



---

# K2/ifsc

---

## User Guide

For the internal use of the Budapest Stock Exchange and its members only!

© Copyright Effice Mérnöki Iroda Kft. 1999, 2001.
---

No part of this publication may be reproduced, stored in a retrieval system or transmitted, in any form or any means, electronic, photostatic, recorded or otherwise, without the written permission of the Effice Mérnöki Iroda Ltd. (2083 Solymár, Toldi u. 21, Hungary, telephone: +361 429 00 29).
--

version 1.1E MMTS I 1999.

version 2.2E MMTS II 2001.

version 2.3E BSE revision September 2002.

version 2.4E BSE revision September 2003.

version 2.5E BSE revision December 2003 MMTS I Market Maker function  
(not published)

version 2.6E BSE revision February 2005 MMTS II Market Maker function

version 2.7E BSE revision April 2005 Modifications and corrections

version 2.8E BSE revision May 2007 Modifications and corrections

version 2.9E BSE revision October 2008 New MMTS I secboard fields added

version 3.1E BSE revision July 2017 MIFID-2 regulation – new fields (MMTS-II)

K2/ifsc program was produced by the Effice Ltd.  
E-mail: <effice@effice.hu>

### *Table of Contents*

<i>1. General Information .....</i>	<i>7</i>
<i>2. Asking connection handle.....</i>	<i>10</i>
<i>3. Connecting to the K2 server.....</i>	<i>11</i>
<i>4. Disconnection from the K2 server .....</i>	<i>12</i>
<i>5. Handling sequence numbers.....</i>	<i>12</i>
<i>6. Handling record index .....</i>	<i>13</i>
<i>7. Handling field change.....</i>	<i>13</i>
<i>8. Query of the first changed record.....</i>	<i>14</i>
<i>9. Query of the first record.....</i>	<i>14</i>
<i>10. Query of the first changed field .....</i>	<i>15</i>
<i>11. The query of the next changed record.....</i>	<i>15</i>
<i>12. The query of the next record.....</i>	<i>16</i>
<i>13. The query of the next changed field .....</i>	<i>17</i>
<i>14. First query of the market by order .....</i>	<i>18</i>
<i>15. Query of the changes of the market by order .....</i>	<i>18</i>
<i>16. Query of the market by order list .....</i>	<i>19</i>
<i>17. First query of the market by price.....</i>	<i>19</i>
<i>18. Query of the changes of the market by price.....</i>	<i>20</i>
<i>19. Query of the market by price list.....</i>	<i>20</i>
<i>20. Order Entry.....</i>	<i>21</i>
<i>21. Transaction entry state change.....</i>	<i>22</i>
<i>22. The Market Maker Order Entry .....</i>	<i>22</i>
<i>23. State change for Market Maker order entry.....</i>	<i>23</i>
<i>24. The configuration of the market by order list .....</i>	<i>23</i>
<i>25. The configuration of the market by price list.....</i>	<i>24</i>
<i>26. System time query.....</i>	<i>24</i>
<i>27. Tracing.....</i>	<i>25</i>
<i>28. Query of error messages.....</i>	<i>25</i>
<i>29. Table codes .....</i>	<i>25</i>
<i>30. Transaction types.....</i>	<i>26</i>

<b>31. Order entry status .....</b>	<b>26</b>
<b>32. Toggling.....</b>	<b>27</b>
<b>33. Data types.....</b>	<b>27</b>
<b>34. Structure of the records - MMTS I.....</b>	<b>27</b>
Table <b>34.1. MMTS I - Market record .....</b>	<b>27</b>
Table <b>34.2. MMTS I - Instrument record.....</b>	<b>27</b>
Table <b>34.3. MMTS I - Sector record .....</b>	<b>28</b>
Table <b>34.4. MMTS I - Board record .....</b>	<b>28</b>
Table <b>34.5. MMTS I - Secboard record .....</b>	<b>28</b>
Table <b>34.6. MMTS I - User record .....</b>	<b>33</b>
Table <b>34.7. MMTS I - Firm record .....</b>	<b>35</b>
Table <b>34.8. MMTS I - Order record .....</b>	<b>36</b>
Table <b>34.9. MMTS I - Trade record.....</b>	<b>38</b>
Table <b>34.10. MMTS I - Negotiated deal order record .....</b>	<b>39</b>
Table <b>34.11. MMTS I - Audit Event record .....</b>	<b>41</b>
Table <b>34.12. MMTS I - Order book by order record .....</b>	<b>41</b>
Table <b>34.13. MMTS I - Order book by price record .....</b>	<b>42</b>
Table <b>34.14. MMTS I - Order book list .....</b>	<b>43</b>
Table <b>34.15. MMTS I - System time.....</b>	<b>43</b>
Table <b>34.16. MMTS I - Field changes of secboard table .....</b>	<b>44</b>
Table <b>34.17. MMTS I – Secboard additional information .....</b>	<b>44</b>
Table <b>34.18. MMTS I - Order entry record (query).....</b>	<b>45</b>
Table <b>34.19. MMTS I - Market Maker Order Entry record (query) .....</b>	<b>47</b>
Table <b>34.20. MMTS I - Order add record .....</b>	<b>48</b>
Table <b>34.21. MMTS I - Order cancellation record .....</b>	<b>49</b>
Table <b>34.22. MMTS I - Order amendment record .....</b>	<b>50</b>
Table <b>34.23. MMTS I - Tick up/tick down .....</b>	<b>51</b>
Table <b>34.24. MMTS I - Market Maker Order entry .....</b>	<b>51</b>
<b>35. Structure of the records - MMTS II. ....</b>	<b>53</b>
Table <b>35.1. MMTS II - Market record.....</b>	<b>53</b>
Table <b>35.2. MMTS II - Instrument record .....</b>	<b>53</b>
Table <b>35.3. MMTS II - Sector record.....</b>	<b>53</b>
Table <b>35.4. MMTS II - Board record.....</b>	<b>53</b>
Table <b>35.5. MMTS II - Secboard record.....</b>	<b>53</b>
Table <b>35.6. MMTS II - User record.....</b>	<b>56</b>
Table <b>35.7. MMTS II - Firm record.....</b>	<b>57</b>
Table <b>35.8. MMTS II - Order record.....</b>	<b>58</b>

Table	35.9. MMTS II - Trade record .....	59
Table	35.10. MMTS II - Negotiated deal record .....	61
Table	35.11. MMTS II - Audit Event record .....	62
Table	35.12. MMTS II - Order book by order record .....	63
Table	35.13. MMTS II - Order book by price record .....	64
Table	35.14. MMTS II - Order book list .....	65
Table	35.15. MMTS II - System time .....	65
Table	35.16. MMTS II - Field changes of secboard table .....	66
Table	35.17. MMTS II – Secboard additional information .....	66
Table	35.18. MMTS II - Order entry record (query) .....	67
Table	35.19. MMTS II – Market Maker Order entry record (query) .....	69
Table	35.20. MMTS II - Order add record .....	71
Table	35.21. MMTS II - Order cancellation record .....	72
Table	35.22. MMTS II - Order amendment record .....	73
Table	35.23. MMTS II - Tick up/Tick down .....	74
Table	35.24. MMTS II – Market Maker Order entry record .....	75
	36. Error codes .....	78
	37. Ifs field coding functions .....	80
	38. Ifs field decoding functions .....	81
	39. Ifs field step function .....	82

## Document History

Version	Author	Date	Summary of changes
1.1E	Effice Kft.	1999.	First version
2.2E	Effice Kft.	2001.	Extension for MMTS II
2.3E	BSE IT Development	September 2002.	BSE revision, corrections
2.4E	BSE IT Development	September 2003.	Corrections, table field descriptions added
2.5E (Not published)	BSE IT Development	December 2003	MMTS I Market Maker function descriptions translated and added to the English version
2.6E	BSE IT Development	February 2005	Revised version MMTS II Market Maker Order Entry functionality added. Change of the AveragePrice field type in the MMTS I Order record table and in the K2 program. The type was changed from int to double. Please note that this change may require changing also the client programs using that part of the table (the field in question is the last but one in the table.
2.7E	BSE IT Development	April 2005	BSE modifications and corrections Tracing and Query of Error Messages function calls modified. IndicativePrice field is added to MMTS I secboard record Field changes of secboard table record structure description corrected both for MMTS I and MMTS II Description of record structure corrected for MMTS I Tick up/tick down
2.8E	BSE IT Development	May 2007	BSE modification for making available the pointvalue (contract size) parameter Added or corrected explanations for a few fields of certain K2 tables.
2.9E	BSE IT Development	October 2008	Four new static datafields related to certificate products are added to MMTS I secboard record
3.1E	BSE IT Development	July 2017	Introducing new fields in connection of MIFID-2 regulations

## List of changes

The table below details the changes of the present version (v3.1E) in comparison to the previous published version: (2.9E). Please note that certain changes are only documentation changes (D), some of them are changes to the K2 server software (K2S) while others may affect the client systems (R and C+).

**Important!** This latter category (R and C+) must be handled carefully as it may require changes in the client application.

Changes marked with R necessitate the review of existing client routines and may need an update if they use the given corrected or modified feature.

Changes marked by C+ are new features. Clients are free to implement them in their applications if they have the proper authorisation in their licence file. This category may affect the client system only in case the client wishes to use and implement the new functionality.

