together with Actual Start Date. Format: HHMMSS.</td></tr>
<tr><td colspan="2">adjusted_c (Adjusted Series)</td></tr>
<tr><td>Datatype</td><td>UINT8_T</td></tr>
<tr><td>Description</td><td>Is the actual adjustment containing new adjusted series?</td></tr>
<tr><td>Value Set</td><td><table><tr><th>value</th><th>description</th></tr><tr><td>1</td><td>Yes</td></tr><tr><td>2</td><td>No</td></tr></table></td></tr>
<tr><td colspan="2">adjust_ident_n (Adjustment Identifier)</td></tr>
<tr><td>Datatype</td><td>UINT16_T</td></tr>
<tr><td>Description</td><td>A number that uniquely identifies an adjustment for series with the same adjustment conditions.</td></tr>
<tr><td colspan="2">allow_interbank_c (Allow interbank)</td></tr>
<tr><td>Datatype</td><td>UINT8_T</td></tr>
<tr><td>Description</td><td>The trade report type is allowed to report between different participant.</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">allow_non_std_settlement_c (Allow non standard settlement)</td></tr>
<tr><td>Datatype</td><td>UINT8_T</td></tr>
<tr><td>Description</td><td>Allow a non standard settlement date in the trade report.</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">allow_within_participant_c (Allow within participant)</td></tr>
<tr><td>Datatype</td><td>UINT8_T</td></tr>
<tr><td>Description</td><td>The trade report type is allowed to report within the same participant.</td></tr>
</table>

<table>
<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">arranger_country_id_s (Arranger, Country)</td></tr>
<tr><td>Datatype</td><td>char[2]</td></tr>
<tr><td>Description</td><td>The exchange identity that together with Arranger, Customer represents the arranger.</td></tr>
<tr><td colspan="2">arranger_ex_customer_s (Arranger, Customer)</td></tr>
<tr><td>Datatype</td><td>char[5]</td></tr>
<tr><td>Description</td><td>This field together with Arranger, Country, identifies the member/participant that represents the arranger.</td></tr>
<tr><td colspan="2">ascii_bin_c (ASCII or Binary)</td></tr>
<tr><td>Datatype</td><td>UINT8_T</td></tr>
<tr><td>Description</td><td>ASCII or Binary?</td></tr>
<tr><td>Value Set</td><td><table><tr><th>value</th><th>description</th></tr><tr><td>1</td><td>ASCII</td></tr><tr><td>2</td><td>Binary</td></tr></table></td></tr>
<tr><td colspan="2">ask_price_i (Ask Price)</td></tr>
<tr><td>Datatype</td><td>UINT32_T</td></tr>
<tr><td>Description</td><td>Price for ask requests (orders selling the given Series). Statistics information.</td></tr>
<tr><td colspan="2">ask_theo_c (Ask, Theoretical Mark)</td></tr>
<tr><td>Datatype</td><td>UINT8_T</td></tr>
<tr><td>Description</td><td>The field indicates the origin of the price:</td></tr>
<tr><td>Value Set</td><td><table><tr><th>value</th><th>description</th></tr><tr><td>0</td><td>Missing</td></tr><tr><td>1</td><td>Theoretically calculated</td></tr><tr><td>2</td><td>From the Orderbook</td></tr><tr><td>3</td><td>Manually updated</td></tr><tr><td>4</td><td>Artificial</td></tr></table></td></tr>
<tr><td colspan="2">asof_date_s (Date, As Of)</td></tr>
<tr><td>Datatype</td><td>char[8]</td></tr>
<tr><td>Description</td><td>The date an object is valid for. Format: YYYYMMDD.</td></tr>
<tr><td colspan="2">asof_time_s (Time, As Of)</td></tr>
<tr><td>Datatype</td><td>char[6]</td></tr>
<tr><td>Description</td><td>The time an object is valid for. Format: HHMMSS.</td></tr>
</table>

<table>
<tr><td colspan="2">atr_id_s (Account Type Rule)</td></tr>
<tr><td>Datatype</td><td>char[12]</td></tr>
<tr><td>Description</td><td>The identity of Account Type Rule.</td></tr>
<tr><td colspan="2">attention_c (Attention)</td></tr>
<tr><td>Datatype</td><td>UINT8_T</td></tr>
<tr><td>Description</td><td>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.</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>
<tr><td>Value Set</td><td><table>
<tr><th>value</th><th>description</th></tr>
<tr><td>1</td><td>Yes</td></tr>
<tr><td>2</td><td>No</td></tr>
</table></td></tr>
<tr><td colspan="2">average_period_c (Average Period)</td></tr>
<tr><td>Datatype</td><td>UINT8_T</td></tr>
<tr><td>Description</td><td>Not applicable.</td></tr>
</table>

<table>
<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>1</td><td>Quarterly</td></tr><tr><td>2</td><td>Half Year</td></tr><tr><td>3</td><td>Year</td></tr></table></td></tr>
<tr><td colspan="2">base_cur_s (Currency, Trading)</td></tr>
<tr><td>Datatype</td><td>char[3]</td></tr>
<tr><td>Description</td><td>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.</td></tr>
<tr><td colspan="2">base_price_u (Base Price)</td></tr>
<tr><td>Datatype</td><td>UINT32_T</td></tr>
<tr><td>Description</td><td>Defines the base price for the derived from with three implicit decimals.</td></tr>
<tr><td colspan="2">benchmark_bond_code_s (Benchmark Bond Code)</td></tr>
<tr><td>Datatype</td><td>char[12]</td></tr>
<tr><td>Description</td><td>Defines the benchmark bond code for the underlying.</td></tr>
<tr><td colspan="2">bic_code_s (BIC Code)</td></tr>
<tr><td>Datatype</td><td>char[15]</td></tr>
<tr><td>Description</td><td>The BIC consists of four parts and is usually written as BANKCCLLMAR. The parts are interpreted as explained in the table:</td></tr>
<tr><td>Value Set</td><td><table><tr><th>value</th><th>description</th></tr><tr><td>BANK</td><td>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]</td></tr><tr><td>CC</td><td>CC is the ISO country code. The country code identifies the country in which the financial institution is located. [2 bytes]</td></tr><tr><td>LL</td><td>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]</td></tr><tr><td>MAR</td><td>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]</td></tr></table></td></tr>
<tr><td colspan="2">bid_or_ask_c (Bid or Ask)</td></tr>
<tr><td>Datatype</td><td>UINT8_T</td></tr>
<tr><td>Description</td><td>Specifies what quotation side is requested.</td></tr>
</table>

<table>
<tr><td>Value Set</td><td><table><tr><th>value</th><th>description</th></tr><tr><td>0</td><td>Bid and Ask</td></tr><tr><td>1</td><td>Bid</td></tr><tr><td>2</td><td>Ask</td></tr></table></td></tr>
<tr><td colspan="2">bid_price_i (Bid Price)</td></tr>
<tr><td>Datatype</td><td>UINT32_T</td></tr>
<tr><td>Description</td><td>Price for bid requests (orders buying the given Series). Statistics information.</td></tr>
<tr><td colspan="2">bid_theo_c (Bid, Theoretical Mark)</td></tr>
<tr><td>Datatype</td><td>UINT8_T</td></tr>
<tr><td>Description</td><td>The field indicates the origin of the price:</td></tr>
<tr><td>Value Set</td><td><table><tr><th>value</th><th>description</th></tr><tr><td>0</td><td>Missing.</td></tr><tr><td>1</td><td>Theoretically calculated.</td></tr><tr><td>2</td><td>From the Orderbook.</td></tr><tr><td>3</td><td>Manually updated.</td></tr><tr><td>4</td><td>Artificial.</td></tr></table></td></tr>
<tr><td colspan="2">big_attention_u (Big Attention)</td></tr>
<tr><td>Datatype</td><td>UINT32_T</td></tr>
<tr><td>Description</td><td>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.</td></tr>
<tr><td>Value Set</td><td><table><tr><th>name</th><th>value</th><th>description</th></tr><tr><td>resent</td><td>1</td><td>Resent (bit 0)
The trade might have been subject to a retransition from the matching system to deal capture.</td></tr><tr><td>error_log</td><td>2</td><td>Error Log (bit 1)
The trade has an entry in the error log, retrievable with CQ22 with error identity as trade number.</td></tr><tr><td>date_phase</td><td>4</td><td>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.</td></tr></table></td></tr>
</table>

name	value	description
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 already 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.
excluded_from_stat	536870912	Excluded from trade statistics (bit 29)

<table>
<tr><td></td><td><table><tr><th>name</th><th>value</th><th>description</th></tr><tr><td></td><td></td><td>Trade belongs to a deal that has been excluded from trade statistics.</td></tr></table></td></tr>
<tr><td colspan="2">binary_variant_c (Option, Binary Variant)</td></tr>
<tr><td>Datatype</td><td>UINT8_T</td></tr>
<tr><td>Description</td><td>Defines the Option Binary Variants.</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>1</td><td>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.</td></tr><tr><td>2</td><td>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.</td></tr></table></td></tr>
<tr><td colspan="2">block_n (Block Size)</td></tr>
<tr><td>Datatype</td><td>UINT32_T</td></tr>
<tr><td>Description</td><td>Minimum number of units (options, futures, forwards and so on) in an order transaction.</td></tr>
<tr><td colspan="2">boolean (BOOLEAN)</td></tr>
<tr><td>Datatype</td><td>CHAR</td></tr>
<tr><td>Description</td><td>Intermediate field.</td></tr>
<tr><td colspan="2">bought_or_sold_c (Bought or Sold)</td></tr>
<tr><td>Datatype</td><td>UINT8_T</td></tr>
<tr><td>Description</td><td>Defines if the item or amount in question is bought or sold.</td></tr>
<tr><td>Value Set</td><td><table><tr><th>value</th><th>description</th></tr><tr><td>1</td><td>Bought</td></tr><tr><td>2</td><td>Sold</td></tr></table></td></tr>
<tr><td colspan="2">broadcast_number_n (Broadcast Number)</td></tr>
<tr><td>Datatype</td><td>UINT16_T</td></tr>
<tr><td>Description</td><td>A number used to distinguish between different broadcasts.</td></tr>
<tr><td colspan="2">business_date_s (Date, Business)</td></tr>
<tr><td>Datatype</td><td>char[8]</td></tr>
<tr><td>Description</td><td>Date in ASCII. Format: YYYYMMDD</td></tr>
</table>

<table>
<tr><td colspan="2">business_day_conv_c (BUSINESS_DAY_CONV_C)</td></tr>
<tr><td>Datatype</td><td>UINT8_T</td></tr>
<tr><td>Description</td><td>Used to find out the nearest business date
to calculated end date of a period.</td></tr>
<tr><td>Value Set</td><td><table><tr><th>name</th><th>value</th></tr><tr><td>Following</td><td>1</td></tr><tr><td>Modified following</td><td>2</td></tr><tr><td>Preceding</td><td>3</td></tr></table></td></tr>
<tr><td colspan="2">buy_or_sell_c (Buy or Sell)</td></tr>
<tr><td>Datatype</td><td>CHAR</td></tr>
<tr><td>Description</td><td>Buy or sell?</td></tr>
<tr><td>Value Set</td><td><table><tr><th>value</th><th>description</th></tr><tr><td>B</td><td>Buy</td></tr><tr><td>S</td><td>Sell</td></tr><tr><td>N</td><td>Not Applicable</td></tr></table></td></tr>
<tr><td colspan="2">buy_sell_back_c (Buy Sell Back)</td></tr>
<tr><td>Datatype</td><td>UINT8_T</td></tr>
<tr><td>Description</td><td>Sets if the REPO is a buy sell back or not.</td></tr>
<tr><td>Value Set</td><td><table><tr><th>value</th><th>description</th></tr><tr><td>1</td><td>Yes</td></tr><tr><td>2</td><td>No</td></tr></table></td></tr>
<tr><td colspan="2">cabinet_format_c (Cabinet Format)</td></tr>
<tr><td>Datatype</td><td>UINT8_T</td></tr>
<tr><td>Description</td><td>Not applicable.</td></tr>
<tr><td colspan="2">cab_price_ind_c (Cabinet Price Indicator)</td></tr>
<tr><td>Datatype</td><td>UINT8_T</td></tr>
<tr><td>Description</td><td>Specifies whether the price in a trade is a cabinet price or not.</td></tr>
<tr><td>Value Set</td><td><table><tr><th>value</th><th>description</th></tr><tr><td>1</td><td>Yes</td></tr><tr><td>2</td><td>No</td></tr></table></td></tr>
<tr><td colspan="2">cadj_trade_price_c (Cadj. Trade Price)</td></tr>
<tr><td>Datatype</td><td>UINT8_T</td></tr>
<tr><td>Description</td><td>Specifies if trade price is adjusted.</td></tr>
</table>

<table>
<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">calculation_conv_c (Calculation Convention)</td></tr>
<tr><td>Datatype</td><td>UINT8_T</td></tr>
<tr><td>Description</td><td>Calculation Convention</td></tr>
<tr><td>Value Set</td><td><table><tr><th>name</th><th>value</th></tr><tr><td>Compound</td><td>1</td></tr><tr><td>CompoundSimple</td><td>2</td></tr><tr><td>Simple_MM</td><td>3</td></tr><tr><td>Discount</td><td>4</td></tr><tr><td>US Treasury</td><td>5</td></tr><tr><td>Proceed</td><td>6</td></tr></table></td></tr>
<tr><td colspan="2">cbo_trade_report_c (Combo Trade Report)</td></tr>
<tr><td>Datatype</td><td>UINT8_T</td></tr>
<tr><td>Description</td><td>Describes if the Trade Report Type is used to do a combo trade report.</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">central_module_c (Central Module)</td></tr>
<tr><td>Datatype</td><td>CHAR</td></tr>
<tr><td>Description</td><td>Denotes essentially what subsystem is associated with the message. ISO Latin-1 representation is used.
Central module:</td></tr>
<tr><td>Value Set</td><td><table><tr><th>value</th><th>description</th></tr><tr><td>M</td><td>Market Place (MP/IMP)</td></tr><tr><td>C</td><td>Clearing (CL)</td></tr><tr><td>I</td><td>Information (IN)</td></tr><tr><td>S</td><td>Settlement (SE)</td></tr><tr><td>D</td><td>Common Database (CDB)</td></tr><tr><td>O</td><td>Operation (OP)</td></tr><tr><td>L</td><td>List Module (LM)</td></tr><tr><td>V</td><td>Settlement and Risk</td></tr><tr><td>R</td><td>Risk Valuation (RIVA)</td></tr></table></td></tr>
</table>

<table>
<tr><td></td><td><table><tr><th>value</th><th>description</th></tr><tr><td>U</td><td>Supervision (SU)</td></tr><tr><td>X</td><td>Deal Capture (DC)</td></tr></table></td></tr>
<tr><td colspan="2">chg_type_n (Change Type)</td></tr>
<tr><td>Datatype</td><td>UINT16_T</td></tr>
<tr><td>Description</td><td>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.</td></tr>
<tr><td>Value Set</td><td><table><tr><th>name</th><th>value</th><th>description</th></tr><tr><td>add</td><td>1</td><td>Addition
The item is added.</td></tr><tr><td>delete</td><td>2</td><td>Deletion
The item is deleted.</td></tr><tr><td>change</td><td>3</td><td>Modification
The item is modified. Examples of modifications would be delistings and change of last trading time.</td></tr></table></td></tr>
<tr><td colspan="2">class_no_i (Class Number)</td></tr>
<tr><td>Datatype</td><td>INT32_T</td></tr>
<tr><td>Description</td><td>Defines the type of settlement.</td></tr>
<tr><td>Value Set</td><td><table><tr><th>name</th><th>value</th><th>description</th></tr><tr><td></td><td>1</td><td>Marketplace fixed fee</td></tr><tr><td></td><td>2</td><td>Clearing variable fee</td></tr><tr><td></td><td>3</td><td>Tax</td></tr><tr><td></td><td>4</td><td>Rebate</td></tr><tr><td></td><td>5</td><td>Settlement
Premium, MTM, etc.</td></tr><tr><td>Settlement_dvp</td><td>6</td><td>Delivery versus payment</td></tr><tr><td>New_contract</td><td>7</td><td>Create a new trade</td></tr><tr><td>Settlement_odvp</td><td>8</td><td>The other qty and base</td></tr><tr><td></td><td>9</td><td>Internal information, API application should ignore this.</td></tr><tr><td></td><td>10</td><td>Variation margin</td></tr><tr><td>Commission</td><td>11</td><td>Commission</td></tr><tr><td>Settlement_intraday_collect</td><td>12</td><td>Intraday settlement collect</td></tr></table></td></tr>
</table>

<table>
<tr><td></td><td>
<table>
<tr><th>name</th><th>value</th><th>description</th></tr>
<tr><td>Accrued_interest</td><td>13</td><td>The interest accrued on cash instruments.</td></tr>
<tr><td>Settlement_dvp_cvr</td><td>16</td><td>Quantity of underlying used as cover to be delivered</td></tr>
<tr><td>Settlement_odvp_cvr</td><td>18</td><td>Payment for delivery of cover quantity</td></tr>
<tr><td></td><td>20</td><td>Rounding</td></tr>
<tr><td>Balance_adjustment</td><td>21</td><td>Balance adjustment</td></tr>
<tr><td></td><td>23</td><td>Fee 3</td></tr>
<tr><td></td><td>24</td><td>Fee 4</td></tr>
<tr><td></td><td>25</td><td>Fee 5</td></tr>
<tr><td></td><td>26</td><td>Fee 6</td></tr>
<tr><td></td><td>27</td><td>Fee 7</td></tr>
<tr><td></td><td>28</td><td>Fee 8</td></tr>
<tr><td></td><td>29</td><td>Fee 9</td></tr>
<tr><td></td><td>30</td><td>Fair value</td></tr>
</table>
</td></tr>
<tr><td colspan="2">clean_pr_round_u (Clean Price Rounding)</td></tr>
<tr><td>Datatype</td><td>UINT32_T</td></tr>
<tr><td>Description</td><td>Clean Price Rounding</td></tr>
<tr><td colspan="2">clean_pr_ud_c (Clean Price Up or Down)</td></tr>
<tr><td>Datatype</td><td>UINT8_T</td></tr>
<tr><td>Description</td><td>Clean Price Up/Down</td></tr>
<tr><td>Value Set</td><td>
<table>
<tr><th>name</th><th>value</th></tr>
<tr><td>Up</td><td>1</td></tr>
<tr><td>Down</td><td>2</td></tr>
</table>
</td></tr>
<tr><td colspan="2">cleared_dec_in_qty_n (Decimals, Quantity)</td></tr>
<tr><td>Datatype</td><td>UINT16_T</td></tr>
<tr><td>Description</td><td>Defines decimals in quantity in clearing related quantities.</td></tr>
<tr><td colspan="2">clearing_date_s (Clearing Date)</td></tr>
<tr><td>Datatype</td><td>char[8]</td></tr>
<tr><td>Description</td><td>Date in ASCII for clearing trade, format is YYYYMMDD.</td></tr>
<tr><td colspan="2">closed_for_clearing_c (Closed, clearing)</td></tr>
<tr><td>Datatype</td><td>UINT8_T</td></tr>
<tr><td>Description</td><td>Specifies if the date is closed for clearing.</td></tr>
</table>