No.	Section/Paragraph/Page	Description	Change type
1	33 Data types on page 27	Data type <b>datetimemsec</b> was introduced	K2S, R, C+
2	Table 35.8 MMTS II - Order record on page 58	Type of field <b>OrderTime</b> was changed; Field <b>ExecutionId</b> was introduced	K2S, R, C+
3	Table 35.9 MMTS II - Trade record on page 59	Type of fields <b>TradeTime</b> and <b>AmendTime</b> were changed; Fields <b>BuyExecutionId</b> and <b>SellExecutionId</b> were introduced	K2S, R, C+
4	Table 35.18 MMTS II - Order entry record (query) on page 67	Field <b>ExecutionId</b> was introduced	K2S, R, C+
5	Table 35.19 MMTS II – Market Maker Order entry record (query) on page 69	Field <b>ExecutionId</b> was introduced	K2S, R, C+
6	Table 35.20 MMTS II - Order add record on page 71	Field <b>ExecutionId</b> was introduced	K2S, R, C+
7	Table 35.22 MMTS II - Order amendment record on page 73	Field <b>ExecutionId</b> was introduced	K2S, R, C+
8	Table 35.23 MMTS II - Tick up/Tick down on page 74	Field <b>ExecutionId</b> was introduced	K2S, R, C+
9	Table 35.24 MMTS II – Market Maker Order entry record on page 75	Field <b>ExecutionId</b> was introduced	K2S, R, C+

### **1. General Information**

---

- By using the K2 the following information can be accessed from the MMTS system:
  1. Market table.
  2. Instrument table.
  3. Sector table.
  4. Board (Equity, Bond, etc.) table.
  5. Secboard table.
  6. User table.
  7. Firm table.
  8. Order table.
  9. Trade table.
  10. Negdeal table.
  11. Audit Event table<sup>1</sup>
  12. Order Book by Order.<sup>2</sup>
  13. Order Book by Price
  14. Order Book List
  15. System time
  16. Field changes of secboard table
  17. Secboard additional information table
  18. Order Entry table.<sup>3</sup>
  19. Market Maker Order Entry table<sup>3</sup>

In addition the K2 makes it possible to enter, amend or cancel the orders in the MMTS system as well as to initiate Tick up/Tick down and Market Maker order entry transactions. The structures used for entering transactions are described by the following records:

20. Order Add record
21. Order cancellation record
22. Order Amendment record
23. Tick up/Tick down record
24. Market Maker Order entry record

- The IFSC library gives possibility of connecting to more K2 servers by asking for a connection handle for each connection. Then every query and transaction refers to

---

<sup>1</sup> The query of the Audit Event table contains also the Audit Event Group and Audit Event Type data, therefore there is no separate query for those data.

<sup>2</sup> Handling the order books is different from that of the other tables and depends on license.

<sup>3</sup> This table does not contain data from the MMTS, but the orders entered by the users of the K2.

those connection handles. The server can be connected to MMTS I or MMTS II systems. The features listed below refer to one connection.

- The K2 system can serve multiple users at the same time. The identifiers, passwords, states and privileges of the users are defined in the K2 system.

The state of the user can be Active or Suspended. The user privileges are the next:

**query** Data query

**entry** Order entry, amendment and cancellation

**confirm** Confirmation of the orders entered into the K2

**config** Configuration of the Order Book list

**bypass** The orders entered by a user having this privilege are automatically confirmed

**admin** Not used

- As the data are received from the K2 in packages, if the query of a table was started, it is recommended to continue until we are synchronized to the changes (until "no more data" is received). If the tables are often changed before all the changes are queried, then the system drops the remaining part of the package, and fetches a package from the new table. It can result in serious performance degradation.
- The query of data is generally done by using sequence numbers. The K2 stores a sequence number for each record in every table. If a record is changed in a table (or a new record is stored) then the sequence number of this record will be the biggest in the given table. The ifsc stores the sequence numbers for the user for each table. In the query this sequence number will be sent to the K2, and the answer will contain the records which have higher sequence numbers than the given one. The sequence number stored in the ifsc will be set to the highest received sequence number. Related functions are **ifsc\_set\_get\_seq**, **ifsc\_get\_first\_record** and **ifsc\_get\_next\_record**.
- In the case of the secboard table it is possible to query and follow the changes on field level also by using an optional function. In this case the static part of the table has to be queried first for all of the records. The answers for these queries do not contain any information for the dynamic fields. If there are no more records, then it has to be continued with the queries of the field level changes. The answers contain the field changes by records. Related functions for query static data are:

**ifsc\_set\_get\_idx**

**ifsc\_get\_first\_idx**

**ifsc\_get\_next\_idx**

and for field level changes:

**ifsc\_set\_get\_field\_seq**

**ifsc\_get\_first\_field\_chg**

**ifsc\_get\_next\_field\_chg**

- The order books are handled specially. The K2 contains a so-called Order Book List, where the users having *config* privilege can register (or cancel) the order books, which are observed by the K2. The actual Order Book List can be queried. When an order book is queried, the answer always contains the full actual order book, independently from the changes happened meantime.



- Initiating trading transactions using the K2 is made in subsequent steps. There are four types of trading transactions: new order entry, order amendment, order cancellation and tickup/tickdown. In the first step the requested trading transaction is sent to the K2 server, where it gets a unique identifier (orderid) and it is checked. Depending on the result of the check, its status will be accepted (**IFS\_ORDER\_ACCEPTED**) or denied (**IFS\_ORDER\_DENIED**). If it is denied then an error message is written in the order entry record. In the second step the order entry record has to be confirmed (**IFS\_ORDER\_CONFIRMED**) or denied (**IFS\_ORDER\_DENIED**) by changing its status. In the third step the K2 server recognizes that an order entry record was confirmed (status is **IFS\_ORDER\_CONFIRMED**), and K2 sets the status to unknown (**IFS\_ORDER\_UNKNOWN**) then sends it to the MMTS system, where it is checked again. The result of this check is got by the K2, and it records it in the status of the order entry record. (Entered **IFS\_ORDER\_ENTERED** or Refused **IFS\_ORDER\_REFUSED**). If it is refused then the error message is recorded. At communication error when the K2 does not receive the result of the order transaction the status remains permanently unknown. The further events in the MMTS (trades, order amendments, etc.) do not influence the status of the order entry record. Further information about the orders can be got by the query of the order table. The order number (OrdNo) in the order table is independent from the identifier of the order entry record (orderid), the former is generated by the MMTS. The order entry records and the MMTS order records can be linked in the client systems by the BrokerRef field. This is not the responsibility of the K2.

In case of order amendment or cancellation the order number (OrdNo) of the original order generated by MMTS system has to be given in the order entry record. In case of delete the orders the fields of which match the given parameters are withdrawn. (In case of order withdraw the brokerref field is not sent to the MMTS.) The empty and 0 parameters are not checked. The MMTS system sends error message only if a parameter is wrong. If no order is found with the given parameters than error message is *not* sent!

### ATTENTION !!!

**If on deleting (withdrawing) orders the OrderNo field and the other non-mandatory (not obligatory) fields of the delete record (order cancellation record) are empty or 0, then all the orders belonging to the given firm will be withdrawn. Therefore special care should be taken when the OrderNo field is left empty.**

If the user has *bypass* privilege, then the order entry record sent in the first phase is confirmed automatically after it is accepted by the K2 and it is sent to the MMTS.

If the user has an appropriate license then in case of successfully entered new order the K2 fills the *OrdNo* fields of order entry record with the order number of the new order by analyzing the received user message. In case of amendment or cancellation the order entry records contain the order numbers, therefore analyzing of the user message is not necessary.

- In case of appropriate license the broker system can give a further identifier to the order entry record by filling *InternalRef* field. This field is not sent to the MMTS system. Broker systems can pass extra information (e.g. client id.) to each other in this field.
- The Market Maker order Entry requests are handled similarly to the normal Order Entry requests, but there are separate functions for the order entry and status change

and also the Market Maker Order Entries can be queried as a separate table. A continuously increment identifier is attached to normal as well as to Market Maker Order Entry requests in the order of their arrival. Thus the order of their arrival can be determined from the value of the identifier given by the K2 and the entry requests are sent to the MMTS (Multi-Market Trading System) in the very same order. The amendment and cancellation (withdrawal) of the Market Maker orders are possible using the amendment and withdraw functions for the normal orders.

- Each table record has an identifier. When using the functions that handle order entry and order books these identifiers have to be given.
- The fields are separated by zero bytes in the buffers.
- In the return code of the functions the success is zero.
- The state of the PGW process connected to the MMTS system can be queried using the **ifsc\_get\_systime** function. The *K2QueryTimeOffset* field of the output record contains the number of seconds passed before the PGW finished processing the answers to queries and transactions. The *PgwState* contains information whether the PGW is running or not. This information is provided by an observer process.
- The following functions can be used only with an appropriate license:

```
ifsc_get_first_orderbook
ifsc_get_next_orderbook
ifsc_get_orderbook_list
ifsc_get_first_marketbyprx
ifsc_get_next_marketbyprx
ifsc_get_marketbyprx_list
ifsc_orderbook_conf
ifsc_marketbyprx_conf
ifsc_get_first_field_chg
ifsc_get_next_field_chg
ifsc get systime
```

The K2 license contains the maximum number of the users that are allowed to be logged in at the same time. If this is exceeded the further connections will be refused.

## *2. Asking connection handle*

---

The connection handle is made by the **ifsc create** function.

```
int ifsc_create(IFSC_HDL * handle,
               char * host_name,
               char * service_name);
```

**int ifsc\_handle**

Error code, zero, if the creation is successful.

**char \* host name**

The name of the host, where the K2 is running.

**char \* service name**

The name of the port used by the K2.

Example:

```
IFSC_HDL handle;
.
if (ifsc_create(&handle,
               "host",
               "service")) {
    fprintf(stderr, "Error at creation: %s\n",
            ifsc_get_last_errmsg());
    exit(1);
}
```

### **3. Connecting to the K2 server**

---

The connection is made by the **ifsc connect** function.

```
int ifsc_connect(IFSC_HDL handle,
                char * user_name,
                char * password,
                int * tradeid,
                int * pid,
                int * mmts_type);
```

**int ifsc\_connect**

Error code, zero, if the connection is successful.

**IFSC\_HDL handle**

Connection descriptor for a given server.

**char \* user\_name**

User identifier for the K2.

**char \* password**

User password for the K2.

**int \* tradeid**

Trade identifier. It is the return value of a UNIX `time()` function. If it is stored, then during a subsequent connection it can be checked whether the K2 was connected to the same MMTS. This value is *not* changed during the operation of the MMTS, only after start or restart.

**int \* pid**

Process identifier of PGW. If the client reconnects, the client program can check whether the same K2 serves that. If yes, the client can set record sequence numbers, record index numbers and field change numbers of tables to continue the broken query.

**int \* mmts\_type**

Type of the MMTS system where PGW is connected to. (MMTS I or MMTS II.)

Example:

```
int trdid, pid, mmts_type;
IFSC_HDL handle;
ifsc_create(&handle, ...
.
```

```
if (ifsc_connect(handle,
                  "user",
                  "password",
                  &trdid,
                  &pid,
                  &mmts_type)) {
    fprintf(stderr, "Error at connection: %s\n",
            ifsc_get_last_errmsg());
    exit(1);
}
```

### **4. Disconnection from the K2 server**

---

The disconnection from the K2 server is made by the **ifsc\_disconnect** function.

```
int ifsc_disconnect(IFSC_HDL handle);
```

**int ifsc\_disconnect**

Error code. Zero, if the disconnection is successful.

**IFSC\_HDL handle**

Connection descriptor for a given server.

Example:

```
int s;
IFSC_HDL handle;

if (s = ifsc_disconnect(handle)) {
    fprintf(stderr, "Error at disconnection: %s\n",
            ifsc_get_last_errmsg());
    exit(s);
}
```

### **5. Handling sequence numbers**

---

The record change sequence number of a table can be queried and set by the **ifsc\_set\_get\_seq** function. These sequence numbers are used for record change request: **ifsc\_get\_next\_record**.

```
int ifsc_set_get_seq(IFSC_HDL handle,
                    int table,
                    int seqno);
```

**int ifsc\_set\_get\_seq**

**IFS\_UNKNOWNTABLE** in the case of an erroneous table code, the old sequence number otherwise.

**IFSC\_HDL handle**

Connection descriptor for a given server.

**int table**

The code of the referenced table.

**int seqno**

The sequence number being set. If it is negative, the sequence number is not changed. In this way the sequence number can be queried.

Example:

```
IFSC_HDL handle;
ifsc_set_get_seq(handle, IFS_T_BOARD, 0);
int seq;
.
seq = ifsc_set_get_seq(handle, IFS_T_BOARD, -1);
```

### **6. Handling record index**

---

The record index of a table can be queried and set by the **ifsc\_set\_get\_idx** function. These record indices are used for static data request: **ifsc\_get\_next\_record**.

```
int ifsc_set_get_idx(IFSC_HDL handle,
                    int table,
                    int idx);
```

**int ifsc\_set\_get\_idx**

**IFS\_UNKNOWNTABLE** in the case of an erroneous table code, the old record index otherwise.

**IFSC\_HDL handle**

Connection descriptor for a given server.

**int table**

The code of the referenced table.

**int idx**

The record index being set. If it is negative, the record index is not changed. In this way the record index can be queried.

Example:

```
IFSC_HDL handle;
ifsc_set_get_idx(handle, IFS_T_BOARD, 0);
int idx;
.
idx = ifsc_set_get_seq(handle, IFS_T_BOARD, -1);
```

### **7. Handling field change**

---

The field change number of the secboard table can be queried and set by the **ifsc\_set\_get\_field\_seq** function. These field change numbers are used for field change request: **ifsc\_get\_next\_field\_chg**.

```
int ifsc_set_get_field_seq(IFSC_HDL handle,
                           int table,
                           int seqno);
```

**int ifsc\_set\_get\_field\_seq**

**IFS\_UNKNOWNTABLE** in the case of an erroneous table code, the old field change number otherwise.

**IFSC\_HDL handle**

Connection descriptor for a given server.

**int table**

The code of the referenced table.

**int seqno**

The field change number being set. If it is negative, the field change number is not changed. In this way the field change number can be queried.

Example:

```
IFSC_HDL handle;
ifsc_set_get_field_seq(handle, IFS_T_BOARD, 0);
int seq;
.
seq = ifsc_set_get_field_seq(handle, IFS_T_BOARD, 1);
```

### **8. Query of the first changed record**

---

The query of the changed records of a table can be started by the **ifsc\_get\_first\_record** function and has to be continued by the **ifsc\_get\_next\_record** function every time. The **ifsc\_get\_first\_record** function sets the sequence number to zero and calls the **ifsc\_get\_next\_record**.

```
int ifsc_get_first_record(IFSC_HDL handle,
                        char ** rec_buf,
                        int * len,
                        int table_code,
                        int * seqno);
```

**int ifsc get first record**

Error code. It is zero if there are any new or changed records left, IFS\_NOMORE if there are no more new or changed records.

**IFSC\_HDL handle**

Connection descriptor for a given server.

**char \*\* rec buf**

The record which was changed.

**int \* len**

The length of the record which was changed (byte).

**int table\_code**

The code of the table.

**int \* seqno**

The sequence number of the record; return value (**ifsc\_set\_get\_seq** also returns with this value).

### **9. Query of the first record**

---

The query of the records of a table in physical order can be started by the **ifsc\_get\_first\_idx** function and has to be continued by the **ifsc\_get\_next\_idx** function every time. The **ifsc\_get\_first\_idx** function sets the record index number to zero and calls the **ifsc\_get\_next\_idx**.

In case of secboard table only the static fields are listed in the returned records.

```
int ifsc_get_first_idx(IFSC_HDL handle,
                    char ** rec_buf,
                    int * len,
                    int table_code,
                    int * idx);
```

**int ifsc\_get\_first\_idx**

Error code. It is zero if there are any records left, IFS\_NOMORE if there are no more records.

**IFSC\_HDL handle**

Connection descriptor for a given server.

**char \*\* rec\_buf**

The record.

**int \* len**

The length of the record (byte).

**int table\_code**

The code of the table.

**int \* idx**

The record index of the record; return value. (**ifsc\_set\_get\_idx** also returns with this value.)

### *10. Query of the first changed field*

---

The query of the changed fields of secboard table can be started by the **ifsc\_get\_first\_field\_chg** function and has to be continued by the **ifsc\_get\_next\_field\_chg** function every time. The **ifsc\_get\_first\_field\_chg** function sets the field change number to zero and calls the **ifsc\_get\_next\_field\_chg**. (Before calling this function the static data can be requested with **ifsc\_get\_first\_idx** and **ifsc\_get\_next\_idx** functions.)

```
int ifsc_get_first_field_chg(IFSC_HDL handle,
                           char ** rec_buf,
                           int * len,
                           int table_code,
                           int * seqno);
```

**int ifsc get first field chg**

Error code. It is zero if there are any changed fields left, IFS\_NOMORE if there are no more changed fields.

**IFSC\_HDL handle**

Connection descriptor for a given server.

**char \*\* rec\_buf**

Secboard id and pairs of field code and field value

**int \* len**

The length of the changed fields (byte).

**int table\_code**

The code of the table.

**int \* seqno**

The field change number; return value. (**ifsc set get field seq** also returns with this value.)

### *11. The query of the next changed record*

---

The records of a table can be queried by the **ifsc\_get\_next\_record** function.

```
int ifsc_get_next_record(IFSC_HDL handle,
                        char ** rec_buf,
                        int * len,
```

```
int table_code,
int * seqno);

int ifsc_get_next_record
    Error code, it is zero if there are any new or changed records left,
    IFS_NOMORE if there are no more new or changed records.

IFSC_HDL handle
    Connection descriptor for a given server.

char ** rec_buf
    The changed record.

int * len
    The length of the changed record (byte).

int table_code
    The code of the table.

int * seqno
    The sequence number of the record.
```

Example:

```
int s, len, seqno;
char *recvb;
IFSC_HDL handle;
.
while ((s = ifsc_get_next_record(handle,
                                &recvb,
                                &len,
                                IFS_T_BOARD,
                                &seqno)) == 0) {
    write(1, recvb, len);
}
```

---

## ***12. The query of the next record***

---

The records of a table in physical order can be queried by the **ifsc\_get\_next\_idx** function.

In case of secboard table only the static fields are listed in the returned records.

```
int ifsc_get_next_idx(IFSC_HDL handle,
                    char ** rec_buf,
                    int * len,
                    int table_code,
                    int * idx);

int ifsc_get_next_idx
    Error code, it is zero if there are any records left, IFS_NOMORE if there are
    no more records.

IFSC_HDL handle
    Connection descriptor for a given server.

char ** rec_buf
    The record.

int * len
    The length of the record (byte).

int table_code
    The code of the table.
```



**int \* idx**

The record index of the record.

Example:

```
int s, len, idx;
char *recvb;
IFSC_HDL handle;
.
while ((s = ifsc_get_next_idx(handle,
                              &recvb,
                              &len,
                              IFS_T_BOARD,
                              &idx)) == 0) {
    write(1, recvb, len);
}
```

### ***13. The query of the next changed field***

---

The field changes of secboard table can be queried by the **ifsc\_get\_next\_field\_chg** function.

```
int ifsc_get_next_field_chg(IFSC_HDL handle,
                           char ** rec_buf,
                           int * len,
                           int table_code,
                           int * seqno);
```

**int ifsc\_get\_next\_field\_chg**

Error code, it is zero if there are any changed fields left, IFS\_NOMORE if there are no more changed fields.

**IFSC\_HDL handle**

Connection descriptor for a given server.

**char \*\* rec\_buf**

Secboard id and pairs of field code and field value

**int \* len**

The length of the field changes (byte).