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	
collateral_type_c (Collateral types)		
Datatype	UINT8_T	
Description	Defines the type of collateral.	
Value Set	**name**	**value**
	Cash Collateral	1
	Guarantee	2
	Member Deposit	3
	Certificate	4
	Fixed Income	5
	Equity	6
combo_deal_price_i (Combo deal price)		
Datatype	INT32_T	

<table>
<tr><td>Description</td><td>Combo deal price.</td></tr>
<tr><td colspan="2">commission_i (Commission)</td></tr>
<tr><td>Datatype</td><td>INT32_T</td></tr>
<tr><td>Description</td><td>The commission to pay for the operation.</td></tr>
<tr><td colspan="2">commodity_n (Commodity Code)</td></tr>
<tr><td>Datatype</td><td>UINT16_T</td></tr>
<tr><td>Description</td><td>Underlying definitions are defined by each exchange. Commodity Code is a part of the Series definition.</td></tr>
<tr><td colspan="2">com_id (COM_ID)</td></tr>
<tr><td>Datatype</td><td>char[6]</td></tr>
<tr><td>Description</td><td>Intermediate field.</td></tr>
<tr><td colspan="2">com_id_s (Underlying Identity)</td></tr>
<tr><td>Datatype</td><td>char[6]</td></tr>
<tr><td>Description</td><td>The ASCII representation of the underlying.</td></tr>
<tr><td colspan="2">condition_s (Trade Report Description)</td></tr>
<tr><td>Datatype</td><td>char[32]</td></tr>
<tr><td>Description</td><td>The description of the trade report type.</td></tr>
<tr><td colspan="2">confirm_reject_c (Confirm or Reject)</td></tr>
<tr><td>Datatype</td><td>UINT8_T</td></tr>
<tr><td>Description</td><td>The field states whether a holding item should be confirmed or rejected.</td></tr>
<tr><td>Value Set</td><td><table>
<tr><th>name</th><th>value</th></tr>
<tr><td>Rejected</td><td>0</td></tr>
<tr><td>Confirmed</td><td>1</td></tr>
</table></td></tr>
<tr><td colspan="2">contracts_modifier_c (Modifier, Number of Contracts)</td></tr>
<tr><td>Datatype</td><td>UINT8_T</td></tr>
<tr><td>Description</td><td>The modifier is used to recalculate the item after an underlying adjustment. The field is stored with 3 implicit decimals.</td></tr>
<tr><td>Value Set</td><td><table>
<tr><th>value</th><th>description</th></tr>
<tr><td>1</td><td>Modifier is added to the item</td></tr>
<tr><td>2</td><td>Modifier is subtracted from the item</td></tr>
<tr><td>3</td><td>Modifier is multiplied with the item</td></tr>
<tr><td>4</td><td>The item is divided by the modifier factor</td></tr>
</table></td></tr>
<tr><td colspan="2">contracts_mod_factor_i (Modifier Factor, Number of Contracts)</td></tr>
<tr><td>Datatype</td><td>INT32_T</td></tr>
<tr><td>Description</td><td>The modifier is used to recalculate the item after an underlying adjustment. The field is stored with 3 implicit decimals.</td></tr>
</table>

<table>
<tr><td colspan="2">contract_share_i (Contract Share)</td></tr>
<tr><td>Datatype</td><td>INT32_T</td></tr>
<tr><td>Description</td><td>The number of contracts in the delivery, including decimals, as defined for the instrument class.</td></tr>
<tr><td colspan="2">contract_size_i (Contract Size)</td></tr>
<tr><td>Datatype</td><td>INT32_T</td></tr>
<tr><td>Description</td><td>Number of Underlying entities per contract.</td></tr>
<tr><td colspan="2">contract_size_modifier_c (Modifier, Contract Size)</td></tr>
<tr><td>Datatype</td><td>UINT8_T</td></tr>
<tr><td>Description</td><td>The modifier is used to recalculate the item after an underlying adjustment. The field is stored with 3 implicit decimals.</td></tr>
<tr><td>Value Set</td><td>
<table>
<tr><th>value</th><th>description</th></tr>
<tr><td>1</td><td>Modifier is added to the item</td></tr>
<tr><td>2</td><td>Modifier is subtracted from the item</td></tr>
<tr><td>3</td><td>Modifier is multiplied with the item</td></tr>
<tr><td>4</td><td>The item is divided by the modifier factor</td></tr>
</table>
</td></tr>
<tr><td colspan="2">contr_size_mod_factor_i (Modifier Factor, Contract Size)</td></tr>
<tr><td>Datatype</td><td>INT32_T</td></tr>
<tr><td>Description</td><td>The modifier is used to recalculate the item after an underlying adjustment. The field is stored with 5 implicit decimals.</td></tr>
<tr><td colspan="2">copies_n (COPIES_N)</td></tr>
<tr><td>Datatype</td><td>UINT16_T</td></tr>
<tr><td colspan="2">country_c (Country Number)</td></tr>
<tr><td>Datatype</td><td>UINT8_T</td></tr>
<tr><td>Description</td><td>Country and exchange identity. Country Number is a part of the Series definition.</td></tr>
<tr><td colspan="2">country_id_s (Name, Country)</td></tr>
<tr><td>Datatype</td><td>char[2]</td></tr>
<tr><td>Description</td><td>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.</td></tr>
<tr><td colspan="2">country_s (Country)</td></tr>
<tr><td>Datatype</td><td>char[2]</td></tr>
<tr><td>Description</td><td>The country ID where the exchange is located.</td></tr>
<tr><td colspan="2">coupon_frequency_n (Coupon Frequency)</td></tr>
<tr><td>Datatype</td><td>UINT16_T</td></tr>
<tr><td>Description</td><td>Number of coupons per year for bond underlying.</td></tr>
<tr><td colspan="2">coupon_interest_i (Coupon Interest)</td></tr>
<tr><td>Datatype</td><td>UINT32_T</td></tr>
</table>

<table>
<tr><td>Description</td><td>Coupon interest, decimal value stored with 6 decimals, e.g. 11% is stored as 110000.</td></tr>
<tr><td colspan="2">coupon_settlement_days_n (Coupon Settlement Days)</td></tr>
<tr><td>Datatype</td><td>UINT16_T</td></tr>
<tr><td>Description</td><td>Number of settlement days at coupon.</td></tr>
<tr><td colspan="2">created_date_s (Date, Created)</td></tr>
<tr><td>Datatype</td><td>char[8]</td></tr>
<tr><td>Description</td><td>Date in ASCII. Format: YYYYMMDD. Defines the creation date of the item.</td></tr>
<tr><td colspan="2">created_time_s (Time, Created)</td></tr>
<tr><td>Datatype</td><td>char[6]</td></tr>
<tr><td>Description</td><td>Defines the creation time of the item. Format: HHMMSS.</td></tr>
<tr><td colspan="2">credit_class_s (Credit Class)</td></tr>
<tr><td>Datatype</td><td>char[3]</td></tr>
<tr><td>Description</td><td>Exchange specific contents and interpretation.</td></tr>
<tr><td colspan="2">csd_id_s (CSD, Identity)</td></tr>
<tr><td>Datatype</td><td>char[12]</td></tr>
<tr><td>Description</td><td>Specifies the clearance system that is connected to instrument class.</td></tr>
<tr><td colspan="2">cst_id_n (Customer Number)</td></tr>
<tr><td>Datatype</td><td>UINT16_T</td></tr>
<tr><td>Description</td><td>A unique number that identified the member, used when subscribing for directed broadcast information.</td></tr>
<tr><td colspan="2">currency_code (CURRENCY_CODE)</td></tr>
<tr><td>Datatype</td><td>char[3]</td></tr>
<tr><td>Description</td><td>Intermediate field.</td></tr>
<tr><td colspan="2">cur_unit_c (Currency Unit)</td></tr>
<tr><td>Datatype</td><td>UINT8_T</td></tr>
<tr><td>Description</td><td>Specifies the currency unit for underlying prices.</td></tr>
<tr><td>Value Set</td><td><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></td></tr>
<tr><td colspan="2">customer_info_s (Customer, Information)</td></tr>
<tr><td>Datatype</td><td>char[15]</td></tr>
<tr><td>Description</td><td>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.</td></tr>
<tr><td colspan="2">cust_bank_id_s (Custodian Bank)</td></tr>
<tr><td>Datatype</td><td>char[12]</td></tr>
<tr><td>Description</td><td>Identity of custodian bank</td></tr>
</table>

date_adjust_s (Date, Adjust)	
Datatype	char[8]
Description	Date of the adjustment. In ASCII format: YYYYMMDD
date_booksclose_s (Booksclose Date)	
Datatype	char[8]
Description	Customer Specific field. Booksclose date for bond underlying, YYYYMMDD.
date_closing_s (Date, Closing)	
Datatype	char[8]
Description	Closing date YYYYMMDD.
date_conversion_s (Date, Conversion)	
Datatype	char[8]
Description	Date in ASCII. Format: YYYYMMDD
date_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_exception_s (Date, Exception)	

Datatype	char[8]
Description	Exception date when a different trading session is used compared to normal days or when the market is open for a day when it is normally closed. 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)	

<table>
<tr><td>Datatype</td><td>char[8]</td></tr>
<tr><td>Description</td><td>The payout date for lottery bonds.
Format: YYYYMMDD.</td></tr>
<tr><td colspan="2">date_proceed_s (Date, Proceed)</td></tr>
<tr><td>Datatype</td><td>char[8]</td></tr>
<tr><td>Description</td><td>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.</td></tr>
<tr><td colspan="2">date_release_s (Date, Issue)</td></tr>
<tr><td>Datatype</td><td>char[8]</td></tr>
<tr><td>Description</td><td>Issue date for fixed income underlying. Format: YYYYMMDD.</td></tr>
<tr><td colspan="2">date_s (Date)</td></tr>
<tr><td>Datatype</td><td>char[8]</td></tr>
<tr><td>Description</td><td>Date in ASCII. Format: YYYYMMDD</td></tr>
<tr><td colspan="2">date_termination_s (Date, Maturity)</td></tr>
<tr><td>Datatype</td><td>char[8]</td></tr>
<tr><td>Description</td><td>Maturity date for fixed income underlying, YYYYMMDD.</td></tr>
<tr><td colspan="2">date_trading_s (Date, Trading)</td></tr>
<tr><td>Datatype</td><td>char[8]</td></tr>
<tr><td>Description</td><td>Date in ASCII. Format: YYYYMMDD.</td></tr>
<tr><td colspan="2">days_in_interest_year_n (Days In Interest Year)</td></tr>
<tr><td>Datatype</td><td>UINT16_T</td></tr>
<tr><td>Description</td><td>Number of days in coupon period used for interest rate calculations.</td></tr>
<tr><td colspan="2">days_in_period_n (Days in Period)</td></tr>
<tr><td>Datatype</td><td>UINT16_T</td></tr>
<tr><td>Description</td><td>Number of days in a period</td></tr>
<tr><td colspan="2">days_in_year_n (Days in year)</td></tr>
<tr><td>Datatype</td><td>UINT16_T</td></tr>
<tr><td>Description</td><td>Number of days in the year according to the day count convention.</td></tr>
<tr><td colspan="2">day_calc_rule_c (Day Calculation Rule)</td></tr>
<tr><td>Datatype</td><td>UINT8_T</td></tr>
<tr><td>Description</td><td>Day Calculation Rule</td></tr>
<tr><td>Value Set</td><td><table><tr><th>name</th><th>value</th></tr><tr><td>ACTACT</td><td>1</td></tr><tr><td>ACTAFB</td><td>2</td></tr><tr><td>EU30360</td><td>3</td></tr></table></td></tr>
</table>

<table>
<tr><td></td><td><table><tr><th>name</th><th>value</th></tr><tr><td>US30360</td><td>4</td></tr><tr><td>ACT365</td><td>5</td></tr><tr><td>ACT360</td><td>6</td></tr></table></td></tr>
<tr><td colspan="2">day_count_conv_c (Day Count Convention)</td></tr>
<tr><td>Datatype</td><td>UINT8_T</td></tr>
<tr><td>Description</td><td>Day Count Convention</td></tr>
<tr><td>Value Set</td><td><table><tr><th>name</th><th>value</th></tr><tr><td>ACTACT</td><td>1</td></tr><tr><td>ACTAFB</td><td>2</td></tr><tr><td>EU30360</td><td>3</td></tr><tr><td>US30360</td><td>4</td></tr><tr><td>ACT365</td><td>5</td></tr><tr><td>ACT360</td><td>6</td></tr></table></td></tr>
<tr><td colspan="2">day_count_n (Day Count)</td></tr>
<tr><td>Datatype</td><td>UINT16_T</td></tr>
<tr><td>Description</td><td>Number of days in the year when calculating interest.</td></tr>
<tr><td colspan="2">db_operation_c (Operation)</td></tr>
<tr><td>Datatype</td><td>UINT8_T</td></tr>
<tr><td>Description</td><td>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.</td></tr>
<tr><td>Value Set</td><td><table><tr><th>name</th><th>value</th></tr><tr><td>Insert</td><td>1</td></tr><tr><td>Update</td><td>2</td></tr><tr><td>Remove</td><td>3</td></tr></table></td></tr>
<tr><td colspan="2">deal_number_i (Deal Number)</td></tr>
<tr><td>Datatype</td><td>INT32_T</td></tr>
<tr><td>Description</td><td>A number that identifies a specific deal. Deal number is unique within Instrument type</td></tr>
<tr><td colspan="2">deal_price_i (Price, Deal)</td></tr>
<tr><td>Datatype</td><td>INT32_T</td></tr>
<tr><td>Description</td><td>Defines the deal price.</td></tr>
<tr><td colspan="2">deal_price_modifier_c (Modifier, Deal Price)</td></tr>
<tr><td>Datatype</td><td>UINT8_T</td></tr>
</table>

<table>
<tr><td>Description</td><td>The modifier is used to recalculate the item after an underlying adjustment. The field is stored with 3 implicit decimals.</td></tr>
<tr><td>Value Set</td><td>
<table>
<tr><th>value</th><th>description</th></tr>
<tr><td>1</td><td>Modifier is added to the item</td></tr>
<tr><td>2</td><td>Modifier is subtracted from the item</td></tr>
<tr><td>3</td><td>Modifier is multiplied with the item</td></tr>
<tr><td>4</td><td>The item is divided by the modifier factor</td></tr>
</table>
</td></tr>
<tr><td colspan="2">deal_price_mod_factor_i (Modifier Factor, Deal Price)</td></tr>
<tr><td>Datatype</td><td>INT32_T</td></tr>
<tr><td>Description</td><td>The modifier is used to recalculate the item after an underlying adjustment. The field is stored with 7 implicit decimals</td></tr>
<tr><td colspan="2">deal_quantity_i (Quantity, Deal)</td></tr>
<tr><td>Datatype</td><td>INT64_T</td></tr>
<tr><td>Description</td><td>Defines number of contracts in a deal.</td></tr>
<tr><td colspan="2">deal_source_c (Deal Source)</td></tr>
<tr><td>Datatype</td><td>UINT8_T</td></tr>
<tr><td>Description</td><td>Refers to where the deal is created during the day :</td></tr>
<tr><td>Value Set</td><td>
<table>
<tr><th>name</th><th>value</th><th>description</th></tr>
<tr><td>deal_source_none</td><td>0</td><td>Internal use. Trades reported directly to the clearing subsystem.</td></tr>
<tr><td>deal_source_auto</td><td>1</td><td>Matched by system, automatically.</td></tr>
<tr><td>deal_source_manually</td><td>2</td><td>Matched by system, manually.</td></tr>
<tr><td>deal_source_outside_different</td><td>3</td><td>Matched Outside Exchange, Different participants</td></tr>
<tr><td>deal_source_outside_different_om</td><td>4</td><td>Matched outside exchange, different brokers, reg. by exchange.</td></tr>
<tr><td>deal_source_outside_same</td><td>5</td><td>Matched Outside Exchange, One participant</td></tr>
<tr><td>deal_source_outside_same_om</td><td>6</td><td>Matched outside exchange, one broker, reg. by exchange.</td></tr>
<tr><td>deal_source_auto_combo</td><td>7</td><td>Combination order matched against another combination order when matched by the Exchange, electronically.</td></tr>
<tr><td>deal_source_swap_box</td><td>8</td><td>Deal in a Swap Box instrument.</td></tr>
<tr><td>deal_source_auto_internal</td><td>9</td><td>Matched electronically, member internal.</td></tr>
</table>
</td></tr>
</table>

name	value	description
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.
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.

name	value	description
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	
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_give_up	130	Trade resulting from a give-up transaction.

name	value	description
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).
dec_in_contr_size_n (Decimals, Contract Size)	
Datatype	UINT16_T
Description	Number of implicit decimals in the Contract Size and the Price Quotation Factor fields.
dec_in_deliv_n (Decimals, Delivery)	
Datatype	UINT16_T
Description	Number of implicit decimals used in the delivery quantity.
dec_in_fixing_n (Decimals, Fixing)	
Datatype	UINT16_T
Description	Number of implicit decimals in Fixing.
dec_in_index_n (DEC_IN_INDEX_N)	
Datatype	UINT16_T
Description	Number of decimals used when calculating index.
dec_in_nominal_n (Decimals, Nominal)	
Datatype	UINT16_T
Description	Number of implicit decimals in the Nominal Value.
dec_in_premium_n (Decimals, Premium)	
Datatype	UINT16_T
Description	Number of implicit decimals in the premium/price.
dec_in_price_n (Decimals, Price)	
Datatype	UINT16_T

<table>
<tr><td>Description</td><td>Number of implicit decimals in the underlying price received from external sources.</td></tr>
<tr><td colspan="2">dec_in_strike_price_n (Decimals, Strike Price)</td></tr>
<tr><td>Datatype</td><td>UINT16_T</td></tr>
<tr><td>Description</td><td>Number of implicit decimals in the strike price.</td></tr>
<tr><td colspan="2">deliverable_c (Deliverable)</td></tr>
<tr><td>Datatype</td><td>UINT8_T</td></tr>
<tr><td>Description</td><td>Defines if a series can be delivered or not (Cash settlement):</td></tr>
<tr><td>Value Set</td><td><table>
<tr><th>value</th><th>description</th></tr>
<tr><td>1</td><td>Yes</td></tr>
<tr><td>2</td><td>No</td></tr>
</table></td></tr>
<tr><td colspan="2">delivery_number_i (Delivery, Number)</td></tr>
<tr><td>Datatype</td><td>INT32_T</td></tr>
<tr><td>Description</td><td>The delivery number for this delivery. Together with key number and series it is a unique number.</td></tr>
<tr><td colspan="2">delivery_origin_i (Delivery Origin)</td></tr>
<tr><td>Datatype</td><td>INT32_T</td></tr>
<tr><td>Description</td><td>The trade number for the trade that this delivery originates from. Together with Series it forms a unique trade identification.</td></tr>
<tr><td colspan="2">delivery_properties_u (Delivery Properties)</td></tr>
<tr><td>Datatype</td><td>UINT32_T</td></tr>
<tr><td>Description</td><td>Bit mask provides specific information about the delivery.</td></tr>
<tr><td>Value Set</td><td><table>
<tr><th>value</th><th>description</th></tr>
<tr><td>0</td><td>No information</td></tr>
<tr><td>1</td><td>DvP (Create DvP instruction)</td></tr>
<tr><td>2</td><td>SWIFT (Entered by SWIFT)</td></tr>
<tr><td>4</td><td>Transfer (Other quantity is zero)</td></tr>
<tr><td>8</td><td>Reversing (Reversing BD18)</td></tr>
<tr><td>16</td><td>Overtaking (Overtaking BD18)</td></tr>
<tr><td>32</td><td>Confirm (Holding DvP instruction needs confirmation)</td></tr>
<tr><td>64</td><td>Settled Ext (Don't create DvP instruction - the delivery will be settled externally)</td></tr>
<tr><td>128</td><td>Bill Delivery</td></tr>
<tr><td>256</td><td>Cross Clearinghouse Balance</td></tr>
<tr><td>512</td><td>Cross Border Give-up</td></tr>
<tr><td>1024</td><td>Additional Basket</td></tr>
<tr><td>8192</td><td>Do not reverse the sign of this delivery item</td></tr>
</table></td></tr>
</table>