**int table code**

The code of the table.

**int \* seqno**

The last field change number.

Example:

```
int s, len, seqno;
char *recvb;
IFSC_HDL handle;
.
while ((s=ifsc_get_next_field_change(handle,
&recvb,
&len,
IFS_T_SECBOARD,
&seqno)) == 0) {
write(1, recvb, len);
}
```

## ***14. First query of the market by order***

---

The order books by order, being in the market by order list, can be queried by the **ifsc\_get\_first\_orderbook** function for the first time. The following order book by order changes can be queried by the **ifsc\_get\_next\_orderbook** function. The market by order list can be got by the **ifsc\_get\_orderbook\_list** function.

```
int ifsc_get_first_orderbook(IFSC_HDL handle,
                             char ** rec_buf,
                             int * len,
                             char * sec_board_id);
```

**int ifsc\_get\_first\_orderbook**

Error code. Zero, if the query was successful.

**IFSC\_HDL handle**

Connection descriptor for a given server.

**char \*\* rec\_buf**

The content of the order book.

**int \* len**

The size of the order book (byte).

**char \* sec\_board\_id**

The identifier of the secboard.

## ***15. Query of the changes of the market by order***

---

The order books by order, being in the market by order list, can be queried by the **ifsc\_get\_next\_orderbook** function. If the order book by order has been changed since the last query the whole order book is received. The market by order list can be got by the **ifsc\_get\_orderbook\_list** function.

```
int ifsc_get_next_orderbook(IFSC_HDL handle,
                             char ** rec_buf,
                             int * len,
                             char * sec_board_id);
```

**int ifsc\_get\_next\_orderbook**

Error code. Zero, if the query was successful, IFS\_NOMORE if there is no change.

**IFSC\_HDL handle**

Connection descriptor for a given server.

**char \*\* rec\_buf**

The content of the order book.

**int \* len**

The size of the order book (byte).

**char \* sec\_board\_id**

The identifier of the secboard.

Example:

```
int s, len;
char *buff;
char secboardid[] = "0FRMOL";
IFSC_HDL handle;
.
```

```
if ((s = ifsc_get_next_orderbook(handle,
                                &buff,
                                &len,
                                secboardid)) == 0) {
    write(1, buff, len);
}
else
    error_handler(...
```

### 16. Query of the market by order list

---

The actual content of the market by order list can be queried by the **ifsc\_get\_orderbook\_list** function.

```
int ifsc_get_orderbook_list(IFSC_HDL handle,
                           char ** rec_buf,
                           int * len);
```

**int ifsc\_get\_orderbook\_list**

Error code, it is zero if the query is successful.

**IFSC\_HDL handle**

Connection descriptor for a given server.

**char \*\* rec\_buf**

The order book list.

**int \* len**

The length of the order book list (byte).

Example:

```
int len;
char *buffer;
IFSC_HDL handle;
.
if ((s = ifsc_get_orderbook_list(handle,
                                &buffer,
                                &len)) == 0) {
    write(1, buffer, len);
} else
    error_handler(...
```

### 17. First query of the market by price

---

The order books by price, being in the market by price list, can be queried for the first time by the **ifsc\_get\_first\_marketbyprx** function. The following changes of the order book by price can be queried by the **ifsc\_get\_next\_marketbyprx** function. The market by price list can be got by the **ifsc\_get\_marketbyprx\_list** function.

```
int ifsc_get_first_marketbyprx(IFSC_HDL handle,
                              char ** rec_buf,
                              int * len,
                              char * sec_board_id);
```

**int ifsc\_get\_first\_marketbyprx**

Error code. Zero, if the query was successful.

**IFSC\_HDL handle**

Connection descriptor for a given server.

**char \*\* rec\_buf**  
The content of the order book.

**int \* len**  
The size of the order book (byte).

**char \* sec\_board\_id**  
The identifier of the secboard.

### *18. Query of the changes of the market by price*

---

The order books by price, being are in the market by price list, can be queried by the **ifsc get next marketbyprx** function. If the order book by price has been changed since the last query the whole order book is received. The market by price list can be got by the **ifsc get marketbyprx list** function.

```
int ifsc_get_next_marketbyprx(IFSC_HDL handle,
                             char ** rec_buf,
                             int * len,
                             char * sec_board_id);
```

**int ifsc\_get\_next\_marketbyprx**  
Error code. Zero, if the query was successful, IFS\_NOMORE if there is no change.

**IFSC\_HDL handle**  
Connection descriptor for a given server.

**char \*\* rec\_buf**  
The content of the order book.

**int \* len**  
The size of the order book (byte).

**char \* sec\_board\_id**  
The identifier of the secboard.

Example:

```
int s, len;
char *buff;
char secboardid[] = "0FRMOL";
IFSC_HDL handle;

if ((s=ifsc_get_next_marketbyprx(handle,
                                &buff,
                                &len,
                                secboardid)) == 0) {

    write(1, buff, len);
}
else
    error_handler(...
```

### *19. Query of the market by price list*

---

The actual content of the market by price list can be queried by the **ifsc\_get\_marketbyprx\_list** function.

```
int ifsc_get_marketbyprx_list(IFSC_HDL handle,
                             char ** rec_buf,
```

```
int * len);
```

**int ifsc\_get\_marketbyprx\_list**  
Error code, it is zero if the query is successful.

**IFSC\_HDL handle**  
Connection descriptor for a given server.

**char \*\* rec\_buf**  
The order book list.

**int \* len**  
The length of the market by price list (byte).

Example:

```
int len;
char *buffer;
IFSC_HDL handle;
.
if ((s = ifsc_get_marketbyprx_list(handle,
                                   &buffer,
                                   &len)) == 0) {
    write(1, buffer, len);
} else
    error_handler(...
```

---

## **20. Order Entry**

Transactions can be entered into the system using the **ifsc\_orderentry** function. The types of transactions are: order entry, order cancellation, order amendment, and Tick up/Tick down. The functions return only the error/success code, the orders that were entered can be queried by the **ifsc\_get\_next\_rekord** function using the **IFS\_T\_ORDERENTRY** table code

```
int ifsc_orderentry(IFSC_HDL handle,
                    int trans_code,
                    char * orderentry,
                    int order_len,
                    int * new_orderid);
```

**int ifsc\_orderentry**  
Error code, it is zero if the entry is successful.

**IFSC\_HDL handle**  
Connection descriptor for a given server.

**int trans\_code**  
Type of the transaction: Entry, cancel, amendment, Tick up/Tick down.

**char \* orderentry**  
The content of the order, it is dependent on the transaction type.

**int order\_len**  
The length of the order (byte). It is dependent on the transaction type.

**int \* new\_orderid**  
The identifier of the order given by the K2.

Example:

```
int s, len, orderid;
char *buf;
IFSC_HDL handle;
.
if ((s = ifsc_orderentry(handle,
                        IFS_ACTION_ORDER_WITHDRAW,
                        buf,
                        len,
                        &orderid)) != 0) {
    error_handler(...
```

---

### **21. Transaction entry state change**

---

The state of the transaction entries can be changed by the **ifsc\_orderentry\_status\_chg** function.

```
int ifsc_orderentry_status_chg(IFSC_HDL handle,
                              int orderid,
                              char new_status_code);
```

**int ifsc\_orderentry\_status\_chg**

Error code, it is zero if the status change was successful.

**IFSC\_HDL handle**

Connection descriptor for a given server.

**int orderid**

Transaction entry identifier.

**char new\_status\_code**

The new status code.

Example:

```
int s, orderid;
IFSC_HDL handle;
.
s = ifsc_orderentry_status_chg(handle,
                              orderid,
                              IFS_ORDER_CONFIRMED);
```

---

### **22. The Market Maker Order Entry**

---

Market Maker orders can be entered using the **ifsc\_mmorder** function. The functions return only the error/success code, the orders that were entered can be queried by the **ifsc\_get\_next\_rekord** function using the **IFS\_T\_MMORDER** table code.

```
int ifsc_mmorder(IFSC_HDL handle,
                char * mmorder,
                int order_len,
                int * new_orderid);
```

**int ifsc\_mmorder**

Error code, it is zero if the entry is successful.

**IFSC\_HDL handle**

Connection descriptor for a given server.

**char \* mmorder**

The contents of the order.

**int order\_len**

The length of the order (byte). It is dependent on the transaction type.

**int \* new\_orderid**

The identifier of the order given by the K2.

Example:

```
int s, len, orderid;
char *buf;
IFSC_HDL handle;
.
if ((s = ifsc_mmorder(handle,
                      buf,
                      len,
                      &orderid)) != 0) {
    error_handler(...
```

### ***23. State change for Market Maker order entry***

---

The state of the Market Maker order entries can be changed by the **ifsc\_mmorder\_status\_chg** function.

```
int ifsc_mmorder_status_chg(IFSC_HDL handle,
                           int orderid,
                           char new_status_code);
```

**int ifsc\_mmorder\_status\_chg**

Error code, it is zero if the status change was successful.

**IFSC\_HDL handle**

Connection descriptor for a given server.

**int orderid**

Market Maker order entry identifier.

**char new\_status\_code**

The new status code.

Example:

```
int s, orderid;
IFSC_HDL handle;
.
s = ifsc_mmorder_status_chg(handle,
                           orderid,
                           IFS_ORDER_CONFIRMED);
```

### ***24. The configuration of the market by order list***

---

Order books by order can be inserted or deleted in the market by order list using the **ifsc\_orderbook\_conf** function.

```
int ifsc_orderbook_conf(IFSC_HDL handle,
                       char * sec_board_id,
                       int add_or_remove);
```

**int ifsc\_orderbook\_conf**

Error code, it is zero if the change of the configuration was successful.

**IFSC\_HDL handle**

Connection descriptor for a given server.

**char \* sec\_board\_id**

The identifier of the secboard being inserted or deleted.

**int add\_or\_remove**

Insertion / Deletion flag.

```
int s;  
IFSC_HDL handle;  
.  
s = ifsc_orderbook_conf(handle,  
                        "0FOREMOL"  
                        IFS_SWITCH_ON);
```

## **25. The configuration of the market by price list**

---

Order books by price can be inserted or deleted in the market by price list using the **ifsc marketbyprx conf** function.

```
int ifsc_marketbyprx_conf(IFSC_HDL handle,  
                          char * sec_board_id,  
                          int add_or_remove);
```

**int ifsc\_marketbyprx\_conf**

Error code, it is zero if the change of the configuration was successful.

**IFSC\_HDL handle**

Connection descriptor for a given server.

**char \* sec\_board\_id**

The identifier of the secboard being inserted or deleted.

**int add\_or\_remove**

Insertion / Deletion flag.

```
int s;  
IFSC_HDL handle;  
.  
s = ifsc_marketbyprx_conf(handle,  
                          "0FOREMOL"  
                          IFS_SWITCH_ON);
```

## **26. System time query**

---

The trading date and time, K2 system date and time and system status can be queried by the **ifsc get systime** function.

```
int ifsc_get_systime(IFSC_HDL handle,  
                    char ** rec_buf,  
                    int * len);
```

**int ifsc\_get\_systime**

Error code, it is zero if the operation is successful.

**IFSC\_HDL handle**

Connection descriptor for a given server.

**char \*\* rec\_buf**

Systime record.

Example:

```
int len;  
char *buffer;  
IFSC_HDL handle;
```



```
.
if ((s = ifsc_get_systime(handle,
                             buffer,
                             &len)) == 0) {
    write(1, buffer, len);
} else
    error_handler(...
```

## 27. Tracing

---

The tracing can be switched on or off with the **ifsc set trace** function. When it is switched on, the K2 system writes trace information about the internal operation into the `ifstrace.log` file in the actual. If this file already exists, then the new information is appended.

```
int ifsc_set_trace(IFSC_HDL handle, int on_off);
```

**int ifsc\_set\_trace**

Error code, it is zero if the operation is successful.

**IFSC\_HDL handle**

Connection descriptor for a given server.

**int on\_off**

On / Off flag.

Example:

```
int s;
IFSC_HDL handle;
s = ifsc_set_trace(handle, IFS_SWITCH_ON);
```

## 28. Query of error messages

---

The last error messages can be got by using the **ifsc get last errmsg** function.

```
char * ifsc_get_last_errmsg(IFSC_HDL handle);
```

**char \* ifsc\_get\_last\_errmsg**

Pointer to the last error message.

**IFSC\_HDL handle**

Connection descriptor for a given server.

Example:

```
IFSC_HDL handle;
printf(stderr, ifsc_get_last_errmsg(handle));
```

## 29. Table codes

---

**IFS\_T\_MARKET** Market table.

**IFS\_T\_INSTR** Instrument table.

**IFS\_T\_SECTOR** Sector table.

**IFS\_T\_BOARD** Board table.

**IFS\_T\_SECBOARD** Secboard table.

**IFS\_T\_FIRM** Firm table.

**IFS\_T\_USER** User table.

**IFS\_T\_ORDER** Order table.

**IFS\_T\_MMORDER** Market Maker order entry table.

**IFS\_T\_AUDITEVENT** Audit event table.

**IFS\_T\_TRADE** Trade table.

**IFS\_T\_NEGDEAL** Negotiated Deal table.

**IFS\_T\_ORDERENTRY** Order Entry table.

**IFS\_T\_PRICEPARAM** Secboard additional information table.

**IFS\_T\_LAST** Last (non existing) table. It is useful for writing program loops.

### 30. Transaction types

---

**IFS\_ACTION\_ORDER\_ADD** Order entry.

**IFS\_ACTION\_ORDER\_WITHDRAW** Order cancellation.

**IFS\_ACTION\_ORDER\_AMEND** Order amendment.

**IFS\_ACTION\_TICKUPTICKDOWN** Tick up/tick down.

### 31. Order entry status

---

**IFS\_ORDER\_ACCEPTED** Accepted by the K2.

**IFS\_ORDER\_DENIED** Denied.

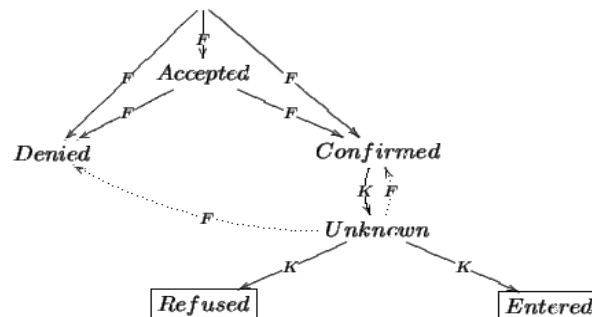
**IFS\_ORDER\_CONFIRMED** Confirmed.

**IFS\_ORDER\_ENTERED** Successfully entered into the MMTS.

**IFS\_ORDER\_REFUSED** Refused by the MMTS.

**IFS\_ORDER\_UNKNOWN** Unknown.

Valid state transitions<sup>4</sup>:



Explanation:

**F** State transition made by the user.

**K** State transition made by the K2.

... Occasional state transition during restart.

---

<sup>4</sup> The states in rectangles are final states.

### 32. Toggling

---

General on / off toggles.

**IFS SWITCH ON** Switch on.

**IFS SWITCH OFF** Switch off.

### 33. Data types

---

The records provided by the K2 consist of standardized fields. The fields are separated by zero bytes. The separating zero byte is *included* in the length. Data field *must not* contain zero bytes.

**char** Single byte.

**int** Integer with **IFS\_INT\_LEN** length.

**double** Value with **IFS\_DOUBLE\_LEN** length.

**string** String with length depending on the type of the field.

**ids** String with **IFS\_IDS\_LEN** length for the record identifiers.

**fixreal** Fix point value fraction number described in two fields. This is a record with two fields. The first field **double** is the value, the second **int** is the number of decimals. (Do not forget, each field is closed with trailing zero.)

**datetime** Date-time structure with two fields. The first **int** field contains the date, the second **int** contains the time. (Do not forget, each field is closed with trailing zero.)

**datetimemsec** Date-time-milliseconds structure with three fields. The first **int** field contains the date, the second **int** contains the time, the third **int** contains the milliseconds (value between 0 and 999). (Do not forget, each field is closed with trailing zero.)

**enum** Enumerated types described in integer.

**bool** Identical with enum. 0 = **FALSE**, 1 = **TRUE**.

### 34. Structure of the records - MMTS I

---

Table 34.1. MMTS I - Market record

Name	Length	Type	Description
Id	IFS_IDS_LEN	ids	Market code
Name	IFS_NAME_LEN	string	Name of the market
Status	IFS_CHAR_LEN	char	current status of the market: (A)ctive, (S)uspended, (D)efunct

Table 34.2. MMTS I - Instrument record

Name	Length	Type	Description
Id	IFS_IDS_LEN	ids	Instrument code

Name	Length	Type	Description
Name	IFS_NAME_LEN	string	Name of the instrument
Status	IFS_CHAR_LEN	char	current status of the instrument: (A)ctive, (S)uspended, (D)efunct

Table 34.3. MMTS I - Sector record

Name	Length	Type	Description
Id	IFS_IDS_LEN	ids	Sector code
Name	IFS_NAME_LEN	string	Name of the sector

Table 34.4. MMTS I - Board record

Name	Length	Type	Description
Id	IFS_IDS_LEN	ids	Unique Identifier for the board
Name	IFS_NAME_LEN	string	Name of the board
Status	IFS_CHAR_LEN	char	Current status of the board, (A)ctive, (S)uspended, (D)efunct
Ordermethods	IFS_ORDERMETHODS_LEN	string	defines the types of orders that will be accepted for the securities. OrderMethods is a bit-mask.

Table 34.5. MMTS I - Secboard record

Name	Length	Type	Description
Id	IFS_SECBOARDID_LEN	string	The name is derived by concatenating the board id and security id
SecName	IFS_SEC_NAME_LEN	string	security name
SecRemarks	IFS_SEC_REMARK_LEN	string	free-format current remarks for the security
SecShortName	IFS_SEC_SHORTNAME_LEN	string	security short name
SecState	IFS_CHAR_LEN	char	security status
SessionName	IFS_NAME_LEN	string	the name of the first session
MarketId	IFS_IDS_LEN	ids	market code
InstrId	IFS_IDS_LEN	ids	instrument code
BoardId	IFS_BOARDID_LEN	string	board code
SectorId	IFS_IDS_LEN	ids	sector code

Name	Length	Type	Description
CurrName	IFS_IDS_LEN	ids	name of the currency in which to express values such as price, commission, yield etc of the securities.
CurrDec	IFS_INT_LEN	int	Decimal values of the currency.
FaceValue	IFS_DOUBLE_LEN	double	face value of the secboard. This value is expressed in the appropriate currency given by the secboards trading rule
PrevEarn	IFS_DOUBLE_LEN	double	last earnings figure for the security. It is manually maintained in the database.
PrevDate	IFS_INT_LEN	int	last date on which the secboard traded before today
PrevPrice	IFS_INT_LEN	int	last price which the security traded before today.
VolumeTraded	IFS_DOUBLE_LEN	double	total volume of trades for the secboard since it was listed
ValueTraded	IFS_DOUBLE_LEN	double	total value of trades for the secboard since it was listed.
PrevYield	IFS_INT_LEN	int	yield for the last price at which the security traded today
AccruedInterest	IFS_DOUBLE_LEN	double	current accrued interest per unit for fixed interest securities
PrimDist	IFS_CHAR_LEN	char	'P' if the security is in primary distribution. 'S' if it is traded in the secondary market
NumOpenOrders	IFS_INT_LEN	int	current number of open orders for the security
OrigIssueQty	IFS_DOUBLE_LEN	double	planned issue size
ListedQty	IFS_DOUBLE_LEN	double	number of security shares listed
TradeableSize	IFS_DOUBLE_LEN	double	tradable quantity of shares for the security
AvgLast	IFS_INT_LEN	int	average closing price of the security over the last 30 days
AvgVolume	IFS_DOUBLE_LEN	double	average volume of the security traded over the last 30 days
AvgValue	IFS_DOUBLE_LEN	double	average value of security traded over the last 30 days