<table>
<tr><td colspan="2">delivery_quantity_q (Quantity, Delivery)</td></tr>
<tr><td>Datatype</td><td>INT64_T</td></tr>
<tr><td>Description</td><td>Defines the quantity the delivery is based on.</td></tr>
<tr><td colspan="2">delivery_state_c (Delivery, State)</td></tr>
<tr><td>Datatype</td><td>UINT8_T</td></tr>
<tr><td>Description</td><td>Defines what state the delivery is in.</td></tr>
<tr><td>Value Set</td><td>
<table>
<tr><th>value</th><th>description</th></tr>
<tr><td>1</td><td>Normal</td></tr>
<tr><td>2</td><td>Rectified
The delivery is rolled back. There exists another rollback delivery that points to this delivery.</td></tr>
</table>
</td></tr>
<tr><td colspan="2">delivery_type_c (Delivery, Type)</td></tr>
<tr><td>Datatype</td><td>UINT8_T</td></tr>
<tr><td>Description</td><td>Defines what type the delivery is.</td></tr>
<tr><td>Value Set</td><td>
<table>
<tr><th>value</th><th>description</th></tr>
<tr><td>1</td><td>Normal</td></tr>
<tr><td>2</td><td>Rollback
The delivery offsets a previous delivery that is no longer valid. The original delivery is identified by the instrument type of the series in combination with original delivery number and original key number. The quantity delivery base reverse the original delivery.</td></tr>
<tr><td>3</td><td>Overtaking
The delivery superseeds a previous delivery that is no longer valid. The original delivery is identified by the instrument type of the series in combination with original delivery number and original key number.</td></tr>
<tr><td>4</td><td>Backdated
The delivery is backdated which entails that it concerns an event occuring on a previous clearing date.</td></tr>
</table>
</td></tr>
<tr><td colspan="2">delivery_unit_u (Delivery Unit)</td></tr>
<tr><td>Datatype</td><td>UINT32_T</td></tr>
<tr><td>Description</td><td>Trade reports, delivery items and dvp-instructions belong to a delivery unit.</td></tr>
<tr><td colspan="2">deliv_base_quantity_q (Quantity, Delivery Base)</td></tr>
<tr><td>Datatype</td><td>INT64_T</td></tr>
<tr><td>Description</td><td>Defines the quantity of the delivery base that is delivered.</td></tr>
<tr><td colspan="2">delta_alloc_time_n (Time, Allocation)</td></tr>
</table>

<table>
<tr><td>Datatype</td><td>UINT16_T</td></tr>
<tr><td>Description</td><td>Delta allocation time in minutes
after last trading time</td></tr>
<tr><td colspan="2">deny_exercise_q (Deny Exercise)</td></tr>
<tr><td>Datatype</td><td>INT64_T</td></tr>
<tr><td>Description</td><td>The number of held position that will NOT participate in exercise.</td></tr>
<tr><td colspan="2">derivate_level_n (Derivate Level)</td></tr>
<tr><td>Datatype</td><td>UINT16_T</td></tr>
<tr><td>Description</td><td>The derivate level of the instrument:</td></tr>
<tr><td>Value Set</td><td><table><tr><th>name</th><th>value</th></tr><tr><td>Spot</td><td>0</td></tr><tr><td>Derivate based on spot.</td><td>1</td></tr><tr><td>Derivative based on instrument level 1.</td><td>2</td></tr></table></td></tr>
<tr><td colspan="2">derived_from_s (Derived From)</td></tr>
<tr><td>Datatype</td><td>char[128]</td></tr>
<tr><td>Description</td><td>Defines what the underlying is derived from.</td></tr>
<tr><td colspan="2">derived_percentage_u (Derived Percentage)</td></tr>
<tr><td>Datatype</td><td>UINT32_T</td></tr>
<tr><td>Description</td><td>Defined how many percent the Derived From represent. Expressed with six implicit decimals.</td></tr>
<tr><td colspan="2">description_s (Description)</td></tr>
<tr><td>Datatype</td><td>char[40]</td></tr>
<tr><td>Description</td><td>Description field.</td></tr>
<tr><td colspan="2">desc_abbreviated_s (Description, Abbreviated)</td></tr>
<tr><td>Datatype</td><td>char[32]</td></tr>
<tr><td>Description</td><td>An abbreviated textual description.</td></tr>
<tr><td colspan="2">desc_long_s (Description, Long)</td></tr>
<tr><td>Datatype</td><td>char[128]</td></tr>
<tr><td>Description</td><td>A textual description.</td></tr>
<tr><td colspan="2">diary_number_s (Diary Number)</td></tr>
<tr><td>Datatype</td><td>char[15]</td></tr>
<tr><td>Description</td><td>The diary number for this account.</td></tr>
<tr><td colspan="2">directed_trade_information_c (Directed Trade Information)</td></tr>
<tr><td>Datatype</td><td>UINT8_T</td></tr>
<tr><td>Description</td><td>Specifies how the directed trade broadcast is distributed.</td></tr>
<tr><td>Value Set</td><td><table><tr><th>name</th><th>value</th></tr><tr><td>Without Counterparty</td><td>1</td></tr></table></td></tr>
</table>

<table>
<tr><td></td><td><table><tr><th>name</th><th>value</th></tr><tr><td>With Counterparty</td><td>2</td></tr></table></td></tr>
<tr><td colspan="2">dividend_i (Dividend)</td></tr>
<tr><td>Datatype</td><td>UINT32_T</td></tr>
<tr><td>Description</td><td>The dividend for the stock.</td></tr>
<tr><td colspan="2">download_ref_number_q (Download Reference Number)</td></tr>
<tr><td>Datatype</td><td>INT64_T</td></tr>
<tr><td>Description</td><td>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.</td></tr>
<tr><td colspan="2">ds_attribute_q (Deal Source Attribute)</td></tr>
<tr><td>Datatype</td><td>INT64_T</td></tr>
<tr><td>Description</td><td>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</td></tr>
<tr><td colspan="2">dvp_account_s (DVP Account)</td></tr>
<tr><td>Datatype</td><td>char[24]</td></tr>
<tr><td>Description</td><td>Sub account/Security account or Cash record/Cash account identification designated for deliveries.</td></tr>
<tr><td colspan="2">effective_exp_date_s (Effective Expiration Date)</td></tr>
<tr><td>Datatype</td><td>char[8]</td></tr>
<tr><td>Description</td><td>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.</td></tr>
<tr><td colspan="2">end_date_s (Date, End)</td></tr>
<tr><td>Datatype</td><td>char[8]</td></tr>
<tr><td>Description</td><td>End date. Format: YYYYMMDD.</td></tr>
<tr><td colspan="2">eom_count_conv_c (End of Month Count Convention)</td></tr>
<tr><td>Datatype</td><td>UINT8_T</td></tr>
<tr><td>Description</td><td>End of Month Count Convention</td></tr>
<tr><td>Value Set</td><td><table><tr><th>name</th><th>value</th></tr><tr><td>SAME</td><td>1</td></tr></table></td></tr>
</table>

<table>
<tr><td></td><td><table><tr><th>name</th><th>value</th></tr><tr><td>LAST360</td><td>2</td></tr><tr><td>LAST</td><td>3</td></tr></table></td></tr>
<tr><td colspan="2">error_id_u (Error Identity)</td></tr>
<tr><td>Datatype</td><td>UINT32_T</td></tr>
<tr><td>Description</td><td>An identity that refers to the source for error. For trade errors, this is the trade number.</td></tr>
<tr><td colspan="2">error_operation_s (Error, Operation)</td></tr>
<tr><td>Datatype</td><td>char[10]</td></tr>
<tr><td>Description</td><td>Defines what type of operation caused the error message.</td></tr>
<tr><td colspan="2">error_problem_s (Error, Problem)</td></tr>
<tr><td>Datatype</td><td>char[40]</td></tr>
<tr><td>Description</td><td>The error message.</td></tr>
<tr><td colspan="2">event_type_i (Stimuli Event)</td></tr>
<tr><td>Datatype</td><td>INT32_T</td></tr>
<tr><td>Description</td><td>Defines the reason that caused the contractual event.</td></tr>
<tr><td>Value Set</td><td><table><tr><th>value</th><th>description</th></tr><tr><td>1</td><td>Trade</td></tr><tr><td>2</td><td>Transfer</td></tr><tr><td>3</td><td>Rectify</td></tr><tr><td>4</td><td>Mark to Market</td></tr><tr><td>5</td><td>Closing</td></tr><tr><td>6</td><td>Exercise</td></tr><tr><td>7</td><td>Assign</td></tr><tr><td>8</td><td>Dividend</td></tr><tr><td>9</td><td>New Contract Trade</td></tr><tr><td>10</td><td>Give Up</td></tr><tr><td>11</td><td>Closing Trade</td></tr><tr><td>12</td><td>Delivery Flow</td></tr><tr><td>13</td><td>DVP Settled</td></tr></table></td></tr>
<tr><td colspan="2">exchange_info_s (Exchange, Information)</td></tr>
<tr><td>Datatype</td><td>CHAR[32]</td></tr>
<tr><td>Description</td><td>This is an exchange specific field and can be used for different purposes, e.g. as a free text field.</td></tr>
<tr><td colspan="2">exchange_short_s (Exchange, Short Name)</td></tr>
<tr><td>Datatype</td><td>char[4]</td></tr>
</table>

<table>
<tr><td>Description</td><td>Short name for exchange</td></tr>
<tr><td colspan="2">exch_order_type_n (Order Type, Exchange)</td></tr>
<tr><td>Datatype</td><td>UINT16_T</td></tr>
<tr><td>Description</td><td>This is bit-coded field for exchange specific order types and attributes.</td></tr>
<tr><td>Value Set</td><td>
<table>
<tr><th>name</th><th>value</th><th>description</th></tr>
<tr><td>EXCH_ORDER_TYPE_NOT_DEFINED</td><td>0</td><td>Not applicable.</td></tr>
<tr><td>EXCH_ORDER_TYPE_FORCE</td><td>1</td><td>Force</td></tr>
<tr><td>EXCH_ORDER_TYPE_SHORT_SELL</td><td>2</td><td>Short Sell
Short sell order condition.</td></tr>
<tr><td>EXCH_ORDER_TYPE_MARKET_BID</td><td>4</td><td>Market Bid
Market bid order condition(exchange specific).</td></tr>
<tr><td>EXCH_ORDER_TYPE_PRICE_STAB</td><td>8</td><td>Price Stabilization
Price stabilization order condition (exchange specific).</td></tr>
<tr><td>EXCH_ORDER_TYPE_OVERRIDE_CRS</td><td>16</td><td>Override Crossing
Override crossing condition (exchange specific).</td></tr>
<tr><td>EXCH_ORDER_TYPE_UNDISCLOSED</td><td>32</td><td>Undisclosed</td></tr>
<tr><td>EXCH_ORDER_TYPE_CENTRE_POINT</td><td>64</td><td>Centre Point</td></tr>
<tr><td>EXCH_ORDER_TYPE_ALWAYS_INACTIVE</td><td>128</td><td>Always Inactive
Always centrally inactive order, not possible to activate.
Only valid for transactions to enter inactive orders (exchange specific).</td></tr>
<tr><td>EXCH_ORDER_TYPE_CPPX</td><td>256</td><td>Centre Point Priority Crossing</td></tr>
<tr><td>EXCH_ORDER_TYPE_SESSION_STATE</td><td>512</td><td>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</td></tr>
</table>
</td></tr>
<tr><td colspan="2">exclusive_opening_sell_c (Exclusive Opening Sell)</td></tr>
<tr><td>Datatype</td><td>UINT8_T</td></tr>
<tr><td>Description</td><td>Is the account allowed to exclusive opening sell?</td></tr>
<tr><td>Value Set</td><td>
<table>
<tr><th>value</th><th>description</th></tr>
<tr><td>1</td><td>Yes</td></tr>
</table>
</td></tr>
</table>

<table>
<tr><td></td><td><table><tr><th>value</th><th>description</th></tr><tr><td>2</td><td>No</td></tr></table></td></tr>
<tr><td colspan="2">execution_event_nbr_u (Execution number)</td></tr>
<tr><td>Datatype</td><td>UINT64_T</td></tr>
<tr><td>Description</td><td>An ever increasing number per partition, assigned to an execution event.</td></tr>
<tr><td colspan="2">exercisenumber (EXERCISENUMBER)</td></tr>
<tr><td>Datatype</td><td>INT32_T</td></tr>
<tr><td>Description</td><td>intermediate field.</td></tr>
<tr><td colspan="2">exercise_number_i (Exercise, Request Number)</td></tr>
<tr><td>Datatype</td><td>INT32_T</td></tr>
<tr><td>Description</td><td>Identifies each part in an exercise request.</td></tr>
<tr><td colspan="2">exerc_limit_i (Exercise, Limit)</td></tr>
<tr><td>Datatype</td><td>INT32_T</td></tr>
<tr><td>Description</td><td>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.</td></tr>
<tr><td colspan="2">exerc_limit_unit_c (Exercise, Limit Unit)</td></tr>
<tr><td>Datatype</td><td>UINT8_T</td></tr>
<tr><td>Description</td><td>What type is the Exercise Limit Unit?</td></tr>
<tr><td>Value Set</td><td><table><tr><th>value</th><th>description</th></tr><tr><td>1</td><td>Absolute Value</td></tr><tr><td>2</td><td>Percentage (%)</td></tr></table></td></tr>
<tr><td colspan="2">expiration_date_n (Date, Expiration)</td></tr>
<tr><td>Datatype</td><td>UINT16_T</td></tr>
<tr><td>Description</td><td>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</td></tr>
<tr><td colspan="2">extended_info_n (Extended Information)</td></tr>
<tr><td>Datatype</td><td>UINT16_T</td></tr>
<tr><td>Description</td><td>Not applicable.</td></tr>
<tr><td>Value Set</td><td><table><tr><th>value</th><th>description</th></tr><tr><td>0</td><td>Not Applicable</td></tr></table></td></tr>
<tr><td colspan="2">external_id_s (External Price Feed Identity)</td></tr>
</table>

<table>
<tr><td>Datatype</td><td>char[40]</td></tr>
<tr><td>Description</td><td>External Price feed identity</td></tr>
<tr><td colspan="2">ext_acc_controller_s (External Account Controller)</td></tr>
<tr><td>Datatype</td><td>char[15]</td></tr>
<tr><td>Description</td><td>External account controller. May hold BIC, CSD member id etc.</td></tr>
<tr><td colspan="2">ext_acc_id_s (External Account ID)</td></tr>
<tr><td>Datatype</td><td>char[34]</td></tr>
<tr><td>Description</td><td>External account id. A bank or CSD account number.</td></tr>
<tr><td colspan="2">ext_acc_registrar_s (External Account Registrar)</td></tr>
<tr><td>Datatype</td><td>char[12]</td></tr>
<tr><td>Description</td><td>External account registrar. May hold names like VPS, SWIFT etc.</td></tr>
<tr><td colspan="2">ext_info_source_c (External Information Source)</td></tr>
<tr><td>Datatype</td><td>UINT8_T</td></tr>
<tr><td>Description</td><td>Specifies whether or not the data source for distributed prices is sent into the system with an external transaction.</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">ext_provider_c (External Price Feed Provider)</td></tr>
<tr><td>Datatype</td><td>CHAR</td></tr>
<tr><td>Description</td><td>External Price feed provider</td></tr>
<tr><td>Value Set</td><td><table>
<tr><th>name</th><th>value</th></tr>
<tr><td>NMF</td><td>N</td></tr>
<tr><td>Six</td><td>S</td></tr>
<tr><td>Six OMX</td><td>O</td></tr>
<tr><td>Direct Feed</td><td>F</td></tr>
<tr><td>Direct Feed OPRA</td><td>R</td></tr>
<tr><td>Transaction</td><td>T</td></tr>
<tr><td>LMIL</td><td>L</td></tr>
<tr><td>Reuter SSL</td><td>E</td></tr>
</table></td></tr>
<tr><td colspan="2">ext_seq_nbr_i (External Clearinghouse, Sequence Number)</td></tr>
<tr><td>Datatype</td><td>INT32_T</td></tr>
<tr><td>Description</td><td>An identity that the clearinghouse or exchange can assign to a trade. Exchange specific.</td></tr>
<tr><td colspan="2">ext_status_i (Return Status)</td></tr>
<tr><td>Datatype</td><td>INT32_T</td></tr>
</table>

<table>
<tr><td>Description</td><td>Defines return status, configuration specific.</td></tr>
<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>
</table>

fee_type_s (Account Fee Type)	
Datatype	char[12]
Description	Defines the account fee type for an account.
file_type_s (File Type)	
Datatype	char[8]
Description	The string representing the file type, i.e. suffix.
filler_1_s (Filler)	
Datatype	CHAR
Description	Filler for alignment.
filler_2_s (Filler)	
Datatype	char[2]
Description	Filler for alignment.
filler_3_s (Filler)	
Datatype	char[3]
Description	Filler for alignment.
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.
first_settlement_date_s (Date, First Settlement)	
Datatype	char[8]
Description	First Settlement Date in format YYYYMMDD.
fixed_income_type_c (Fixed Income Type)	
Datatype	UINT8_T
Description	Type of fixed income security:
Value Set	

value	description
0	Not applicable
1	Bill
2	Bond
3	Index Linked Bonds
4	Bond Floating
5	Lottery Bond
6	Convertible Bond
7	Structured Bond