Name	Length	Type	Description
PriceDecimals <sup>5</sup>	IFS_INT_LEN	int	Number of decimal places to display when expressing various values and prices. This is NOT changed during the trading day,
VisibleStat	IFS_CHAR_LEN	char	Are Static Security Parameters visible to the trader 'Y' , 'N'
LotSize	IFS_INT_LEN	int	lot size of a security. This is the marketable parcel of securities. Order quantities are entered in terms of lotsize. E.g. If lot size for a security is 10 and an order of quantity of 5 is placed, the order is actually for 50 shares, I.e. 5 lots of 10
IndexSec	IFS_CHAR_LEN	char	Index Id that the security is linked to index table 'Y' , 'N'
SecCode	IFS_IDS_LEN	ids	A string used by the exchange to identify the security (similar to short name)
EntitlementIndicator	IFS_CHAR_LEN	char	With (C) or without deident (X) or , ,
DeliveryBasis	IFS_CHAR_LEN	char	Type of delivery, ' ' at present
SecIndicators	IFS_SEC_INDICATOR_LEN	string	I)slamic, F)oreign
DepositoryType	IFS_CHAR_LEN	char	Type of depository 'I', 'P'
MarketValue	IFS_DOUBLE_LEN	double	Listed quantity * last traded price
IssuerId	IFS_IDS_LEN	ids	Issuing Firm
Isin	IFS_IDS_LEN	ids	security isin code
ListingType	IFS_IDS_LEN	ids	Type of listing
YieldDecimals	IFS_INT_LEN	int	Yield Decimal Places
PrevBidDate	IFS_INT_LEN	int	Previous Bid Date
PrevOfferDate	IFS_INT_LEN	int	Previous Offer Date
MatDat	IFS_INT_LEN	int	Maturity date for certain types of certificate products
StrikePrice	IFS_STRIKEPRICE_LEN	string	Strike price for certain types of certificate products
Barrier	IFS_BARRIER_LEN	string	Barrier for certain types of certificate products
UnSecProdName	IFS_UNSECPRODNAME_LEN	string	Underlying security product name for certain types of certificate products

<sup>5</sup> All fields (int or double) containing prices have to be adjusted according to the number in PriceDecimals (have to be divided by the appropriate power of 10) in order to get the correct price value.

Changeable fields			
Name	Length	Type	Description
Remarks	IFS_SEC_REMARK_LEN	string	free-format current remarks for the security.
Status	IFS_CHAR_LEN	char	the status of the secboard
BidPrice	IFS_INT_LEN	int	current best buy price for the security.
BidDepth	IFS_INT_LEN	int	total quantities of all open buy orders at the best bid price for the security.
BidDepthT	IFS_INT_LEN	int	total quantities of all open buy orders for the security.
BidN	IFS_INT_LEN	int	total number of open buy orders for the security.
OfferPrice	IFS_INT_LEN	int	current best sell price for the security
OfferDepth	IFS_INT_LEN	int	total quantities of all open sell orders at the best offer price for the security.
OfferDepthT	IFS_INT_LEN	int	total quantities of all open sell orders for the security.
OfferN	IFS_INT_LEN	int	total number of open sell orders for the security.
openPrice	IFS_INT_LEN	int	open price for the secboard today. For securities that have a pre-opening period, this is the calculated open price. For other securities, this is the first traded price today
highPrice	IFS_INT_LEN	int	highest price at which the security has traded today.
lowPrice	IFS_INT_LEN	int	lowest price at which the security has traded today.
lastTradedPrice	IFS_INT_LEN	int	last price at which the security traded today.
lastOffMktPrice	IFS_INT_LEN	int	last traded market price
changePrice	IFS_INT_LEN	int	change of the last traded price today from the previous day's close price for the security
qty	IFS_INT_LEN	int	quantity of the last trade for the security today, expressed as a number of lots
time	IFS_INT_LEN	int	time at which the last trade for the security was executed today.
volumeToday	IFS_DOUBLE_LEN	double	total quantity of shares traded in the secboard today
valueToday	IFS_DOUBLE_LEN	double	total value traded in the secboard today

Name	Length	Type	Description
OrderMethods	IFS_ORDERMETHOD_LEN	string	OrderMethods defines the types of orders that will be accepted for the securities. OrderMethods is a bit-mask. Each bit in the field indicates one type of order. If the bit is set to 1, then that type of order is accepted
changeLTP	IFS_INT_LEN	int	The change from the last traded price
Session	IFS_NAME_LEN	string	name of the current trading session
Value	IFS_DOUBLE_LEN	double	The total value of all trades matched for the security today
lastYield	IFS_INT_LEN	int	last traded yield
qtyOffMkt	IFS_INT_LEN	int	Off market last quantity.
refPrice	IFS_DOUBLE_LEN	double	Reference price
NumTrades	IFS_INT_LEN	int	total number of matched trades today for the secboard.
NumOrders	IFS_INT_LEN	int	sum of NumBuys and NumSells for the security
changePricePct	IFS_INT_LEN	int	percentage change of the last price traded today from the previous day's close price for the security.
WAPrice	IFS_INT_LEN	int	Weighted average price
NegdealInit	IFS_CHAR_LEN	char	Which side enters the Negotiated deals into the system? Following are the valid values for this field. B - Buy side of the deal must enter the deal into the system S - Sell side of the deal must enter the deal into the system. E - Either side of the deal can enter the deal into the system.
SuspDate	IFS_INT_LEN	int	secboard suspend date
SuspTime	IFS_INT_LEN	int	secboard suspend time
AnnounceInd	IFS_CHAR_LEN	char	Has news 'Y' , ' '
UpperPriceLimit	IFS_INT_LEN	int	Upper price limit
LowerPriceLimit	IFS_INT_LEN	int	Lower price limit
CBLimitUpper	IFS_INT_LEN	int	Upper circuit breaker limit
CBLimitLower	IFS_INT_LEN	int	Lower circuit breaker limit
ClosePrice	IFS_INT_LEN	int	Closing price
VolTraded	IFS_DOUBLE_LEN	double	Volume traded
ValTraded	IFS_DOUBLE_LEN	double	Value traded
VolOffMktToday	IFS_DOUBLE_LEN	double	Daily traded off market volume



Name	Length	Type	Description
ValOffMktToday	IFS_DOUBLE_LEN	double	Daily traded off market value
VolOffMktTraded	IFS_DOUBLE_LEN	double	Traded off market volume
ValOffMktTraded	IFS_DOUBLE_LEN	double	Traded off market value
OpenYield	IFS_INT_LEN	int	open yield for the secboard today. For securities that have a pre-opening period, this is the calculated open yield. For other securities, this is the first traded yield today
HighYield	IFS_INT_LEN	int	highest yield at which the security has traded today.
LowYield	IFS_INT_LEN	int	lowest yield at which the security has traded today.
CloseYield	IFS_INT_LEN	int	Close yield
RefYield	IFS_INT_LEN	int	reference yield
ChangeYield	IFS_INT_LEN	int	change of the last traded yield today from the previous day's close yield for the security
ChangeLTY	IFS_INT_LEN	int	change of last traded yield for the last two trades
LastOffMktYield	IFS_INT_LEN	int	Last off market yield
BidYield	IFS_INT_LEN	int	best bid yield
OfferYield	IFS_INT_LEN	int	best offer yield
WAYield	IFS_INT_LEN	int	Weighted average yield
PrivateOB	IFS_CHAR_LEN	char	Private Orderbook is on Y – Yes N – No
IndicativePrice	IFS_INT_LEN	int	Indicative price calculated only during the opening and closing sessions. It has no meaning (set to zero) in other trading sessions.

Table 34.6. MMTS I - User record

Name	Length	Type	Description
Id	IFS_IDS_LEN	ids	user code
Name	IFS_NAME_LEN	string	name of the user.
FirmId	IFS_IDS_LEN	ids	firm code
RoleId	IFS_IDS_LEN	ids	Role identifier
DefaultRoleId	IFS_IDS_LEN	ids	default role identifier
Status	IFS_CHAR_LEN	char	status of the user
ApproveOrders	IFS_CHAR_LEN	char	if set to (Y)es, it indicates whether all orders entered by the user must be verified by a Firm Manager in the user's firm, before being placed in the Trading Engine's order book.. 'Y' or 'N'