<table>
<tr><td></td><td><table><tr><th>value</th><th>description</th></tr><tr><td>8</td><td>Fixing</td></tr><tr><td>9</td><td>Credit Certificates</td></tr><tr><td>10</td><td>Deposit</td></tr><tr><td>11</td><td>RIBA</td></tr></table></td></tr>
<tr><td colspan="2">fixed_or_float_c (Fixed or Float)</td></tr>
<tr><td>Datatype</td><td>UINT8_T</td></tr>
<tr><td>Description</td><td>Fixed or float rate</td></tr>
<tr><td>Value Set</td><td><table><tr><th>name</th><th>value</th></tr><tr><td>Fixed</td><td>1</td></tr><tr><td>Float</td><td>2</td></tr></table></td></tr>
<tr><td colspan="2">fixing_req_c (FIXING_REQ_C)</td></tr>
<tr><td>Datatype</td><td>UINT8_T</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">fixing_value_i (Fixing Value)</td></tr>
<tr><td>Datatype</td><td>INT32_T</td></tr>
<tr><td>Description</td><td>A value defined for a series a given date, used for clearing purposes. The Decimals, Fixing field defines the number decimals used.</td></tr>
<tr><td colspan="2">forward_style_c (Style, Forward)</td></tr>
<tr><td>Datatype</td><td>UINT8_T</td></tr>
<tr><td>Description</td><td>Defines if this an Instrument Group where corresponding Instrument Series are forward styled.</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>1</td><td>Normal</td></tr><tr><td>2</td><td>CfD</td></tr></table></td></tr>
<tr><td colspan="2">free_text_80_s (Text , Free)</td></tr>
<tr><td>Datatype</td><td>char[80]</td></tr>
<tr><td>Description</td><td>Defines a free text buffer.</td></tr>
<tr><td colspan="2">from_date_s (Date, From)</td></tr>
<tr><td>Datatype</td><td>char[8]</td></tr>
<tr><td>Description</td><td>From date. Format: YYYYMMDD.</td></tr>
</table>

<table>
<tr><td colspan="2">from_time_s (Time, From)</td></tr>
<tr><td>Datatype</td><td>char[6]</td></tr>
<tr><td>Description</td><td>Defines the from time. Format: HHMMSS.</td></tr>
<tr><td colspan="2">full_answer_c (Full Answer)</td></tr>
<tr><td>Datatype</td><td>UINT8_T</td></tr>
<tr><td>Description</td><td>A full answer is enforced in the delta query.</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">future_styled_c (Option, Future Styled)</td></tr>
<tr><td>Datatype</td><td>UINT8_T</td></tr>
<tr><td>Description</td><td>If the option is a future styled option:</td></tr>
<tr><td>Value Set</td><td><table>
<tr><th>value</th><th>description</th></tr>
<tr><td>1</td><td>Yes</td></tr>
<tr><td>2</td><td>No</td></tr>
</table></td></tr>
<tr><td colspan="2">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>
</table></td></tr>
</table>

<table>
<tr><td></td><td><table><tr><th>value</th><th>description</th></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>
<tr><td>Description</td><td>User-supplied information to a give-up request. This information is passed through the clearing system without any processing or validation.</td></tr>
<tr><td colspan="2">giving_up_exchange_s (Giving Up Exchange)</td></tr>
<tr><td>Datatype</td><td>char[2]</td></tr>
<tr><td>Description</td><td>The exchange of the owner of the trade that was given up.</td></tr>
<tr><td colspan="2">global_deal_no_u (Global Deal Number)</td></tr>
<tr><td>Datatype</td><td>UINT32_T</td></tr>
<tr><td>Description</td><td>A number that together with series identifies a specific deal. The number is used as reference from outside clearing system.</td></tr>
<tr><td colspan="2">gross_open_interest_q (Gross Open Interest)</td></tr>
<tr><td>Datatype</td><td>UINT64_T</td></tr>
<tr><td>Description</td><td>Defines gross open interest.</td></tr>
<tr><td colspan="2">group_short_name_s (Short Name, Instrument Group)</td></tr>
<tr><td>Datatype</td><td>char[15]</td></tr>
<tr><td>Description</td><td>Defines a short description of the instrument group.</td></tr>
<tr><td colspan="2">group_type_c (Group, Type)</td></tr>
<tr><td>Datatype</td><td>UINT8_T</td></tr>
<tr><td>Description</td><td>Defines the type of instrument group.</td></tr>
<tr><td>Value Set</td><td><table>
<tr><th>name</th><th>value</th><th>description</th></tr>
<tr><td>group_type_undefined</td><td>0</td><td>Undefined</td></tr>
<tr><td>group_type_option</td><td>1</td><td>Option</td></tr>
<tr><td>group_type_forward</td><td>2</td><td>Forward</td></tr>
<tr><td>group_type_future</td><td>3</td><td>Future</td></tr>
<tr><td>group_type_fra</td><td>4</td><td>FRA</td></tr>
<tr><td>group_type_cash</td><td>5</td><td>Cash</td></tr>
<tr><td>group_type_payment</td><td>6</td><td>Payment</td></tr>
<tr><td>group_type_exchange_rate</td><td>7</td><td>Exchange Rate</td></tr>
<tr><td>group_type_interest_rate_swap</td><td>8</td><td>Interest Rate Swap</td></tr>
<tr><td>group_type_repo</td><td>9</td><td>REPO</td></tr>
<tr><td>group_type_synth_box_leg</td><td>10</td><td>Synthetic Box Leg/Reference</td></tr>
<tr><td>group_type_standard_combo</td><td>11</td><td>Standard Combination</td></tr>
</table></td></tr>
</table>

name	value	description
group_type_guarantee	12	Guarantee
group_type_otc_general	13	OTC General
group_type_equity_warrant	14	Equity Warrant
group_type_security_lending	15	Security Lending

gup_reason_i (Give Up, Broadcast Reason)	
Datatype	INT32_T
Description	Defines the reason why the Directed Give Up broadcast was sent.
Value Set	

value	description
1	Holding
2	Confirmed .
3	Rejected .
4	Delete Holding .
5	Deleted .
6	Extended .

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

hhmmss_s (Time, External)	
Datatype	char[6]
Description	Time in ASCII. Format: HHMMSS.
hidden_vol_meth_n (Method, Hidden Volume)	
Datatype	UINT16_T
Description	Hidden Volume Method:
Value Set	

value	description
0	No hidden used

<table>
<tr><td></td><td><table><tr><th>value</th><th>description</th></tr><tr><td>1</td><td>Normal</td></tr><tr><td>2</td><td>Additional</td></tr></table></td></tr>
<tr><td colspan="2">history_ex_i (HISTORY_EX_I)</td></tr>
<tr><td>Datatype</td><td>INT32_T</td></tr>
<tr><td>Description</td><td>History exercised</td></tr>
<tr><td colspan="2">identity (IDENTITY)</td></tr>
<tr><td>Datatype</td><td>char[5]</td></tr>
<tr><td>Description</td><td>Intermediate field.</td></tr>
<tr><td colspan="2">inc_id (INC_ID)</td></tr>
<tr><td>Datatype</td><td>char[14]</td></tr>
<tr><td>Description</td><td>Intermediate field.</td></tr>
<tr><td colspan="2">inc_id_s (Instrument Class, Identity)</td></tr>
<tr><td>Datatype</td><td>char[14]</td></tr>
<tr><td>Description</td><td>The ASCII representation of the instrument class.</td></tr>
<tr><td colspan="2">index_at_dated_i (INDEX_AT_DATED_I)</td></tr>
<tr><td>Datatype</td><td>INT32_T</td></tr>
<tr><td>Description</td><td>Index Value at Dated Date, 2 decimals</td></tr>
<tr><td colspan="2">index_market_c (Index Market)</td></tr>
<tr><td>Datatype</td><td>UINT8_T</td></tr>
<tr><td>Description</td><td>Indicates if the market is an index market or not</td></tr>
<tr><td>Value Set</td><td><table><tr><th>value</th><th>description</th></tr><tr><td>1</td><td>Yes</td></tr><tr><td>2</td><td>No</td></tr></table></td></tr>
<tr><td colspan="2">index_value_i (INDEX_VALUE_I)</td></tr>
<tr><td>Datatype</td><td>INT32_T</td></tr>
<tr><td>Description</td><td>Index Value, 2 decimals</td></tr>
<tr><td colspan="2">indicative_prices_c (Indicative Prices)</td></tr>
<tr><td>Datatype</td><td>UINT8_T</td></tr>
<tr><td>Description</td><td>Indicative Prices</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">info_type_i (Information Type)</td></tr>
</table>

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
19	Fixing ready
20	All securities closed
22	Extracted data for report generating are ready (Kofex)
23	NRS batch data loaded completed
24	NRS batch data loaded started
26	Stock deliveries ready
27	Reversed Stock deliveries ready
28	Bilateral Delivery Instructions ready
32	Delivery
41	Margin Evening Prices ready
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

value	description
47	Margin information from Lateevening ready
48	Margin summary information from Lateevening ready
49	API data from Intra Day Margin Calculation ready
52	Margin summary information from old dateready
53	Start owl cycle
54	Intra Day Margin Calculation product area ready
64	Expiration information
98	Final Fixing value established
100	Daily Trade statistics information
101	Revised Daily Trade statistics information
128	Paynote information
200	Official price ready (LME only)
201	Evening margin file ready (KOFEX specific)
202	Intra day margin file ready (KOFEX specific)
256	Used in queries to get possible reports
257	Vector files ready
260	Settlement note
261	Trades on trading account zero days forward
263	Settlement note futures
265	Settlement note ELEX
280	Cancellation note
285	Settlement notes, overtaking trades older than 1 day
290	Settlement note (position accounts)
291	Cancellation note (position accounts)
292	Settlement notes, overtaking... (position account)
293	Settlement note futures (position accounts)
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

value	description
344	Monthly cash settlement, security lending
350	Cash settlement options
351	Cash settlement forwards
352	Cash settlement forwards trading accounts
353	Cash settlement swaps
355	Monthly cash settlement forwards & IMM-FRA, detailed
356	Monthly cash settlement forwards & IMM-FRA
357	Expiration cash settlement forwards & IMM-FRA
358	Expiration cash settlement forwards & IMM-FRA/summary on account
359	Expiration cash settlement forwards & IMM-FRA/sumary on member
360	Expiration settlement FX Forwards
361	Expiration Tailor-Made Bond Forward
362	Cash settlement STINA
363	Accumulated Compound Rate STINA
370	Delivery
371	Delivery instruction security lending
373	Delivery advice summary
374	Delivery instruction collect note security lending
375	Delivery summary
376	Delivery fees new contracts
377	Delivery fees new contracts, summary on customer
379	DPMON Clearing Mgr Total Margin Req Summary
380	DPMON Product Area Collateral Summary
381	Margin and position listing
382	Margin requirement summary
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

value	description
389	Clearing manager total margin requirement summary
390	Position and position overview
391	Non-propagated Margin and position listing
392	Member product area collateral summary
393	Evening Risk Parameter File, Central, Exchange 1
394	Evening Risk Parameter File, Central, Exchange 2
395	Intra Day Risk Parameter File, Central, Exchange 1
396	Intra Day Risk Parameter File, Central, Exchange 2
397	Preliminary Risk Parameter File, Central, Exchange 1
398	Preliminary Risk Parameter File, Central, Exchange 2
400	Delivery instruction stocks (net)
401	Delivery instruction bonds
403	Evening Risk Parameter File, Member, Exchange 1
404	Evening Risk Parameter File, Member, Exchange 2
405	Intra Day Risk Parameter File, Member, Exchange 1
406	Intra Day Risk Parameter File, Member, Exchange 2
407	Preliminary Risk Parameter File, Member, Exchange 1
408	Preliminary Risk Parameter File, Member, Exchange 2
410	Payment notes
411	Settlement amounts, customer
412	Separate fees
420	Changes of position
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

value	description
430	Trades/daily account
431	Rectified trades during the day
432	Position transfer trades during the day
433	Forecast closing
434	Forecast closing, summary
436	After hours trades
437	Customer Position Exceeding the Limits
438	Rebate per customer
439	FX clearing
440	FX expiration
441	Total margin requirements
442	Total settlement amounts
443	Power positions
444	Cascade options
445	Cascade forwards
446	Trades with counterparts
447	Trades per customer account with fees
448	Position not assign in exercise
449	FX Clearing, sorted by counterparts
450	Nord pool daily trade list
451	Nord pool clearing list summary for brokers
452	Nord pool clearing list
453	Pulpex option exercise note
454	Pulpex future expiration note
455	Clearing information on exercise, closing & markto-market
456	Discount per customer, rule and account
457	NOS fee list
458	Delivery note, zero-day forwards
459	Delivery note, summary
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)

value	description
506	Daily settlement reports (general clearing members)
507	Daily settlement reports
508	Margin activity reports
509	Cash transfer instructions (credit)
510	Cash transfer instructions (debit)
511	Options exercised and assigns
512	Consolidated positions activity (options)
513	Final contract reports (options)
514	Consolidated positions activity (futures)
515	Final contract reports (futures)
516	Monthly interest and accommodation
517	Monthly fees reports
518	Unsettled delivery report
519	Deliver/Receive reports
520	Exercise by exceptions
521	Options expired positions
522	Intra day margin activity reports
523	Give-up trades for executor
524	Give-up trades for clearing broker
525	Exercised/Expired options to be settled
541	DPMON margin and position
542	DPMON margin requirement summary
543	DPMON data used for margin calc
544	DPMON data used for margin calc CO
545	DPMON security bank summary
546	DPMON clearing manager summary
547	DPMON non-prop margin and position
548	DPMON margins
549	DPMON price alarm limit
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

value	description
556	IDMON security bank summary
557	IDMON clearing manager summary
558	IDMON non-prop margin and position
559	IDMON margin report
560	IDMON price dump
561	RCAR worst
562	RCAR final scenario
563	RCAR top 10
564	RCAR detailed
566	DPMON Margin alarm limits
567	IDMON Margin alarm report
568	Risk parameter report
566	DPMON Margin alarm limits
590	DPMON Margin and position external
591	DPMON Data used for margin calc external
592	Data used for margin calc CO
593	Margin evening prices
594	Intray Param Change Report
595	Parameter Value Report
596	Window class Value Report
597	DPMON Parameter Value Report
598	DPMON Window class Value Report
600	Member order list report (CED only)
601	Member trade list report (CED only)
602	Market trades
603	Option Give up (for the executor member)
604	Option Give up (for the clearing broker member)
605	MS33 (CASSA report id)
606	MS59 (CASSA report id)
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

value	description
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 requirement summary for Late Evening1
1389	Clearing manager total margin requirement summary for Late Evening1
1390	Position and position overview for Late Evening1
1391	Non-propagated Margin and position listing for Late Evening1
1392	Member product area collateral summary for Late Evening1
1561	RCAR worst for Late Evening1
1562	RCAR final scenario for Late Evening1
1563	RCAR top 10 for Late Evening1
1564	RCAR detailed for Late Evening1
1592	Data used for margin calc CO for Late Evening1

ing_id_s (Instrument Group Identity)	
Datatype	char[3]
Description	The ASCII representation of the instrument group.
initial_trr_min_value_u (Initial Trade Report, Minimum Order Value.)	
Datatype	INT64_T
Description	Not applicable.
instance_c (Instance, Number)	
Datatype	UINT8_T
Description	Defines one specific instance for multiple processes.
instance_next_c (Next Instance Number)	
Datatype	UINT8_T
Description	Next instance number for multiple processes.
instigant_c (Instigant)	

<table>
<tr><td>Datatype</td><td>UINT8_T</td></tr>
<tr><td>Description</td><td>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.</td></tr>
<tr><td>Value Set</td><td><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></td></tr>
<tr><td colspan="2">instrument_group_c (Instrument Group)</td></tr>
<tr><td>Datatype</td><td>UINT8_T</td></tr>
<tr><td>Description</td><td>A unique binary representation of the instrument group.</td></tr>
<tr><td colspan="2">ins_id (INS_ID)</td></tr>
<tr><td>Datatype</td><td>char[32]</td></tr>
<tr><td>Description</td><td>Intermediate field.</td></tr>
<tr><td colspan="2">ins_id_s (Series, Identity)</td></tr>
<tr><td>Datatype</td><td>char[32]</td></tr>
<tr><td>Description</td><td>Instrument Series name is ASCII.</td></tr>
<tr><td colspan="2">interest_rate_i (Interest Rate)</td></tr>
<tr><td>Datatype</td><td>INT32_T</td></tr>
<tr><td>Description</td><td>Defines the Interest Rate for the underlying. Decimal value stored with 6 implicit decimal, e.g. 11% is stored as 110000.</td></tr>
<tr><td colspan="2">int_id (INT_ID)</td></tr>
<tr><td>Datatype</td><td>char[8]</td></tr>
<tr><td>Description</td><td>Intermediate field.</td></tr>
<tr><td colspan="2">int_id_s (Instrument, Identity)</td></tr>
<tr><td>Datatype</td><td>char[8]</td></tr>
<tr><td>Description</td><td>The ASCII representation of the instrument type.</td></tr>
<tr><td colspan="2">investor_type_s (Investor Type)</td></tr>
<tr><td>Datatype</td><td>char[4]</td></tr>
<tr><td>Description</td><td>Defines the investor type for the account.</td></tr>
<tr><td colspan="2">inv_scheme_c (Investment Scheme)</td></tr>
<tr><td>Datatype</td><td>CHAR</td></tr>
<tr><td>Description</td><td>Not applicable.</td></tr>
<tr><td>Value Set</td><td><table><tr><th>value</th><th>description</th></tr><tr><td>Blank</td><td>Not Applicable</td></tr></table></td></tr>
</table>

<table>
<tr><td colspan="2">isin_code_old_s (ISIN Code, Old Series)</td></tr>
<tr><td>Datatype</td><td>char[12]</td></tr>
<tr><td>Description</td><td>This is the old ISIN Code if a new code was assigned to the series after a recapitalization.</td></tr>
<tr><td colspan="2">isin_code_s (ISIN Code)</td></tr>
<tr><td>Datatype</td><td>char[12]</td></tr>
<tr><td>Description</td><td>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.</td></tr>
<tr><td colspan="2">issued_price_u (Issued Price)</td></tr>
<tr><td>Datatype</td><td>UINT32_T</td></tr>
<tr><td>Description</td><td>Defined the issued price for the underlying with three implicit decimals.</td></tr>
<tr><td colspan="2">is_exclusive_opening_sell_c (Exclusive Open Sell)</td></tr>
<tr><td>Datatype</td><td>UINT8_T</td></tr>
<tr><td>Description</td><td>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.</td></tr>
<tr><td>Value Set</td><td><table><tr><th>value</th><th>description</th></tr><tr><td>1</td><td>Yes</td></tr><tr><td>2</td><td>No</td></tr></table></td></tr>
<tr><td colspan="2">is_fractions_c (Fraction, Premium)</td></tr>
<tr><td>Datatype</td><td>CHAR</td></tr>
<tr><td>Description</td><td>Is the premium internally represented as fractions?</td></tr>
<tr><td>Value Set</td><td><table><tr><th>name</th><th>value</th></tr><tr><td>Yes</td><td>Y</td></tr><tr><td>No</td><td>N</td></tr></table></td></tr>
<tr><td colspan="2">items_block_n (Item, Block)</td></tr>
<tr><td>Datatype</td><td>UINT16_T</td></tr>
<tr><td>Description</td><td>Number of items.</td></tr>
<tr><td colspan="2">items_c (Item)</td></tr>
<tr><td>Datatype</td><td>UINT8_T</td></tr>
<tr><td>Description</td><td>Number of items.</td></tr>
<tr><td colspan="2">items_n (Items)</td></tr>
<tr><td>Datatype</td><td>UINT16_T</td></tr>
<tr><td>Description</td><td>Number of items.</td></tr>
</table>

<table>
<tr><td></td><td>This field used in a variable message counts the number of sub items provided in the variable message.</td></tr>
<tr><td colspan="2">ixv_id_s (IXV_ID_S)</td></tr>
<tr><td>Datatype</td><td>char[16]</td></tr>
<tr><td>Description</td><td>Index Value Id</td></tr>
<tr><td colspan="2">key_number_i (Key Number)</td></tr>
<tr><td>Datatype</td><td>INT32_T</td></tr>
<tr><td>Description</td><td>The key number within one delivery number.</td></tr>
<tr><td colspan="2">knock_variant_c (Knock Variant)</td></tr>
<tr><td>Datatype</td><td>UINT8_T</td></tr>
<tr><td>Description</td><td>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.</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>1</td><td>Down</td></tr>
<tr><td>2</td><td>Up</td></tr>
</table></td></tr>
<tr><td colspan="2">lag_in_index_n (LAG_IN_INDEX_N)</td></tr>
<tr><td>Datatype</td><td>UINT16_T</td></tr>
<tr><td>Description</td><td>Number of month the index is lagging</td></tr>
<tr><td colspan="2">last_paid_i (Last, Paid)</td></tr>
<tr><td>Datatype</td><td>INT32_T</td></tr>
<tr><td>Description</td><td>Last paid for the Instrument Series.</td></tr>
<tr><td colspan="2">last_theo_c (Last Paid, Theoretical Mark)</td></tr>
<tr><td>Datatype</td><td>UINT8_T</td></tr>
<tr><td>Description</td><td>Defines the origin of the price.</td></tr>
<tr><td>Value Set</td><td><table>
<tr><th>value</th><th>description</th></tr>
<tr><td>0</td><td>Missing</td></tr>
<tr><td>1</td><td>Theoretically calculated</td></tr>
<tr><td>2</td><td>From the order book</td></tr>
<tr><td>3</td><td>Manually updated</td></tr>
<tr><td>4</td><td>Artificial</td></tr>
</table></td></tr>
<tr><td colspan="2">lead_manager_country_id_s (Lead Manager, Country)</td></tr>
</table>

<table>
<tr><td>Datatype</td><td>char[2]</td></tr>
<tr><td>Description</td><td>The exchange identity that together with Lead Manager, Customer represents the lead manager.</td></tr>
<tr><td colspan="2">lead_manager_ex_customer_s (Lead Manager, Customer)</td></tr>
<tr><td>Datatype</td><td>char[5]</td></tr>
<tr><td>Description</td><td>This field together with Lead Manager, Country, identifies the member/participant that represents the lead manager.</td></tr>
<tr><td colspan="2">leg_number_c (Leg Number)</td></tr>
<tr><td>Datatype</td><td>UINT8_T</td></tr>
<tr><td>Description</td><td>Member or Party leg.</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>Member leg</td><td>1</td></tr>
<tr><td>Party leg</td><td>2</td></tr>
</table>
</td></tr>
<tr><td colspan="2">level_type_i (Level Type)</td></tr>
<tr><td>Datatype</td><td>INT32_T</td></tr>
<tr><td>Description</td><td>Position to be retrieved at what level?</td></tr>
<tr><td>Value Set</td><td>
<table>
<tr><th>value</th><th>description</th></tr>
<tr><td>1</td><td>Origin</td></tr>
<tr><td>2</td><td>Margin</td></tr>
</table>
</td></tr>
<tr><td colspan="2">le_state_c (Type, Legal Event)</td></tr>
<tr><td>Datatype</td><td>UINT8_T</td></tr>
<tr><td>Description</td><td>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:</td></tr>
<tr><td>Value Set</td><td>
<table>
<tr><th>name</th><th>value</th><th>description</th></tr>
<tr><td>None</td><td>0</td><td>None</td></tr>
<tr><td>holding</td><td>1</td><td>Holding
Object is holding and awaits countersign.</td></tr>
<tr><td>holding_indirectly</td><td>2</td><td>Holding Indirectly
Object is awaiting a holding object.</td></tr>
<tr><td>pending</td><td>3</td><td>Pending
Object is awaiting a later operation.</td></tr>
<tr><td>active</td><td>4</td><td>Active</td></tr>
</table>
</td></tr>
</table>

name	value	description
		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

linked_commodity_n (Linked Commodity Code)	
Datatype	UINT16_T
Description	If one or several underlying entries are linked together they are referenced to the real underlying by a pointer to the linked underlying code. If the underlyings are linked this code contains another Commodity Code distributed as another entry. 0 means that the underlyings are not linked.
list_name_s (Name, List)	
Datatype	char[40]
Description	List file name
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_ins_id_s (Series Name, Long)	

<table>
<tr><td>Datatype</td><td>char[32]</td></tr>
<tr><td>Description</td><td>Defines an additional instrument information to an instrument series.</td></tr>
<tr><td colspan="2">long_name (LONG_NAME)</td></tr>
<tr><td>Datatype</td><td>char[32]</td></tr>
<tr><td>Description</td><td>Intermediate field.</td></tr>
<tr><td colspan="2">long_underlying_id_s (Long Underlying Id)</td></tr>
<tr><td>Datatype</td><td>char[32]</td></tr>
<tr><td>Description</td><td>Specifies an additional the long name for the underlying.</td></tr>
<tr><td colspan="2">lot_type_c (Lot, Type)</td></tr>
<tr><td>Datatype</td><td>UINT8_T</td></tr>
<tr><td>Description</td><td>Specifies the lot type per block size.</td></tr>
<tr><td>Value Set</td><td><table>
<tr><th>value</th><th>description</th></tr>
<tr><td>1</td><td>Odd Lot</td></tr>
<tr><td>2</td><td>Round Lot</td></tr>
<tr><td>3</td><td>Block Lot</td></tr>
<tr><td>4</td><td>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.</td></tr>
</table></td></tr>
<tr><td colspan="2">lower_limit_i (Premium/Price, Low Limit)</td></tr>
<tr><td>Datatype</td><td>INT32_T</td></tr>
<tr><td>Description</td><td>The lower limit in the price interval.</td></tr>
<tr><td colspan="2">maintain_positions_c (Maintain Positions)</td></tr>
<tr><td>Datatype</td><td>UINT8_T</td></tr>
<tr><td>Description</td><td>Maintain positions?</td></tr>
<tr><td>Value Set</td><td><table>
<tr><th>value</th><th>description</th></tr>
<tr><td>1</td><td>Keep Position</td></tr>
<tr><td>2</td><td>No Keep Position</td></tr>
</table></td></tr>
<tr><td colspan="2">market_c (Market Code)</td></tr>
<tr><td>Datatype</td><td>UINT8_T</td></tr>
<tr><td>Description</td><td>Binary representation of the market. Unique together with COUNTRY_C.</td></tr>
<tr><td colspan="2">market_maker_c (Market Maker)</td></tr>
<tr><td>Datatype</td><td>UINT8_T</td></tr>
<tr><td>Description</td><td>Is the account a market maker account?</td></tr>
</table>

<table>
<tr><td>Value Set</td><td><table><tr><th>value</th><th>description</th></tr><tr><td>1</td><td>Yes</td></tr><tr><td>2</td><td>No</td></tr></table></td></tr>
<tr><td colspan="2">market_type_c (Market, Type)</td></tr>
<tr><td>Datatype</td><td>UINT8_T</td></tr>
<tr><td>Description</td><td>Defines the type of market.</td></tr>
<tr><td>Value Set</td><td><table><tr><th>value</th><th>description</th></tr><tr><td>0</td><td>Generic</td></tr><tr><td>1</td><td>Stock</td></tr><tr><td>2</td><td>Fixed Income</td></tr><tr><td>3</td><td>Currency</td></tr><tr><td>4</td><td>Power/Energy</td></tr><tr><td>5</td><td>Commodity</td></tr><tr><td>6</td><td>Payment</td></tr><tr><td>7</td><td>Index</td></tr><tr><td>8</td><td>General</td></tr></table></td></tr>
<tr><td colspan="2">mar_id_s (Market, Identity)</td></tr>
<tr><td>Datatype</td><td>char[5]</td></tr>
<tr><td>Description</td><td>The ASCII representation of the market.</td></tr>
<tr><td colspan="2">master_clh_id_s (Master CLH, Identity)</td></tr>
<tr><td>Datatype</td><td>char[12]</td></tr>
<tr><td>Description</td><td>The master clearinghouse for the exchange.</td></tr>
<tr><td colspan="2">match_group_nbr_u (Match group number, group inside an execution)</td></tr>
<tr><td>Datatype</td><td>UINT32_T</td></tr>
<tr><td>Description</td><td>A sequential number of an execution sequence number.</td></tr>
<tr><td colspan="2">match_item_nbr_u (Match Item Number)</td></tr>
<tr><td>Datatype</td><td>UINT32_T</td></tr>
<tr><td>Description</td><td>Match item number inside a match group number.</td></tr>
<tr><td colspan="2">maturity_c (Maturity)</td></tr>
<tr><td>Datatype</td><td>UINT8_T</td></tr>
<tr><td>Description</td><td>Defines if this an Instrument Group where corresponding Instrument Series has an Expiration Date defined.</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>
</table>

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.
mbs_id_s (Minimum Bid Schedule)	
Datatype	CHAR[2]
Description	Not applicable.
member_circ_numb_s (Member, Circular Number)	
Datatype	char[4]
Description	Not applicable.
member_net_open_interest_q (Net Open interest, Member)	
Datatype	UINT64_T
Description	Defines the member net open interest.
mic_code_s (MIC Code)	
Datatype	char[8]
Description	Specifies the MIC Code for the market.
minimum_size_n (Block Size, Minimum Volume)	
Datatype	UINT32_T
Description	The minimum volume required for the order per block size. Note! A value of 0 means no limit.
min_qty_increment_i (Minimum Quantity Increment)	
Datatype	INT32_T
Description	Not applicable.
min_show_vol_u (Order, Min Show Volume)	
Datatype	UINT32_T
Description	Minimum visible volume that must be specified in hidden orders.
modified_date_s (Date, Modified)	
Datatype	char[8]
Description	Date what the item was modified in ASCII. Format: YYYYMMDD.
modified_time_s (Time, Modified)	
Datatype	char[6]
Description	Defines what time the item was last changed. Format: HHMMSS.
modifier_c (Modifier)	
Datatype	UINT8_T
Description	Expiration date modifier. This value is set to zero when the instrument is new. The value is incremented by one each time the instrument is involved in an issue, split, etc. Note that the modifier value can be different for bid and ask options in the same Series.
money_or_par_c (Money or Par)	

<table>
<tr><td>Datatype</td><td>UINT8_T</td></tr>
<tr><td>Description</td><td>Money or Par filled repo</td></tr>
<tr><td>Value Set</td><td><table><tr><th>name</th><th>value</th></tr><tr><td>Money</td><td>1</td></tr><tr><td>Par</td><td>2</td></tr></table></td></tr>
<tr><td colspan="2">named_struct_n (Named Struct, Number)</td></tr>
<tr><td>Datatype</td><td>UINT16_T</td></tr>
<tr><td>Description</td><td>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.</td></tr>
<tr><td colspan="2">name_s (Name)</td></tr>
<tr><td>Datatype</td><td>char[32]</td></tr>
<tr><td>Description</td><td>The full ASCII representation.</td></tr>
<tr><td colspan="2">name_short (NAME_SHORT)</td></tr>
<tr><td>Datatype</td><td>char[10]</td></tr>
<tr><td>Description</td><td>intermediate field.</td></tr>
<tr><td colspan="2">nationality_s (Nationality)</td></tr>
<tr><td>Datatype</td><td>char[4]</td></tr>
<tr><td>Description</td><td>Defined the nationality for the account.</td></tr>
<tr><td colspan="2">nbr_held_q (Held)</td></tr>
<tr><td>Datatype</td><td>INT64_T</td></tr>
<tr><td>Description</td><td>Number of held (long) contracts</td></tr>
<tr><td colspan="2">nbr_written_q (Written)</td></tr>
<tr><td>Datatype</td><td>INT64_T</td></tr>
<tr><td>Description</td><td>Number of written (short) contracts</td></tr>
<tr><td colspan="2">net_open_interest_q (Net Open Interest)</td></tr>
<tr><td>Datatype</td><td>UINT64_T</td></tr>
<tr><td>Description</td><td>Defines the net open interest.</td></tr>
<tr><td colspan="2">new_commodity_n (Commodity Code, New)</td></tr>
<tr><td>Datatype</td><td>UINT16_T</td></tr>
<tr><td>Description</td><td>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.</td></tr>
<tr><td colspan="2">new_deal_price_i (Price, New Deal)</td></tr>
<tr><td>Datatype</td><td>INT32_T</td></tr>
<tr><td>Description</td><td>Defines the new deal price on a rectified deal..</td></tr>
<tr><td colspan="2">next_clearing_date_s (Clearing Date, Next)</td></tr>
<tr><td>Datatype</td><td>char[8]</td></tr>
</table>

<table>
<tr><td>Description</td><td>Date in ASCII for clearing trade, format is YYYYMMDD.</td></tr>
<tr><td colspan="2">next_planned_start_date_s (Planned Start Date, Next)</td></tr>
<tr><td>Datatype</td><td>char[8]</td></tr>
<tr><td>Description</td><td>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.</td></tr>
<tr><td colspan="2">next_planned_start_time_s (Planned Start Time, Next)</td></tr>
<tr><td>Datatype</td><td>char[6]</td></tr>
<tr><td>Description</td><td>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.</td></tr>
<tr><td colspan="2">nominal_value_q (Nominal Value)</td></tr>
<tr><td>Datatype</td><td>INT64_T</td></tr>
<tr><td>Description</td><td>Nominal value for the underlying.</td></tr>
<tr><td colspan="2">non_traded_ref_c (Non Traded Reference)</td></tr>
<tr><td>Datatype</td><td>UINT8_T</td></tr>
<tr><td>Description</td><td>Not applicable.</td></tr>
<tr><td>Value Set</td><td><table><tr><th>value</th><th>description</th></tr><tr><td>2</td><td>No</td></tr></table></td></tr>
<tr><td colspan="2">normal_clearing_days_n (Normal Clearing Days)</td></tr>
<tr><td>Datatype</td><td>UINT16_T</td></tr>
<tr><td>Description</td><td>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.</td></tr>
<tr><td colspan="2">normal_settl_days_n (Normal Settlement Days)</td></tr>
<tr><td>Datatype</td><td>UINT16_T</td></tr>
<tr><td>Description</td><td>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.</td></tr>
<tr><td colspan="2">normal_trading_days_n (Normal Trading Days)</td></tr>
<tr><td>Datatype</td><td>UINT16_T</td></tr>
<tr><td>Description</td><td>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.</td></tr>
<tr><td colspan="2">no_of_sub_n (Substitution, Max Number)</td></tr>
<tr><td>Datatype</td><td>UINT16_T</td></tr>
<tr><td>Description</td><td>Maximum allowed number of substitutions</td></tr>
<tr><td colspan="2">ntd_id_s (Non-trading Days, Identity)</td></tr>
</table>

<table>
<tr><td>Datatype</td><td>char[5]</td></tr>
<tr><td>Description</td><td>Defines the identity of holiday table.</td></tr>
<tr><td colspan="2">number_short (NUMBER_SHORT)</td></tr>
<tr><td>Datatype</td><td>UINT16_T</td></tr>
<tr><td>Description</td><td>Intermediate field.</td></tr>
<tr><td colspan="2">old_trade_c (Old Trade Indicator)</td></tr>
<tr><td>Datatype</td><td>UINT8_T</td></tr>
<tr><td>Description</td><td>Indicates whether the trade emanates from a deal cleared prior to the current clearing date.</td></tr>
<tr><td>Value Set</td><td><table>
<tr><th>value</th><th>description</th></tr>
<tr><td>1</td><td>Yes</td></tr>
<tr><td>2</td><td>No
Given up trade cleared today</td></tr>
</table></td></tr>
<tr><td colspan="2">omex_version_s (OMEX Version)</td></tr>
<tr><td>Datatype</td><td>char[16]</td></tr>
<tr><td>Description</td><td>This is the current Genium INET version running on the system.</td></tr>
<tr><td colspan="2">on_off_c (On or Off)</td></tr>
<tr><td>Datatype</td><td>UINT8_T</td></tr>
<tr><td>Description</td><td>Status field for Suspend, Resume.
Resume=On, Suspend=Off</td></tr>
<tr><td>Value Set</td><td><table>
<tr><th>value</th><th>description</th></tr>
<tr><td>1</td><td>On, keep orders</td></tr>
<tr><td>2</td><td>Off, remove orders</td></tr>
<tr><td>3</td><td>On, remove orders</td></tr>
<tr><td>4</td><td>Off, keep orders</td></tr>
</table></td></tr>
<tr><td colspan="2">open_close_c (Open or Closed)</td></tr>
<tr><td>Datatype</td><td>UINT8_T</td></tr>
<tr><td>Description</td><td>Defines the position update for the account. None if positions not maintained or not applicable for instrument.</td></tr>
<tr><td>Value Set</td><td><table>
<tr><th>value</th><th>description</th></tr>
<tr><td>0</td><td>None
No position update</td></tr>
<tr><td>1</td><td>Open</td></tr>
<tr><td>2</td><td>Closed</td></tr>
</table></td></tr>
<tr><td colspan="2">open_close_req_c (Open Close Request)</td></tr>
<tr><td>Datatype</td><td>UINT8_T</td></tr>
</table>

<table>
<tr><td>Description</td><td>Describes how the requested position account should be updated:</td></tr>
<tr><td>Value Set</td><td>
<table>
<tr><th>name</th><th>value</th><th>description</th></tr>
<tr><td>OPEN_CLOSE_REQ_DEFAULT</td><td>0</td><td>Default for the account</td></tr>
<tr><td>OPEN_CLOSE_REQ_OPEN</td><td>1</td><td>Open</td></tr>
<tr><td>OPEN_CLOSE_REQ_CLOSE</td><td>2</td><td>Close/net</td></tr>
<tr><td>OPEN_CLOSE_REQ_MND_CLOSE</td><td>3</td><td>Mandatory close</td></tr>
<tr><td>OPEN_CLOSE_REQ_RESET</td><td>4</td><td>Set to default to the account (valid only for alter order)</td></tr>
</table>
</td></tr>
<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">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>

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.