Name	Length	Type	Description
Security_on	IFS_CHAR_LEN	char	'Y ' or 'N'
ContactDetail	IFS_NAME_LEN	string	contact details
FreeText	IFS_FREE_TEXT_LEN	string	Text of the free
IPGateway	IFS_INT_LEN	int	network address of the Gateway through which the user has logged into the Trading Engine. This address is automatically detected by the Trading Engine when the user logs into the System. For a user who is connected directly to the Trading Engine and does not route through a Gateway, IPGateway is the actual network address from which the user has logged on, and IPClient will be zero
IPClient	IFS_INT_LEN	int	network address from which the user has logged into the Trading Engine (if the user is currently logged in). This address is automatically detected by the Trading Engine when the user logs into the System. For a user who is connected directly to the Trading Engine and does not route through a Gateway, IPClient will be zero and IPGateway is the network address from which the user has logged on.
Client_GW	IFS_INT_LEN	int	
Today buy order			
SingleVolume	IFS_DOUBLE_LEN	double	Highest volume (quantity) for a single order (open or matched)
SingleValue	IFS_DOUBLE_LEN	double	Highest value (price*quantity) for a single order (open or matched)
TotalVolume	IFS_DOUBLE_LEN	double	Total volume (quantity) for all orders (open or matched)
TotalValue	IFS_DOUBLE_LEN	double	Total value (price*quantity) for all orders (open or matched)
Today sell order			
SingleVolume	IFS_DOUBLE_LEN	double	Highest volume (quantity) for a single order (open or matched)
SingleValue	IFS_DOUBLE_LEN	double	Highest value (price*quantity) for a single order (open or matched)
TotalVolume	IFS_DOUBLE_LEN	double	Total volume (quantity) for all orders (open or matched)
TotalValue	IFS_DOUBLE_LEN	double	Total value (price*quantity) for all orders (open or matched)

Today buy trade			
SingleVolume	IFS_DOUBLE_LEN	double	Highest volume (quantity) for a single trade
SingleValue	IFS_DOUBLE_LEN	double	Highest value (price*quantity) for a single trade
TotalVolume	IFS_DOUBLE_LEN	double	Total volume (quantity) for all trades
TotalValue	IFS_DOUBLE_LEN	double	Total value (price*quantity) for all trades
Today sell trade			
SingleVolume	IFS_DOUBLE_LEN	double	Highest volume (quantity) for a single trade
SingleValue	IFS_DOUBLE_LEN	double	Highest value (price*quantity) for a single trade
TotalVolume	IFS_DOUBLE_LEN	double	Total volume (quantity) for all trades
TotalValue	IFS_DOUBLE_LEN	double	Total value (price*quantity) for all trades

Table 34.7. MMTS I - Firm record

Name	Length	Type	Description
Id	IFS_IDS_LEN	ids	firm code
Name	IFS_NAME_LEN	string	name of the firm
Status	IFS_CHAR_LEN	char	status of the firm
AvgVolume	IFS_DOUBLE_LEN	double	Average volume of securities traded by the firm
AvgValue	IFS_DOUBLE_LEN	double	Average value of securities traded by the firm
Security_on	IFS_CHAR_LEN	char	'Y' 'N'
Type	IFS_IDS_LEN	ids	Firm type
ContactDetail	IFS_NAME_LEN	string	Contact detail
FreeText	IFS_FREE_TEXT_LEN	string	Text of the free
UsersLogged	IFS_INT_LEN	int	Number of users logged in
Today buy order			
SingleVolume	IFS_DOUBLE_LEN	double	Highest volume (quantity) for a single order (open or matched)
SingleValue	IFS_DOUBLE_LEN	double	Highest value (price*quantity) for a single order (open or matched)
TotalVolume	IFS_DOUBLE_LEN	double	Total volume (quantity) for all orders (open or matched)
TotalValue	IFS_DOUBLE_LEN	double	Total value (price*quantity) for all orders (open or matched)
Today sell order			
SingleVolume	IFS_DOUBLE_LEN	double	Highest volume (quantity) for a single order (open or matched)
SingleValue	IFS_DOUBLE_LEN	double	Highest value (price*quantity) for a single order (open or matched)

TotalVolume	IFS_DOUBLE_LEN	double	Total volume (quantity) for all orders (open or matched)
TotalValue	IFS_DOUBLE_LEN	double	Total value (price*quantity) for all orders (open or matched)
Today buy trade			
SingleVolume	IFS_DOUBLE_LEN	double	Highest volume (quantity) for a single trade
SingleValue	IFS_DOUBLE_LEN	double	Highest value (price*quantity) for a single trade
TotalVolume	IFS_DOUBLE_LEN	double	Total volume (quantity) for all trades
TotalValue	IFS_DOUBLE_LEN	double	Total value (price*quantity) for all trades
Today sell trade			
SingleVolume	IFS_DOUBLE_LEN	double	Highest volume (quantity) for a single trade
SingleValue	IFS_DOUBLE_LEN	double	Highest value (price*quantity) for a single trade
TotalVolume	IFS_DOUBLE_LEN	double	Total volume (quantity) for all trades
TotalValue	IFS_DOUBLE_LEN	double	Total value (price*quantity) for all trades

Table 34.8. MMTS I - Order record

Name	Length	Type	Description
OrdNo	IFS_INT_LEN	int	System-assigned order number of the order. Order numbers are assigned sequentially by the System as each order is validated and placed in the order book. Order numbers are continuous from day-to-day, that is the first order number today is one greater than the last order number on the previous day's trading
Date	IFS_INT_LEN	int	Date on which the order was entered
Time	IFS_INT_LEN	int	Time on which the order was entered
OrderStatus	IFS_CHAR_LEN	char	current MMTS status of the order A – Amended I – Inactive stop order C – Unconfirmed M – Matched O – Open U – Unapproved W – Withdrawn Y – unconfirmed buy L – unconfirmed sell D – disabled

Name	Length	Type	Description
BuySell	IFS_CHAR_LEN	char	buy or sell indicator for the order (B)uy order or (S)ell order
OrderMethods	IFS_ORDERMETHOD_LEN	string	OrderMethods field defines the type of the order
BrokerRef	IFS_BROKERREF_LEN	string	Broker reference, free text. Upto 40 character long but truncated to 20 ch overnight for expiries longer than one day
UserId	IFS_IDS_LEN	ids	user code
FirmId	IFS_IDS_LEN	ids	firm code
SecBoardId	IFS_SECBOARDID_LEN	string	secboard code
TrdAccId	IFS_IDS_LEN	ids	Trading account identifier
TrdAccName	IFS_NAME_LEN	string	Trading account name
Price	IFS_DOUBLE_LEN	double	order price. Price may be zero for a market order. Because Price is a long value, the position of the decimal point is not maintained in the field. The number of decimals is obtained from the Decimals field in the Securities table for the order's security.
Yield	IFS_DOUBLE_LEN	double	order yield
Quantity	IFS_DOUBLE_LEN	double	is the disclosed quantity of the order expressed as a number units (shares, bonds, etc).
Hidden	IFS_DOUBLE_LEN	double	is the hidden quantity of the order expressed as a number units (shares, bonds, etc).
Balance	IFS_DOUBLE_LEN	double	is the total unmatched quantity of the order, taking into account both the disclosed and the hidden quantities of the order expressed as a number of units (shares, bonds, etc).
Value	IFS_DOUBLE_LEN	double	Value of the order
SettleDate	IFS_INT_LEN	int	settlement date for trades executed today for this secboard
PrevOrdNo	IFS_INT_LEN	int	Previous order number. (In case of amendment.)
ExpDate	IFS_INT_LEN	int	Expiration date
ExpTime	IFS_INT_LEN	int	Expiration time
TriggerPrice	IFS_INT_LEN	int	Trigger price
NextOrdNo	IFS_INT_LEN	int	Next order number. (In case of amendment.)
AveragePrice	IFS_DOUBLE_LEN	double	Average price. <b>Please note the warning below the table!</b>
MinFillQty	IFS_INT_LEN	int	Minimal fill quantity

**WARNING!!!** For this specific case and specific table PriceDecimals + 2 decimal digits should be used when adjusting the content of the AveragePrice field in order to get the real average price.

Table 34.9. MMTS I - Trade record

Name	Length	Type	Description
TrdNo	IFS_INT_LEN	int	System-assigned trade number of the trade. Trade numbers are assigned sequentially by the System as each trade is executed by the System. The buy record and the sell record for each trade have the same TradeNo value. Trade numbers are continuous from day-to-day, that is the first trade number today is one greater than the last trade number on the previous day's trading.
BuyOrdNo	IFS_INT_LEN	int	Buy order number. This field is completed only for own orders (own trades).
SellOrdNo	IFS_INT_LEN	int	Sell order number. This field is completed only for own orders (own trades).
TradeTime	IFS_INT_LEN	int	time at which the trade was executed by the System
AmendTime	IFS_INT_LEN	int	time at which the trade was cancelled (if it is cancelled).
BuyBrokerRef	IFS_BROKERREF_LEN	string	Buy broker reference, free text. Upto 40 character long but may be truncated to 20 ch if the trade is generated from an order older than one day
SellBrokerRef	IFS_BROKERREF_LEN	string	Sell broker reference, free text. Upto 40 character long but may be truncated to 20 ch if the trade is generated from an order older than one day
BuyUserId	IFS_IDS_LEN	ids	Buy user code
SellUserId	IFS_IDS_LEN	ids	Sell user code
BuyFirmId	IFS_IDS_LEN	ids	Buy firm code
SellFirmId	IFS_IDS_LEN	ids	Sell firm code
BuyTrdAccId	IFS_IDS_LEN	ids	Buy trading- account code
SellTrdAccId	IFS_IDS_LEN	ids	Sell trading- account code
SecBoardId	IFS_SECBOARDID_LEN	string	secboard code
Price	IFS_INT_LEN	int	trade price. Because Price is a long value, the position of the decimal point is not maintained in the field. The number of decimals is obtained from the Decimals field in the SecBoard table for the trade's security
Yield	IFS_INT_LEN	int	trade yield
Quantity	IFS_INT_LEN	int	trade quantity expressed as a number units (shares, bonds, etc).
Value	IFS_DOUBLE_LEN	double	value of the trade

Name	Length	Type	Description
SettleDate	IFS_INT_LEN	int	Settlement date
TradeDate	IFS_INT_LEN	int	Trade Date as integer in YYYYMMDD format. (not operating system date!)
TradeStatus	IFS_CHAR_LEN	char	current status of the trade M – Matched (This is a normally matched trade) W – Withdrawn (This trade has been cancelled)
TradeSource	IFS_CHAR_LEN	char	A or F automated or fixed.
DissemTime	IFS_INT_LEN	int	Disseminate time
DissemStatus	IFS_CHAR_LEN	char	Disseminate status 'N' – Nomatched 'I' – Nodelayed 'D' – Delayed 'S' – Sent 'R' – Released 'L' – Lapsed