<table>
<tr><td></td><td>The value is defined as quantity*price*price quotation factor.</td></tr>
<tr><td colspan="2">order_number_u (Order Number)</td></tr>
<tr><td>Datatype</td><td>QUAD_WORD</td></tr>
<tr><td>Description</td><td>A unique identity for each order transaction.</td></tr>
<tr><td colspan="2">org_number_s (Organization number)</td></tr>
<tr><td>Datatype</td><td>char[16]</td></tr>
<tr><td>Description</td><td>Organization number for owner of account.</td></tr>
<tr><td colspan="2">original_date_s (Original Date)</td></tr>
<tr><td>Datatype</td><td>char[8]</td></tr>
<tr><td>Description</td><td>As of date for delivery. Format is YYYYMMDD</td></tr>
<tr><td colspan="2">original_delivery_number_i (Original, Delivery Number)</td></tr>
<tr><td>Datatype</td><td>INT32_T</td></tr>
<tr><td>Description</td><td>When not zero, it is used to point out another delivery together with fields Series and Original, Key Number.</td></tr>
<tr><td colspan="2">original_key_number_i (Original, Key Number)</td></tr>
<tr><td>Datatype</td><td>INT32_T</td></tr>
<tr><td>Description</td><td>When not zero, it is used to point out another delivery together with fields Series and Original, Delivery Number.</td></tr>
<tr><td colspan="2">originator_type_c (Originator Type)</td></tr>
<tr><td>Datatype</td><td>UINT8_T</td></tr>
<tr><td>Description</td><td>Defines the type of originator for the delivery.</td></tr>
<tr><td>Value Set</td><td><table>
<tr><th>value</th><th>description</th></tr>
<tr><td>1</td><td>Normal</td></tr>
<tr><td>2</td><td>Reversing
This delivery is created from a reversing trade</td></tr>
</table></td></tr>
<tr><td colspan="2">origin_c (Origin, Account Type)</td></tr>
<tr><td>Datatype</td><td>CHAR</td></tr>
<tr><td>Description</td><td>Defines how trading activites on accounts of the account type are to be classified.</td></tr>
<tr><td>Value Set</td><td><table>
<tr><th>name</th><th>value</th></tr>
<tr><td>House</td><td>H</td></tr>
<tr><td>Client</td><td>C</td></tr>
</table></td></tr>
<tr><td colspan="2">orig_clearing_date_s (Clearing Date, Original)</td></tr>
<tr><td>Datatype</td><td>char[8]</td></tr>
<tr><td>Description</td><td>The date the deal was originally cleared. Date in ASCII, format is YYYYMMDD</td></tr>
<tr><td colspan="2">orig_ext_trade_number_u (Trade Number, Original External)</td></tr>
<tr><td>Datatype</td><td>UINT32_T</td></tr>
</table>

<table>
<tr><td>Description</td><td>Original trade number assigned by external system.</td></tr>
<tr><td colspan="2">orig_trade_number_i (Trade Number, Original)</td></tr>
<tr><td>Datatype</td><td>INT32_T</td></tr>
<tr><td>Description</td><td>For an overtaking trade, this field references the original trade.</td></tr>
<tr><td colspan="2">orig_trade_type_c (Trade Type, Original)</td></tr>
<tr><td>Datatype</td><td>UINT8_T</td></tr>
<tr><td>Description</td><td>Defines the original trade type, for further description see Trade Type.</td></tr>
<tr><td colspan="2">outstanding_amount_q (Outstanding Amount)</td></tr>
<tr><td>Datatype</td><td>INT64_T</td></tr>
<tr><td>Description</td><td>The outstanding amount for the underlying.</td></tr>
<tr><td colspan="2">own_inventory_c (Own Inventory)</td></tr>
<tr><td>Datatype</td><td>UINT8_T</td></tr>
<tr><td>Description</td><td>Is the account an own inventory account?</td></tr>
<tr><td>Value Set</td><td><table><tr><th>value</th><th>description</th></tr><tr><td>1</td><td>Yes</td></tr><tr><td>2</td><td>No</td></tr></table></td></tr>
<tr><td colspan="2">passthrough_s (Passthrough Information)</td></tr>
<tr><td>Datatype</td><td>char[32]</td></tr>
<tr><td>Description</td><td>A reserved field for information sent from external sources to be passed through the clearing system without any processing or validation.</td></tr>
<tr><td colspan="2">payment_set_c (Payment Set)</td></tr>
<tr><td>Datatype</td><td>UINT8_T</td></tr>
<tr><td>Description</td><td>Decides if payment should occur in the beginning or in the end of a period.</td></tr>
<tr><td>Value Set</td><td><table><tr><th>name</th><th>value</th></tr><tr><td>First</td><td>1</td></tr><tr><td>Last</td><td>2</td></tr></table></td></tr>
<tr><td colspan="2">physical_delivery_c (Physical Delivery)</td></tr>
<tr><td>Datatype</td><td>UINT8_T</td></tr>
<tr><td>Description</td><td>Defines if this an Instrument Group where corresponding Instrument Series are physically delivered.</td></tr>
<tr><td>Value Set</td><td><table><tr><th>value</th><th>description</th></tr><tr><td>1</td><td>Yes</td></tr><tr><td>2</td><td>No</td></tr></table></td></tr>
<tr><td colspan="2">positions_allowed_c (Positions, Allowed)</td></tr>
<tr><td>Datatype</td><td>UINT8_T</td></tr>
</table>

<table>
<tr><td>Description</td><td>Is it allowed to hold positions on the account?</td></tr>
<tr><td>Value Set</td><td><table><tr><th>name</th><th>value</th></tr><tr><td>Yes</td><td>1</td></tr><tr><td>No</td><td>2</td></tr></table></td></tr>
<tr><td colspan="2">post_trade_proc_c (Post Trade processed)</td></tr>
<tr><td>Datatype</td><td>UINT8_T</td></tr>
<tr><td>Description</td><td>Specifies if instrument series connected to the instrument type is processed in the Clearing System.</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">pos_handling_c (Position handling)</td></tr>
<tr><td>Datatype</td><td>UINT8_T</td></tr>
<tr><td>Value Set</td><td><table><tr><th>name</th><th>value</th></tr><tr><td>No position keeping</td><td>1</td></tr><tr><td>Single session position keeping</td><td>2</td></tr><tr><td>Invariant dual session position keeping</td><td>3</td></tr><tr><td>Sequential dual session position keeping</td><td>4</td></tr></table></td></tr>
<tr><td colspan="2">pqf_modifier_c (Modifier, Price Quotation Factor)</td></tr>
<tr><td>Datatype</td><td>UINT8_T</td></tr>
<tr><td>Description</td><td>The modifier is used to recalculate the item after an underlying adjustment. The field is stored with 3 implicit decimals.</td></tr>
<tr><td>Value Set</td><td><table><tr><th>value</th><th>description</th></tr><tr><td>1</td><td>Modifier is added to the item</td></tr><tr><td>2</td><td>Modifier is subtracted from the item</td></tr><tr><td>3</td><td>Modifier is multiplied with the item</td></tr><tr><td>4</td><td>The item is divided by the modifier factor</td></tr></table></td></tr>
<tr><td colspan="2">pqf_mod_factor_i (Modifier Factor, Price Quotation Factor)</td></tr>
<tr><td>Datatype</td><td>INT32_T</td></tr>
<tr><td>Description</td><td>The modifier is used to recalculate the item after an underlying adjustment. The field is stored with 7 implicit decimals</td></tr>
<tr><td colspan="2">prev_clearing_date_s (Clearing Date, Previous)</td></tr>
<tr><td>Datatype</td><td>char[8]</td></tr>
<tr><td>Description</td><td>Date in ASCII for clearing trade, format is YYYYMMDD.</td></tr>
</table>

<table>
<tr><th colspan="2">price (PRICE)</th></tr>
<tr><td>Datatype</td><td>INT32_T</td></tr>
<tr><td>Description</td><td>Intermediate field.</td></tr>
<tr><th colspan="2">price_format_c (Premium/Price Format)</th></tr>
<tr><td>Datatype</td><td>UINT8_T</td></tr>
<tr><td>Description</td><td>Not applicable.</td></tr>
<tr><th colspan="2">price_quot_factor_i (Price, Quotation Factor)</th></tr>
<tr><td>Datatype</td><td>INT32_T</td></tr>
<tr><td>Description</td><td>Defines the price quotation factor used to calculate the trade price from the order.</td></tr>
<tr><th colspan="2">price_unit_c (Price Unit, Underlying)</th></tr>
<tr><td>Datatype</td><td>UINT8_T</td></tr>
<tr><td>Description</td><td>The price unit for the underlying can be one of the following:</td></tr>
<tr><td>Value Set</td><td><table>
<tr><th>value</th><th>description</th></tr>
<tr><td>1</td><td>Price</td></tr>
<tr><td>2</td><td>Yield</td></tr>
<tr><td>3</td><td>Points</td></tr>
<tr><td>4</td><td>Yield Diff</td></tr>
<tr><td>5</td><td>IMM Index</td></tr>
<tr><td>6</td><td>Basis Points</td></tr>
<tr><td>7</td><td>Inverted Yield</td></tr>
<tr><td>8</td><td>Percentage of Nominal</td></tr>
<tr><td>9</td><td>Dirty Price</td></tr>
</table></td></tr>
<tr><th colspan="2">price_unit_premium_c (Price Unit, Premium)</th></tr>
<tr><td>Datatype</td><td>UINT8_T</td></tr>
<tr><td>Description</td><td>The premium unit that describes the price unit in the order.</td></tr>
<tr><td>Value Set</td><td><table>
<tr><th>value</th><th>description</th></tr>
<tr><td>1</td><td>Price</td></tr>
<tr><td>2</td><td>Yield</td></tr>
<tr><td>3</td><td>Points</td></tr>
<tr><td>4</td><td>Yield Diff</td></tr>
<tr><td>5</td><td>IMM Index</td></tr>
<tr><td>6</td><td>Basis Points</td></tr>
<tr><td>7</td><td>Inverted Yield</td></tr>
<tr><td>8</td><td>Percentage of Nominal</td></tr>
<tr><td>9</td><td>Dirty Price</td></tr>
</table></td></tr>
</table>

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

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

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	
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

<table>
<tr><td></td><td><table><tr><th>value</th><th>description</th></tr><tr><td>4</td><td>Settlement</td></tr><tr><td>5</td><td>Origin</td></tr></table></td></tr>
<tr><td colspan="2">protect_coupon_c (PROTECT_COUPON_C)</td></tr>
<tr><td>Datatype</td><td>UINT8_T</td></tr>
<tr><td>Description</td><td>Protect index from beeing negative for coupons</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">protect_redempt_c (PROTECT_REDEMPT_C)</td></tr>
<tr><td>Datatype</td><td>UINT8_T</td></tr>
<tr><td>Description</td><td>Protect index from beeing negative for redempt.</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">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">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></table></td></tr>
</table>

<table>
<tr><td></td><td><table>
<tr><th>name</th><th>value</th><th>description</th></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">qty_closed_out_q (Quantity, Closed out)</td></tr>
<tr><td>Datatype</td><td>INT64_T</td></tr>
<tr><td>Description</td><td>Quantity closed out on position</td></tr>
<tr><td colspan="2">quantity_cover_u (Quantity Cover)</td></tr>
<tr><td>Datatype</td><td>UINT32_T</td></tr>
<tr><td>Description</td><td>Defines the number of underlying shares used as cover for a short position.</td></tr>
<tr><td colspan="2">quantity_i (Quantity)</td></tr>
<tr><td>Datatype</td><td>INT64_T</td></tr>
<tr><td>Description</td><td>Defines the quantity.</td></tr>
<tr><td colspan="2">query_on_date_c (Query on Date)</td></tr>
<tr><td>Datatype</td><td>UINT8_T</td></tr>
<tr><td>Description</td><td>Defines whether date is part of the search criteria.</td></tr>
<tr><td>Value Set</td><td><table>
<tr><th>value</th><th>description</th></tr>
<tr><td>0</td><td>No</td></tr>
<tr><td>1</td><td>Yes</td></tr>
</table></td></tr>
<tr><td colspan="2">rank_class_i (Risk Ranking Class)</td></tr>
<tr><td>Datatype</td><td>INT32_T</td></tr>
<tr><td>Description</td><td>The risk ranking class of an account or member.</td></tr>
<tr><td colspan="2">rate_determ_days_n (Rate Determination Days)</td></tr>
<tr><td>Datatype</td><td>UINT16_T</td></tr>
<tr><td>Description</td><td>Specifies number of rate determination days.</td></tr>
<tr><td colspan="2">rate_i (Rate)</td></tr>
<tr><td>Datatype</td><td>INT32_T</td></tr>
<tr><td>Description</td><td>Specifies the rate value for the reference rate and date. Given with 4 decimals.</td></tr>
<tr><td colspan="2">read_access_c (Read Access)</td></tr>
<tr><td>Datatype</td><td>UINT8_T</td></tr>
<tr><td>Description</td><td>Defines what type of data the owner of the account can read.</td></tr>
</table>

<table>
<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>Position</td></tr><tr><td>2</td><td>Trade</td></tr></table></td></tr>
<tr><td colspan="2">rectify_deal_number_q (Rectify Deal Number)</td></tr>
<tr><td>Datatype</td><td>INT64_T</td></tr>
<tr><td>Description</td><td>A number that together with series identifies a specific rectified deal.</td></tr>
<tr><td colspan="2">rectify_trade_number_i (Rectify Trade Number)</td></tr>
<tr><td>Datatype</td><td>INT32_T</td></tr>
<tr><td>Description</td><td>A number that together with series identifies a specific rectified trade.</td></tr>
<tr><td colspan="2">redemption_value_i (Redemption Value)</td></tr>
<tr><td>Datatype</td><td>INT32_T</td></tr>
<tr><td>Description</td><td>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.</td></tr>
<tr><td colspan="2">rem_quantity_i (Quantity, Remaining)</td></tr>
<tr><td>Datatype</td><td>INT64_T</td></tr>
<tr><td>Description</td><td>Number of contracts, etc. Depending of instrument type.
It reflects:
Quantity still to be transferred from a transitory trade, for example, if a buy trade is created with quantity 25 on a transitory account, then rem_quantity_i will contain 25, as this quantity is still remaining to be moved to a position account.
Quantity still to be exercised for trade with an instrument type that has trade exercise ability, for example if a trade is created with quantity 25 on a option series then rem_quantity_i will contain 25, as this quantity is still remaining to be exercised.</td></tr>
<tr><td colspan="2">report_owner_s (Report owner)</td></tr>
<tr><td>Datatype</td><td>char[12]</td></tr>
<tr><td>Description</td><td>Name of member or customer that is the owner of the report.</td></tr>
<tr><td colspan="2">report_version_s (Report Version)</td></tr>
<tr><td>Datatype</td><td>char[3]</td></tr>
<tr><td>Description</td><td>Zero padded sequence number of the report.</td></tr>
<tr><td colspan="2">repo_type_c (Repo Type)</td></tr>
<tr><td>Datatype</td><td>UINT8_T</td></tr>
<tr><td>Description</td><td>Defines the type of the REPO.</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>1</td><td>GC</td></tr></table></td></tr>
</table>

<table>
<tr><td></td><td><table><tr><th>value</th><th>description</th></tr><tr><td>2</td><td>GCF</td></tr><tr><td>3</td><td>Special</td></tr><tr><td>4</td><td>Security Lending</td></tr><tr><td>5</td><td>IR Swap</td></tr></table></td></tr>
<tr><td colspan="2">reserved_1_c (Reserved)</td></tr>
<tr><td>Datatype</td><td>CHAR</td></tr>
<tr><td>Description</td><td>Filler for alignment.</td></tr>
<tr><td colspan="2">reserved_2_s (Reserved)</td></tr>
<tr><td>Datatype</td><td>char[2]</td></tr>
<tr><td>Description</td><td>Filler for alignment.</td></tr>
<tr><td colspan="2">reserved_prop_c (Reserved Properties)</td></tr>
<tr><td>Datatype</td><td>UINT8_T</td></tr>
<tr><td>Description</td><td>Generic bit mask flag dependant on the specific configuration or installation.</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>Anonymized</td><td>1</td></tr></table></td></tr>
<tr><td colspan="2">reset_days_c (Reset Days)</td></tr>
<tr><td>Datatype</td><td>UINT8_T</td></tr>
<tr><td>Description</td><td>Specifies the number of reset days to use for a leg</td></tr>
<tr><td colspan="2">reset_days_type_c (Reset days type)</td></tr>
<tr><td>Datatype</td><td>UINT8_T</td></tr>
<tr><td>Description</td><td>The day type for the Reset Days.
The business day convention is always following for the reset days.</td></tr>
<tr><td>Value Set</td><td><table><tr><th>name</th><th>value</th></tr><tr><td>Trading Days</td><td>1</td></tr><tr><td>Calendar Days</td><td>2</td></tr></table></td></tr>
<tr><td colspan="2">residual_i (Residual)</td></tr>
<tr><td>Datatype</td><td>INT32_T</td></tr>
<tr><td>Description</td><td>Residual due to rounding in average price trade.</td></tr>
<tr><td colspan="2">risk_currency_s (Currency, Risk)</td></tr>
<tr><td>Datatype</td><td>char[3]</td></tr>
<tr><td>Description</td><td>Currency after currency conversion.</td></tr>
<tr><td colspan="2">risk_cur_conv_c (Risk, Currency Conversion)</td></tr>
</table>

<table>
<tr><td>Datatype</td><td>UINT8_T</td></tr>
<tr><td>Description</td><td>Condition for currency conversion for margin requirements.</td></tr>
<tr><td>Value Set</td><td><table>
<tr><th>value</th><th>description</th></tr>
<tr><td>0</td><td>Default</td></tr>
<tr><td>1</td><td>Only Positive
Only convert margin gains to risk currency</td></tr>
<tr><td>2</td><td>Always
Always convert margin to risk currency</td></tr>
<tr><td>3</td><td>None
Do not convert margin to risk currency</td></tr>
</table></td></tr>
<tr><td colspan="2">risk_margin_net_c (Risk, Margin Net)</td></tr>
<tr><td>Datatype</td><td>UINT8_T</td></tr>
<tr><td>Description</td><td>Net margin requirements between markets.</td></tr>
<tr><td>Value Set</td><td><table>
<tr><th>value</th><th>description</th></tr>
<tr><td>1</td><td>Do not Net</td></tr>
<tr><td>2</td><td>Net</td></tr>
</table></td></tr>
<tr><td colspan="2">rnt_id_n (Ranking Type)</td></tr>
<tr><td>Datatype</td><td>UINT16_T</td></tr>
<tr><td>Description</td><td>This identifies how the instrument is ranked.</td></tr>
<tr><td>Value Set</td><td><table>
<tr><th>value</th><th>description</th></tr>
<tr><td>1</td><td>Rule 1
1. Price
2. Time</td></tr>
<tr><td>2</td><td>Rule 2
1. Inverted Price
2. Time</td></tr>
<tr><td>3</td><td>Rule 3
1. Price
2. Traders before MM
3. Time</td></tr>
<tr><td>4</td><td>Rule 4
1. Inverted Price
2. Traders before MM
3. Time</td></tr>
<tr><td>5</td><td>Rule 5
1. Price
2. MM before Traders</td></tr>
</table></td></tr>
</table>

<table>
<tr><td></td><td>
<table>
<tr><th>value</th><th>description</th></tr>
<tr><td></td><td>3. Time</td></tr>
<tr><td>6</td><td>Rule 6
1. Inverted Price
2. MM before Traders
3. Time</td></tr>
<tr><td>7</td><td>Rule 7
1. Price
2. Baits before Normal Orders
3. Time</td></tr>
<tr><td>8</td><td>Rule 8
1. Inverted Price
2. Baits before Normal Orders
3. Time</td></tr>
<tr><td>11</td><td>Rule 11
1. Price
2. Own Orders
3. Time</td></tr>
<tr><td>12</td><td>Rule 12
1. Inverted Price
2. Own Orders
3. Time</td></tr>
</table>
</td></tr>
<tr><td colspan="2">rollover_period_c (Rollover Period)</td></tr>
<tr><td>Datatype</td><td>UINT8_T</td></tr>
<tr><td>Description</td><td>Length of the rollover period</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>1 Month</td><td>1</td></tr>
<tr><td>3 Month</td><td>3</td></tr>
<tr><td>6 Month</td><td>6</td></tr>
<tr><td>12 Month</td><td>12</td></tr>
<tr><td>1 Week</td><td>21</td></tr>
</table>
</td></tr>
<tr><td colspan="2">rounding_before_index_c (Rounding before index)</td></tr>
<tr><td>Datatype</td><td>UINT8_T</td></tr>
<tr><td>Description</td><td>Specifies if the rounding of the price is done before the index value is multiplied with the price.</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>
</table>
</td></tr>
</table>

<table>
<tr><td></td><td><table><tr><th>name</th><th>value</th></tr><tr><td>No</td><td>2</td></tr></table></td></tr>
<tr><td colspan="2">rqst_type_i (RQST_TYPE_I)</td></tr>
<tr><td>Datatype</td><td>INT32_T</td></tr>
<tr><td>Value Set</td><td><table><tr><th>name</th><th>value</th></tr><tr><td>Exercise by exception</td><td>2</td></tr></table></td></tr>
<tr><td colspan="2">seconds_to_state_change_n (State Change, Seconds)</td></tr>
<tr><td>Datatype</td><td>UINT16_T</td></tr>
<tr><td>Description</td><td>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</td></tr>
<tr><td colspan="2">sector_code_s (Sector Code)</td></tr>
<tr><td>Datatype</td><td>char[4]</td></tr>
<tr><td>Description</td><td>The sector code that the underlying is connected to.</td></tr>
<tr><td colspan="2">segment_number_n (Segment Number)</td></tr>
<tr><td>Datatype</td><td>UINT16_T</td></tr>
<tr><td>Description</td><td>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.</td></tr>
<tr><td colspan="2">send_or_receive_c (Send or Receive)</td></tr>
<tr><td>Datatype</td><td>UINT8_T</td></tr>
<tr><td>Description</td><td>Indicates if a commission rule should be used while sending or receiving a give-up.</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>Send</td></tr><tr><td>2</td><td>Receive</td></tr></table></td></tr>
<tr><td colspan="2">sent_date_s (Date, Sent)</td></tr>
<tr><td>Datatype</td><td>char[8]</td></tr>
<tr><td>Description</td><td>Defines the sent date. Format: YYYYMMDD.</td></tr>
<tr><td colspan="2">sent_time_s (Time, Sent)</td></tr>
<tr><td>Datatype</td><td>char[6]</td></tr>
<tr><td>Description</td><td>Defines the sent time. Format: HHMMSS</td></tr>
<tr><td colspan="2">sequence (SEQUENCE)</td></tr>
</table>

<table>
<tr><td>Datatype</td><td>INT32_T</td></tr>
<tr><td>Description</td><td>intermediate field.</td></tr>
<tr><td colspan="2">sequence_first_i (Number, First Sequential)</td></tr>
<tr><td>Datatype</td><td>INT32_T</td></tr>
<tr><td>Description</td><td>First number in a sequence.</td></tr>
<tr><td colspan="2">sequence_last_i (Number, Last Sequential)</td></tr>
<tr><td>Datatype</td><td>INT32_T</td></tr>
<tr><td>Description</td><td>Last number in a sequence.</td></tr>
<tr><td colspan="2">sequence_number_i (Sequence Number)</td></tr>
<tr><td>Datatype</td><td>INT32_T</td></tr>
<tr><td>Description</td><td>Define a sequence number.</td></tr>
<tr><td colspan="2">seq_num_srm_n (Sequence number for SRM)</td></tr>
<tr><td>Datatype</td><td>UINT16_T</td></tr>
<tr><td>Description</td><td>An unique sequence number used by SRM</td></tr>
<tr><td colspan="2">series_id_s (Series, Identity)</td></tr>
<tr><td>Datatype</td><td>char[32]</td></tr>
<tr><td>Description</td><td>Instrument Series name is ASCII.</td></tr>
<tr><td colspan="2">series_sequence_number_u (Series, Sequence Number)</td></tr>
<tr><td>Datatype</td><td>UINT32_T</td></tr>
<tr><td>Description</td><td>Not applicable.</td></tr>
<tr><td colspan="2">series_status_c (Series, Status)</td></tr>
<tr><td>Datatype</td><td>UINT8_T</td></tr>
<tr><td>Description</td><td>The actual status of the series:</td></tr>
<tr><td>Value Set</td><td><table>
<tr><th>value</th><th>description</th></tr>
<tr><td>1</td><td>Active (both expired and not expired)</td></tr>
<tr><td>2</td><td>Suspended (temporarily stopped)</td></tr>
<tr><td>3</td><td>Issued</td></tr>
<tr><td>4</td><td>Delisted</td></tr>
</table></td></tr>
<tr><td colspan="2">server_type_c (Server Type)</td></tr>
<tr><td>Datatype</td><td>CHAR</td></tr>
<tr><td>Description</td><td>The server type at the central Exchange. Different target servers exist for different tasks.
The values below are only examples.</td></tr>
<tr><td>Value Set</td><td><table>
<tr><th>value</th><th>description</th></tr>
<tr><td>O</td><td>Order</td></tr>
<tr><td>Q</td><td>Query</td></tr>
<tr><td>D</td><td>Deal</td></tr>
</table></td></tr>
</table>

<table>
<tr><td></td><td><table><tr><th>value</th><th>description</th></tr><tr><td>A</td><td>Answer (only from the Central System)</td></tr><tr><td>I</td><td>Information</td></tr></table></td></tr>
<tr><td colspan="2">settlement_date_s (Date, Settlement)</td></tr>
<tr><td>Datatype</td><td>char[8]</td></tr>
<tr><td>Description</td><td>Settlement date for delivery or payment. Format YYYYMMDD.</td></tr>
<tr><td colspan="2">settlement_days_n (Settlement, Days or Month)</td></tr>
<tr><td>Datatype</td><td>UINT16_T</td></tr>
<tr><td>Description</td><td>Number of settlement days (or month) calculation rule.</td></tr>
<tr><td colspan="2">settlement_instr_date_s (Date, Settlement instruction)</td></tr>
<tr><td>Datatype</td><td>char[8]</td></tr>
<tr><td>Description</td><td>Date for generating instructions for settlement in
external settlement systems. Format: YYYYMMDD.</td></tr>
<tr><td colspan="2">settl_cur_id_s (Currency, Settlement)</td></tr>
<tr><td>Datatype</td><td>char[32]</td></tr>
<tr><td>Description</td><td>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.</td></tr>
<tr><td colspan="2">settl_day_unit_c (Settlement Day Unit)</td></tr>
<tr><td>Datatype</td><td>UINT8_T</td></tr>
<tr><td>Description</td><td>Describes the unit of the number of Settlement Days Rule for the instrument class</td></tr>
<tr><td>Value Set</td><td><table><tr><th>name</th><th>value</th></tr><tr><td>Not applicable</td><td>0</td></tr><tr><td>Days</td><td>1</td></tr><tr><td>Month</td><td>2</td></tr></table></td></tr>
<tr><td colspan="2">set_end_consid_c (Set End Consideration)</td></tr>
<tr><td>Datatype</td><td>UINT8_T</td></tr>
<tr><td>Description</td><td>End Consideration</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">set_start_consid_c (Calculate Settlement Amount)</td></tr>
<tr><td>Datatype</td><td>UINT8_T</td></tr>
<tr><td>Description</td><td>Specifies if settlement amount should be calculated in the post trade message.</td></tr>
</table>

<table>
<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">short_code (SHORT_CODE)</td></tr>
<tr><td>Datatype</td><td>CHAR</td></tr>
<tr><td>Description</td><td>Intermediate field.</td></tr>
<tr><td colspan="2">size_n (Size)</td></tr>
<tr><td>Datatype</td><td>UINT16_T</td></tr>
<tr><td>Description</td><td>Size of following struct including header where size resides.</td></tr>
<tr><td colspan="2">so_commodity_n (Commodity code, Spin Off)</td></tr>
<tr><td>Datatype</td><td>UINT16_T</td></tr>
<tr><td>Description</td><td>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.</td></tr>
<tr><td colspan="2">so_contract_size_modifier_c (Modifier, Contract Size)</td></tr>
<tr><td>Datatype</td><td>UINT8_T</td></tr>
<tr><td>Description</td><td>The modifier is used to recalculate the item after an underlying adjustment. The field is stored with 3 implicit decimals.</td></tr>
<tr><td>Value Set</td><td><table><tr><th>value</th><th>description</th></tr><tr><td>1</td><td>Modifier is added to the item</td></tr><tr><td>2</td><td>Modifier is subtracted from the item</td></tr><tr><td>3</td><td>Modifier is multiplied with the item</td></tr><tr><td>4</td><td>The item is divided by the modifier factor</td></tr></table></td></tr>
<tr><td colspan="2">so_contr_size_mod_factor_i (Modifier Factor, Spin Off Contract Size)</td></tr>
<tr><td>Datatype</td><td>INT32_T</td></tr>
<tr><td>Description</td><td>The modifier is used to recalculate the item after an underlying adjustment. The field is stored with 5 implicit decimals.</td></tr>
<tr><td colspan="2">so_country_c (Market, Spin Off)</td></tr>
<tr><td>Datatype</td><td>UINT8_T</td></tr>
<tr><td>Description</td><td>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.</td></tr>
<tr><td colspan="2">so_deal_price_modifier_c (Modifier, Spin Off Deal Price)</td></tr>
<tr><td>Datatype</td><td>UINT8_T</td></tr>
<tr><td>Description</td><td>The modifier is used to recalculate the item after an underlying adjustment. The field is stored with 3 implicit decimals.</td></tr>
<tr><td>Value Set</td><td><table><tr><th>value</th><th>description</th></tr><tr><td>1</td><td>Modifier is added to the item</td></tr></table></td></tr>
</table>

<table>
<tr><td></td><td><table><tr><th>value</th><th>description</th></tr><tr><td>2</td><td>Modifier is subtracted from the item</td></tr><tr><td>3</td><td>Modifier is multiplied with the item</td></tr><tr><td>4</td><td>The item is divided by the modifier factor</td></tr></table></td></tr>
<tr><td colspan="2">so_deal_price_mod_factor_i (Modifier Factor, Spin Off Deal Price)</td></tr>
<tr><td>Datatype</td><td>INT32_T</td></tr>
<tr><td>Description</td><td>The modifier is used to recalculate the item after an underlying adjustment. The field is stored with 7 implicit decimals</td></tr>
<tr><td colspan="2">so_market_c (Market, Spin Off)</td></tr>
<tr><td>Datatype</td><td>UINT8_T</td></tr>
<tr><td>Description</td><td>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.</td></tr>
<tr><td colspan="2">so_pqf_modifier_c (Modifier, Spin Off Price Quotation Factor)</td></tr>
<tr><td>Datatype</td><td>UINT8_T</td></tr>
<tr><td>Description</td><td>The modifier is used to recalculate the item after an underlying adjustment. The field is stored with 3 implicit decimals.</td></tr>
<tr><td>Value Set</td><td><table><tr><th>value</th><th>description</th></tr><tr><td>1</td><td>Modifier is added to the item</td></tr><tr><td>2</td><td>Modifier is subtracted from the item</td></tr><tr><td>3</td><td>Modifier is multiplied with the item</td></tr><tr><td>4</td><td>The item is divided by the modifier factor</td></tr></table></td></tr>
<tr><td colspan="2">so_pqf_mod_factor_i (Modifier Factor, Spin Off Price Quotation Factor)</td></tr>
<tr><td>Datatype</td><td>INT32_T</td></tr>
<tr><td>Description</td><td>The modifier is used to recalculate the item after an underlying adjustment. The field is stored with 7 implicit decimals</td></tr>
<tr><td colspan="2">so_strike_price_modifier_c (Modifier, Spin Off Strike Price)</td></tr>
<tr><td>Datatype</td><td>UINT8_T</td></tr>
<tr><td>Description</td><td>The modifier is used to recalculate the item after an underlying adjustment. The field is stored with 3 implicit decimals.</td></tr>
<tr><td>Value Set</td><td><table><tr><th>value</th><th>description</th></tr><tr><td>1</td><td>Modifier is added to the item</td></tr><tr><td>2</td><td>Modifier is subtracted from the item</td></tr><tr><td>3</td><td>Modifier is multiplied with the item</td></tr><tr><td>4</td><td>The item is divided by the modifier factor</td></tr></table></td></tr>
<tr><td colspan="2">so_strike_price_mod_factor_i (Modifier Factor, Spin Off Strike Price)</td></tr>
<tr><td>Datatype</td><td>INT32_T</td></tr>
</table>

<table>
<tr><td>Description</td><td>The modifier is used to recalculate the item after an underlying adjustment. The field is stored with 7 implicit decimals</td></tr>
<tr><td colspan="2">spinoff_c (Spinoff)</td></tr>
<tr><td>Datatype</td><td>UINT8_T</td></tr>
<tr><td>Description</td><td>Is the actual adjustment containing also Spin off series?</td></tr>
<tr><td>Value Set</td><td>
<table>
<tr><th>value</th><th>description</th></tr>
<tr><td>1</td><td>Yes</td></tr>
<tr><td>2</td><td>No</td></tr>
</table>
</td></tr>
<tr><td colspan="2">start_date_s (Date, Start)</td></tr>
<tr><td>Datatype</td><td>char[8]</td></tr>
<tr><td>Description</td><td>Start date. Format: YYYYMMDD.</td></tr>
<tr><td colspan="2">state_c (State)</td></tr>
<tr><td>Datatype</td><td>UINT8_T</td></tr>
<tr><td>Description</td><td>Defines the state of a request.</td></tr>
<tr><td>Value Set</td><td>
<table>
<tr><th>name</th><th>value</th><th>description</th></tr>
<tr><td>None</td><td>0</td><td>None</td></tr>
<tr><td>holding</td><td>1</td><td>Holding
Object is holding and awaits countersign.</td></tr>
<tr><td>holding_indirectly</td><td>2</td><td>Holding Indirectly
Object is awaiting a holding object.</td></tr>
<tr><td>pending</td><td>3</td><td>Pending
Object is awaiting a later operation.</td></tr>
<tr><td>active</td><td>4</td><td>Active
Object has been confirmed, if it was originally holding.</td></tr>
<tr><td>completed</td><td>5</td><td>Completed
A pending object has been completed.</td></tr>
<tr><td>rejected</td><td>6</td><td>Rejected
Object has been rejected.</td></tr>
<tr><td>business_completed</td><td>7</td><td>Business Completed
Realtime events done. This value is logically between Active and Completed.</td></tr>
<tr><td>delivered</td><td>8</td><td>Delivered
Object has been completed due to delivery.</td></tr>
<tr><td>rectified</td><td>9</td><td>Rectified</td></tr>
</table>
</td></tr>
</table>

name	value	description
deleted	10	Deleted
pending_rectify	11	Pending Rectify
expired	12	Expired
pending_authorize	13	Pending Authorize
delete_holding	14	Delete Holding Object is holding for delete and awaits countersign.

state_i (State, Product)	
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_level_e (Level)	
Datatype	UINT16_T
Description	Indicates the level which a state applies to:
Value Set	

value	description
0	All_Levels
1	Market
2	Instrument_Type
3	Instrument_Class
4	Instrument_Series
5	Underlying

state_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.

<table>
<tr><th colspan="2">step_size_i (Tick Size)</th></tr>
<tr><td>Datatype</td><td>INT32_T</td></tr>
<tr><td>Description</td><td>The tick size is the minimum valid step in the Premium or Price.</td></tr>
<tr><th colspan="2">step_size_multiple_n (Tick Size, Multiple)</th></tr>
<tr><td>Datatype</td><td>UINT16_T</td></tr>
<tr><td>Description</td><td>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.</td></tr>
<tr><th colspan="2">stock_code_s (Stock Code)</th></tr>
<tr><td>Datatype</td><td>char[6]</td></tr>
<tr><td>Description</td><td>Not applicable.</td></tr>
<tr><th colspan="2">stopped_by_issue_c (Stopped By Issue)</th></tr>
<tr><td>Datatype</td><td>UINT8_T</td></tr>
<tr><td>Description</td><td>The series is stopped from trading depending on an issue.</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><th colspan="2">strike_price_format_c (Strike Price, Format)</th></tr>
<tr><td>Datatype</td><td>UINT8_T</td></tr>
<tr><td>Description</td><td>Not applicable.</td></tr>
<tr><th colspan="2">strike_price_i (Strike Price)</th></tr>
<tr><td>Datatype</td><td>INT32_T</td></tr>
<tr><td>Description</td><td>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.</td></tr>
<tr><th colspan="2">strike_price_modifier_c (Modifier, Strike Price)</th></tr>
<tr><td>Datatype</td><td>UINT8_T</td></tr>
<tr><td>Description</td><td>The modifier is used to recalculate the item after an underlying adjustment. The field is stored with 3 implicit decimals.</td></tr>
<tr><td>Value Set</td><td><table>
<tr><th>value</th><th>description</th></tr>
<tr><td>1</td><td>Modifier is added to the item</td></tr>
<tr><td>2</td><td>Modifier is subtracted from the item</td></tr>
<tr><td>3</td><td>Modifier is multiplied with the item</td></tr>
<tr><td>4</td><td>The item is divided by the modifier factor</td></tr>
</table></td></tr>
<tr><th colspan="2">strike_price_mod_factor_i (Modifier Factor, Strike Price)</th></tr>
<tr><td>Datatype</td><td>INT32_T</td></tr>
</table>

<table>
<tr><td>Description</td><td>The modifier is used to recalculate the item after an underlying adjustment. The field is stored with 7 implicit decimals.</td></tr>
<tr><td colspan="2">subscription_price_i (Subscription, Price)</td></tr>
<tr><td>Datatype</td><td>INT32_T</td></tr>
<tr><td>Description</td><td>Not applicable.</td></tr>
<tr><td colspan="2">sub_fix_income_type_s (Sub Fixed Income Type)</td></tr>
<tr><td>Datatype</td><td>char[32]</td></tr>
<tr><td>Description</td><td>Defines any additional categorization of the Underlying, e.g. Callable or Putable.</td></tr>
<tr><td colspan="2">summary_i (Summary)</td></tr>
<tr><td>Datatype</td><td>INT32_T</td></tr>
<tr><td>Description</td><td>Defines whether or not to aggregate positions by the account level selected.</td></tr>
<tr><td>Value Set</td><td><table><tr><th>value</th><th>description</th></tr><tr><td>1</td><td>Yes</td></tr><tr><td>2</td><td>No</td></tr></table></td></tr>
<tr><td colspan="2">suspended_c (Suspended)</td></tr>
<tr><td>Datatype</td><td>UINT8_T</td></tr>
<tr><td>Description</td><td>Defines if the series is suspended or not.</td></tr>
<tr><td>Value Set</td><td><table><tr><th>value</th><th>description</th></tr><tr><td>1</td><td>Yes</td></tr><tr><td>2</td><td>No</td></tr></table></td></tr>
<tr><td colspan="2">swap_style_c (Style, Swap)</td></tr>
<tr><td>Datatype</td><td>UINT8_T</td></tr>
<tr><td>Description</td><td>Defines if this an Instrument Group where corresponding Instrument Series are swap styled.</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>1</td><td>Fixed-Fixed</td></tr><tr><td>2</td><td>Fixed-Float</td></tr><tr><td>3</td><td>Float-Float</td></tr><tr><td>4</td><td>TOM next</td></tr><tr><td>5</td><td>Generic</td></tr></table></td></tr>
<tr><td colspan="2">synthetic_type_c (Type, Synthetic)</td></tr>
<tr><td>Datatype</td><td>UINT8_T</td></tr>
<tr><td>Description</td><td>Not Applicable.</td></tr>
</table>

<table>
<tr><td>Value Set</td><td><table><tr><th>value</th><th>description</th></tr><tr><td>0</td><td>Not applicable</td></tr></table></td></tr>
<tr><td colspan="2">tailor_made_c (Tailor Made)</td></tr>
<tr><td>Datatype</td><td>UINT8_T</td></tr>
<tr><td>Description</td><td>Is the instrument group used for tailor made created series:</td></tr>
<tr><td>Value Set</td><td><table><tr><th>value</th><th>description</th></tr><tr><td>1</td><td>Yes</td></tr><tr><td>2</td><td>No</td></tr></table></td></tr>
<tr><td colspan="2">term_code_s (TERM_CODE_S)</td></tr>
<tr><td>Datatype</td><td>char[12]</td></tr>
<tr><td>Description</td><td>Term Code desc. for REPO instruments</td></tr>
<tr><td colspan="2">text_buffer_s (Text, Buffer)</td></tr>
<tr><td>Datatype</td><td>char[50000]</td></tr>
<tr><td>Description</td><td>The text buffer contains text records with an uint16 followed by the text line. The records are word aligned in the text buffer.</td></tr>
<tr><td colspan="2">text_id (TEXT_ID)</td></tr>
<tr><td>Datatype</td><td>char[12]</td></tr>
<tr><td>Description</td><td>Intermediate field.</td></tr>
<tr><td colspan="2">text_line (TEXT_LINE)</td></tr>
<tr><td>Datatype</td><td>char[80]</td></tr>
<tr><td>Description</td><td>intermediate field.</td></tr>
<tr><td colspan="2">text_line_s (Text, Line)</td></tr>
<tr><td>Datatype</td><td>char[80]</td></tr>
<tr><td>Description</td><td>One line of text information.</td></tr>
<tr><td colspan="2">time_delivery_start_s (Time, Delivery Start)</td></tr>
<tr><td>Datatype</td><td>char[6]</td></tr>
<tr><td>Description</td><td>Delivery start time. Format: HHMMSS.</td></tr>
<tr><td colspan="2">time_delivery_stop_s (Time, Delivery Stop)</td></tr>
<tr><td>Datatype</td><td>char[6]</td></tr>
<tr><td>Description</td><td>Delivery stop time. Format: HHMMSS.</td></tr>
<tr><td colspan="2">time_first_trading_s (Time, First Trading)</td></tr>
<tr><td>Datatype</td><td>char[6]</td></tr>
<tr><td>Description</td><td>The first valid trading time of the series. The time is together with DATE, FIRST TRADING distributed as UTC.
Time in ASCII, format is HHMMSS.</td></tr>
<tr><td colspan="2">time_last_trading_s (Time, Last Trading)</td></tr>
</table>

<table>
<tr><td>Datatype</td><td>char[6]</td></tr>
<tr><td>Description</td><td>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.</td></tr>
<tr><td colspan="2">time_of_agree_gran_c (Time of agreement granularity)</td></tr>
<tr><td>Datatype</td><td>UINT8_T</td></tr>
<tr><td>Description</td><td>Specifies if the time of agreement contains date or both date and time.</td></tr>
<tr><td>Value Set</td><td><table>
<tr><th>name</th><th>value</th></tr>
<tr><td>Not applicable</td><td>0</td></tr>
<tr><td>Date</td><td>1</td></tr>
<tr><td>Date and Time</td><td>2</td></tr>
</table></td></tr>
<tr><td colspan="2">time_of_agree_req_c (Time of agreement required)</td></tr>
<tr><td>Datatype</td><td>UINT8_T</td></tr>
<tr><td>Description</td><td>Specifies how time of agreement is specified and validated in the trade report.</td></tr>
<tr><td>Value Set</td><td><table>
<tr><th>name</th><th>value</th></tr>
<tr><td>Not required</td><td>0</td></tr>
<tr><td>On first reported</td><td>1</td></tr>
<tr><td>On both sides - not matched</td><td>2</td></tr>
<tr><td>On both sides - must match</td><td>3</td></tr>
<tr><td>On both sides - must match on date</td><td>4</td></tr>
</table></td></tr>
<tr><td colspan="2">tm_series_c (Tailor Made Series)</td></tr>
<tr><td>Datatype</td><td>UINT8_T</td></tr>
<tr><td>Description</td><td>Not applicable.</td></tr>
<tr><td colspan="2">tm_template_c (Template Series)</td></tr>
<tr><td>Datatype</td><td>UINT8_T</td></tr>
<tr><td>Description</td><td>Defines if this a template series.</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">total_ex_day_i (TOTAL_EX_DAY_I)</td></tr>
<tr><td>Datatype</td><td>INT32_T</td></tr>
<tr><td>Description</td><td>Totally exercised in the day</td></tr>
<tr><td colspan="2">total_held_q (Held, Total)</td></tr>
<tr><td>Datatype</td><td>INT64_T</td></tr>
</table>

<table>
<tr><td>Description</td><td>The total number of held in position, i.e. including any trades for the following clearing date.</td></tr>
<tr><td colspan="2">total_written_q (Written Total)</td></tr>
<tr><td>Datatype</td><td>INT64_T</td></tr>
<tr><td>Description</td><td>The total number of written in position, i.e. including any trades for the following clearing date.</td></tr>
<tr><td colspan="2">to_date_s (Date, To)</td></tr>
<tr><td>Datatype</td><td>char[8]</td></tr>
<tr><td>Description</td><td>To date. Format: YYYYMMDD.</td></tr>
<tr><td colspan="2">to_time_s (Time, To)</td></tr>
<tr><td>Datatype</td><td>char[6]</td></tr>
<tr><td>Description</td><td>Defines the to time. Format: HHMMSS.</td></tr>
<tr><td colspan="2">traded_c (Traded)</td></tr>
<tr><td>Datatype</td><td>UINT8_T</td></tr>
<tr><td>Description</td><td>Defines if the instrument is a tradable instrument or not.</td></tr>
<tr><td>Value Set</td><td><table><tr><th>name</th><th>value</th></tr><tr><td>Yes</td><td>1</td></tr><tr><td>No</td><td>2</td></tr></table></td></tr>
<tr><td colspan="2">traded_in_click_c (Traded in GENIUM)</td></tr>
<tr><td>Datatype</td><td>UINT8_T</td></tr>
<tr><td>Description</td><td>Specifies whether the series is traded in the system or not.</td></tr>
<tr><td>Value Set</td><td><table><tr><th>value</th><th>description</th></tr><tr><td>1</td><td>Yes</td></tr><tr><td>2</td><td>No</td></tr></table></td></tr>
<tr><td colspan="2">tradenumber (TRADENUMBER)</td></tr>
<tr><td>Datatype</td><td>INT32_T</td></tr>
<tr><td>Description</td><td>intermediate field.</td></tr>
<tr><td colspan="2">trades_allowed_c (Trades, Allowed)</td></tr>
<tr><td>Datatype</td><td>UINT8_T</td></tr>
<tr><td>Description</td><td>Is it allowed to store trades on the account</td></tr>
<tr><td>Value Set</td><td><table><tr><th>name</th><th>value</th></tr><tr><td>Yes</td><td>1</td></tr><tr><td>No</td><td>2</td></tr></table></td></tr>
<tr><td colspan="2">trade_number_i (Trade Number)</td></tr>
<tr><td>Datatype</td><td>INT32_T</td></tr>
</table>

<table>
<tr><td>Description</td><td>An increasing sequence number assigned to each trade. Trade number is unique within Instrument type</td></tr>
<tr><td colspan="2">trade_quantity_i (Quantity, Trade)</td></tr>
<tr><td>Datatype</td><td>INT64_T</td></tr>
<tr><td>Description</td><td>Define the number of contracts in the trade.</td></tr>
<tr><td colspan="2">trade_reporting_only_c (Only trade reports allowed)</td></tr>
<tr><td>Datatype</td><td>UINT8_T</td></tr>
<tr><td>Description</td><td>Specifies whether the series only allows trade reporting.</td></tr>
<tr><td>Value Set</td><td><table>
<tr><th>value</th><th>description</th></tr>
<tr><td>1</td><td>Yes</td></tr>
<tr><td>2</td><td>No</td></tr>
</table></td></tr>
<tr><td colspan="2">trade_rep_code_n (Trade Report Code)</td></tr>
<tr><td>Datatype</td><td>UINT16_T</td></tr>
<tr><td>Description</td><td>Defines the trade report type.</td></tr>
<tr><td colspan="2">trade_state_c (Trade, State)</td></tr>
<tr><td>Datatype</td><td>UINT8_T</td></tr>
<tr><td>Description</td><td>In what state is the trade?</td></tr>
<tr><td>Value Set</td><td><table>
<tr><th>value</th><th>description</th></tr>
<tr><td>1</td><td>Active. The trade is active.</td></tr>
<tr><td>2</td><td>Rectified. The trade has been rectified.</td></tr>
<tr><td>3</td><td>Deleted. The trade has been deleted.</td></tr>
<tr><td>4</td><td>Transferred. The trade has been transferred.</td></tr>
</table></td></tr>
<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>
</table></td></tr>
</table>

<table>
<tr><td></td><td>
<table>
<tr><th>value</th><th>description</th></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>
<tr><td>Datatype</td><td>UINT8_T</td></tr>
<tr><td>Description</td><td>Defines the Trade venue, i.e from where the trade emanates.</td></tr>
<tr><td colspan="2">transaction_number_n (Transaction Type Number)</td></tr>
<tr><td>Datatype</td><td>UINT16_T</td></tr>
<tr><td>Description</td><td>A number used to distinguish between different transactions to the same central subsystem.</td></tr>
<tr><td colspan="2">transitory_c (Transitory)</td></tr>
<tr><td>Datatype</td><td>UINT8_T</td></tr>
<tr><td>Description</td><td>Is the account a transitory account?</td></tr>
<tr><td>Value Set</td><td>
<table>
<tr><th>value</th><th>description</th></tr>
<tr><td>1</td><td>Yes</td></tr>
<tr><td>2</td><td>No</td></tr>
</table>
</td></tr>
<tr><td colspan="2">tra_cl_next_day_c (Cleared Next Day)</td></tr>
</table>

<table>
<tr><td>Datatype</td><td>CHAR</td></tr>
<tr><td>Description</td><td>Indicates whether the clearing date has been switched over to next clearing date or not for the instrument type.</td></tr>
<tr><td>Value Set</td><td><table><tr><th>value</th><th>description</th></tr><tr><td>Y</td><td>Yes</td></tr><tr><td>N</td><td>No</td></tr></table></td></tr>
<tr><td colspan="2">trc_id_s (Trade Report Class)</td></tr>
<tr><td>Datatype</td><td>char[10]</td></tr>
<tr><td>Description</td><td>The ID string for a trade report class. The trade report class contains a list of Trade Report Types.</td></tr>
<tr><td colspan="2">trd_cur_unit_c (Traded Currency Unit)</td></tr>
<tr><td>Datatype</td><td>UINT8_T</td></tr>
<tr><td>Description</td><td>Specifies the currency unit the instrument is traded in.</td></tr>
<tr><td>Value Set</td><td><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></td></tr>
<tr><td colspan="2">trr_id_s (Trade Report, Identity)</td></tr>
<tr><td>Datatype</td><td>char[4]</td></tr>
<tr><td>Description</td><td>The ID string for a trade report type.</td></tr>
<tr><td colspan="2">tv_nsec (Time in nanoseconds)</td></tr>
<tr><td>Datatype</td><td>INT32_T</td></tr>
<tr><td>Description</td><td>Elapsed time since the time in tv_sec, expressed in nanoseconds.</td></tr>
<tr><td colspan="2">tv_sec (Time in seconds)</td></tr>
<tr><td>Datatype</td><td>UINT32_T</td></tr>
<tr><td>Description</td><td>Elapsed time in seconds since the Epoch (1970-01-01 00:00:00 UTC).</td></tr>
<tr><td colspan="2">tz_exchange_s (Time Zone, Exchange)</td></tr>
<tr><td>Datatype</td><td>char[40]</td></tr>
<tr><td>Description</td><td>The time zone environment variable for the exchange.
(POSIX standard)
e.g. MET-1MET_DST-2,M3.5.0/2,M10.5.0/3</td></tr>
<tr><td colspan="2">tz_variable_s (TZ-Variable)</td></tr>
<tr><td>Datatype</td><td>char[40]</td></tr>
<tr><td>Description</td><td>The TZ environment variable for the exchange (POSIX standard).
e.g. MET-1MET_DST-2,M3.5.0/2,M10.5.0/3</td></tr>
<tr><td colspan="2">underlying_issuer_s (Underlying Issuer)</td></tr>
</table>

<table>
<tr><td>Datatype</td><td>char[6]</td></tr>
<tr><td>Description</td><td>Defines the issuer of the underlying.</td></tr>
<tr><td colspan="2">underlying_status_c (Underlying Status)</td></tr>
<tr><td>Datatype</td><td>UINT8_T</td></tr>
<tr><td>Description</td><td>Define the status of the underlying.</td></tr>
<tr><td>Value Set</td><td><table>
<tr><th>value</th><th>description</th></tr>
<tr><td>1</td><td>Active</td></tr>
<tr><td>2</td><td>Delisted</td></tr>
</table></td></tr>
<tr><td colspan="2">underlying_type_c (Type, Underlying)</td></tr>
<tr><td>Datatype</td><td>UINT8_T</td></tr>
<tr><td>Description</td><td>What type of underlying is it?</td></tr>
<tr><td>Value Set</td><td><table>
<tr><th>value</th><th>description</th></tr>
<tr><td>1</td><td>Stock</td></tr>
<tr><td>2</td><td>Currency</td></tr>
<tr><td>3</td><td>Interest rate</td></tr>
<tr><td>4</td><td>Energy</td></tr>
<tr><td>5</td><td>Soft and Agrics</td></tr>
<tr><td>6</td><td>Metal</td></tr>
<tr><td>7</td><td>Stock Index</td></tr>
<tr><td>8</td><td>Currency Index</td></tr>
<tr><td>9</td><td>Interest Rate Index</td></tr>
<tr><td>10</td><td>Energy Index</td></tr>
<tr><td>11</td><td>Softs and Agrics Index</td></tr>
<tr><td>12</td><td>Metal Index</td></tr>
</table></td></tr>
<tr><td colspan="2">undisclosed_min_ord_val_i (Minimum Order Value, Undisclosed Quantity)</td></tr>
<tr><td>Datatype</td><td>INT32_T</td></tr>
<tr><td>Description</td><td>Minimum order value for undisclosed quantity orders.
The value is always expressed in the primary currency unit.
The value is defined as quantity*price*price quotation factor.</td></tr>
<tr><td colspan="2">und_price_modifier_c (Modifier, Underlying Price)</td></tr>
<tr><td>Datatype</td><td>UINT8_T</td></tr>
<tr><td>Description</td><td>The modifier is used to recalculate the item after an underlying adjustment. The field is stored with 3 implicit decimals.</td></tr>
<tr><td>Value Set</td><td><table>
<tr><th>value</th><th>description</th></tr>
<tr><td>1</td><td>Modifier is added to the item</td></tr>
</table></td></tr>
</table>

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

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
upper_limit_i (Premium/Price, High Limit)	
Datatype	INT32_T
Description	The upper limit in the price interval.
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.
usr_id_n (User, Number)	
Datatype	UINT16_T
Description	A unique number that identified the user, used when subscribing for directed broadcast information.
utc_date_s (UTC, Date)	
Datatype	char[8]
Description	UTC date, format: YYYYMMDD.
utc_offset_i (UTC, Offset)	
Datatype	INT32_T
Description	Current offset between UTC and the local time specified in the TZ-variable.
utc_time_s (UTC, Time)	
Datatype	char[6]
Description	UTC time, format: HHMMSS.
vag_id_s (VAG, Identity)	
Datatype	char[12]
Description	Collateral valuation group ID
virtual_c (Virtual)	
Datatype	UINT8_T
Description	Is the underlying a virtual underlying?

<table>
<tr><td>Value Set</td><td><table><tr><th>value</th><th>description</th></tr><tr><td>1</td><td>Yes</td></tr><tr><td>2</td><td>No</td></tr></table></td></tr>
<tr><td colspan="2">virt_commodity_n (Virtual Underlying)</td></tr>
<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">warning_msg_s (Warning Message)</td></tr>
<tr><td>Datatype</td><td>char[80]</td></tr>
<tr><td>Description</td><td>This is a warning message that will be shown at a trading state change.</td></tr>
<tr><td colspan="2">warrant_c (Warrant)</td></tr>
<tr><td>Datatype</td><td>UINT8_T</td></tr>
<tr><td>Description</td><td>If the instrument is a warrant:</td></tr>
<tr><td>Value Set</td><td><table><tr><th>value</th><th>description</th></tr><tr><td>1</td><td>Yes</td></tr><tr><td>2</td><td>No</td></tr></table></td></tr>
<tr><td colspan="2">when_issued_c (When Issued)</td></tr>
<tr><td>Datatype</td><td>UINT8_T</td></tr>
<tr><td>Description</td><td>Not applicable.</td></tr>
<tr><td>Value Set</td><td><table><tr><th>value</th><th>description</th></tr><tr><td>2</td><td>No</td></tr></table></td></tr>
<tr><td colspan="2">yield_conv_n (Yield Convention)</td></tr>
<tr><td>Datatype</td><td>UINT16_T</td></tr>
<tr><td>Description</td><td>Yield Convention
Number of month</td></tr>
<tr><td colspan="2">yyyymmdd (YYYYMMDD)</td></tr>
<tr><td>Datatype</td><td>char[8]</td></tr>
<tr><td>Description</td><td>Intermediate field for date in YYYYMMDD format.</td></tr>
<tr><td colspan="2">yyyymmdd_s (Date)</td></tr>
<tr><td>Datatype</td><td>char[8]</td></tr>
<tr><td>Description</td><td>Date in ASCII. Format: YYYYMMDD</td></tr>
</table>

Index Register