Table 34.10. MMTS I - Negotiated deal order record

Name	Length	Type	Description
NegDealNo	IFS_INT_LEN	int	NegDeal Order Id
CPNegDealNo	IFS_INT_LEN	int	Counter Party Order Id
Time	IFS_INT_LEN	int	Negdeal order time
OrderStatus	IFS_CHAR_LEN	char	current status of the order A – Amended I – Inactive C – Unconfirmed M – Matched O – Open U – Unapproved W – Withdrawn Y – unconfirmed buy L – unconfirmed sell D – disabled
BuySell	IFS_CHAR_LEN	char	buy or sell indicator for the order (B)uy order or (S)ell order
BrokerRef	IFS_BROKERREF_LEN	string	free-format field that may be entered by the user for each order
UserId	IFS_IDS_LEN	ids	User code
FirmId	IFS_IDS_LEN	ids	Firm code
CPUserId	IFS_IDS_LEN	ids	Counter Party user code
CPFirmId	IFS_IDS_LEN	ids	Counter Party firm code

Name	Length	Type	Description
TrdAccId	IFS_IDS_LEN	ids	Trading account identifier
SecBoardId	IFS_SECBOARDID_LEN	string	secboard code
Price	IFS_INT_LEN	int	order price. Price may be zero for a market order. Because Price is a long value, the position of the decimal point is not maintained in the field. The number of decimals is obtained from the Decimals field in the Secboard table for the order's security.
Quantity	IFS_INT_LEN	int	is the disclosed quantity of the order expressed as a number units (shares, bonds, etc).
SettleDate	IFS_INT_LEN	int	Settlement date
Timeout	IFS_INT_LEN	int	time in seconds that is allowed for negotiated deals to be confirmed. After this time, the users involved in the deal will be sent messages from the Trading System to remind them that their deal is unconfirmed
Yield	IFS_INT_LEN	int	Yield of the order
DissemTime	IFS_INT_LEN	int	Disseminate time
DissemStatus	IFS_CHAR_LEN	char	Disseminate status 'N' – Nomatched 'I' – Nodelayed 'D' – Delayed 'S' – Sent 'R' – Released 'L' – Lapsed
TradeSource	IFS_CHAR_LEN	char	F for negotiated deals (fix orders).
WarningTimeout	IFS_INT_LEN	int	If not an unconfirmed neg. deal without a reminder, where neg. deals do have a confirmation warningTimeout
OrderMethods	IFS_ORDERMETHOD_LEN	string	OrderMethods defines the type of the order. OrderMethods is a bit-mask. Each bit in the field indicates an order type.



Table 34.11. MMTS I - Audit Event record

Name	Length	Type	Description
AuditEventId	IFS_INT_LEN	int	Sequence no. of the event message
isFirmSpecific	IFS_CHAR_LEN	char	If the message is intended for a certain firm
EventGroup	IFS_CAPTION_LEN	string	Event Group (e.g. Market Maker group)
EventType	IFS_CAPTION_LEN	string	Event type (e.g. breach of obligation)
Time	IFS_INT_LEN	int	Time of sending
DispMode	IFS_CHAR_LEN	char	Mode of display (e.g. popup or status line)
ToUserId	IFS_IDS_LEN	ids	Target UserId of the message
ToFirmId	IFS_IDS_LEN	ids	Target FirmId of the message
FromUserId	IFS_IDS_LEN	ids	UserId of sender (empty in case of automatic messages sent by MMTS)
BoardId	IFS_BOARDID_LEN	string	Identifier of board the event is related to
SecId	IFS_SECCODE_LEN	string	Identifier of the security the event is related to
UserId	IFS_IDS_LEN	ids	Target User identifier
FirmId	IFS_IDS_LEN	ids	Target Firm identifier
TextLen	IFS_INT_LEN	int	Actual length of the message
EventText	IFS_MSG_LEN	string	Text of the message

Table 34.12. MMTS I - Order book by order record

Name	Length	Type	Description
SecBoardId	IFS_SECBOARDID_LEN	string	Secboard code
Occupied <sup>6</sup>	IFS_CHAR_LEN	char	If Y then this orderbook is available for usage. If N then the order book is not on the monitoring list.
NumBuys	IFS_INT_LEN	int	Number of rows on the buy side.
NumSells	IFS_INT_LEN	int	Number of rows on the sell side.
UserId <sup>7</sup>	IFS_IDS_LEN	ids	Buy user code
Price	IFS_INT_LEN	int	Buy price of the order.

<sup>6</sup> K2 observing status: 'Y'<sup>7</sup> part repeated **NumBuys** times.

Name	Length	Type	Description
Qty	IFS_INT_LEN	int	Buy balance of the order. (Open volume.)
Hidden	IFS_CHAR_LEN	char	Buy hidden flag. It is 'H' if the order has hidden part.
UserId <sup>10</sup>	IFS_IDS_LEN	ids	Sell user code
Price	IFS_INT_LEN	int	Sell price of the order.
Qty	IFS_INT_LEN	int	Sell balance of the order. (Open volume.)
Hidden	IFS_CHAR_LEN	char	Sell hidden flag. It is 'H' if the order has hidden part.

Table 34.13. MMTS I - Order book by price record

Name	Length	Type	Description
SecBoardId	IFS_SECBOARDID_LEN	string	Secboard code
Occupied <sup>8</sup>	IFS_CHAR_LEN	char	If Y then this orderbook is available for usage. If N then the order book is not on the monitoring list.
OtherNOrder	IFS_INT_LEN	int	Number of orders on other boards.
NumBuys	IFS_INT_LEN	int	Number of rows on the buy side.
NumSells	IFS_INT_LEN	int	Number of rows on the sell side.
Price <sup>9</sup>	IFS_INT_LEN	int	Buy price level of row
Yield	IFS_INT_LEN	int	Buy yield level of row
Qty	IFS_INT_LEN	int	Buy Balance on the price level row. (Open volume.)
UserQty	IFS_INT_LEN	int	Buy Quantity of the current user on the row.
VOrders	IFS_INT_LEN	int	Number of visible buy orders on the row
VFirms	IFS_INT_LEN	int	No of firms placing Visible Buy orders on the given row
MM	IFS_CHAR_LEN	char	'M' if there is marker maker order on the row
Hidden	IFS_CHAR_LEN	char	Hidden flag. It is 'H' if any order on the row has hidden part.

<sup>8</sup> K2 observing status: 'Y'<sup>9</sup> part repeated **NumBuys** times.

Name	Length	Type	Description
Flag	IFS_CHAR_LEN	char	User has order in row = '*', User has no order in row = 'N', User has best order in row = 'I'
Price <sup>10</sup>	IFS_INT_LEN	int	Sell price level of row
Yield	IFS_INT_LEN	int	Sell yield level of row
Qty	IFS_INT_LEN	int	Sell Balance on the price level row. (Open volume.)
UserQty	IFS_CHAR_LEN	char	Sell Quantity of the current user on the row.
VOrders	IFS_INT_LEN	int	Number of visible sell orders on the row
VFirms	IFS_INT_LEN	int	No of firms placing Visible Sell orders on the given row
MM	IFS_CHAR_LEN	char	'M' if there is marker maker order on the row
Hidden	IFS_CHAR_LEN	char	Hidden flag. It is 'H' if any order on the row has hidden part.
Flag	IFS_CHAR_LEN	char	User has order in row = '*', User has no order in row = 'N', User has best order in row = 'I'

Table 34.14. MMTS I - Order book list

Name	Length	Type	Description
SecBoardId <sup>8</sup>	IFS_SECBOARDID_LEN	string	Secboard code

Table 34.15. MMTS I - System time

<sup>10</sup> part repeated **NumSells** times.<sup>8</sup> Variable length record, this field can be repeated several times.

Name	Length	Type	Description
TradeDate	IFS_INT_LEN	int	Trade Date as integer in YYYYMMDD format. (not operating system date!)
TETime <sup>9</sup>	IFS_INT_LEN	int	MMTS trade time HHMMSS format
K2Date	IFS_INT_LEN	int	K2 system date
K2Time <sup>10</sup>	IFS_INT_LEN	int	K2 system time
K2QueryTimeOffset	IFS_INT_LEN	int	Time offset to the MMTS engine (seconds since the Epoch)
PgwState	IFS_INT_LEN	int	State of the Pgw process. 0 is start state, 1 is termination.

Table 34.16. MMTS I - Field changes of secboard table

Name	Length	Type	Description
Id	IFS_SECBOARDID_LEN	string	Secboard identifier
FieldId <sup>11</sup> + FieldData	1 + IFS_???_LEN		One byte FieldId followed by Field's data content (This is repeated several times according to the number of fields changed.)

Note:

The field changes of secboard table can be queried by the **ifsc\_get\_next\_field\_chg** function.

```
int ifsc_get_next_field_chg(IFSC_HDL handle,
                           char ** rec_buf,
                           int * len,
                           int table_code,
                           int * seqno);
```

The function returns the length of the received string in the len parameter. By decrementing this returned length value after reading each field by (1+IFS\_???\_LEN) programmers can determine if there are any remaining field changes in the received string.

Table 34.17. MMTS I – Secboard additional information

This table is used to query additional static information on the given secboard. The data are static and they will not change during the trading day, but they may change from one trading day to another.

Name	Length	Type	Description
Id	IFS_SECBOARDID_LEN	string	Secboard identifier

<sup>9</sup> Value of trading system clock in (HHMMSS) format.

<sup>10</sup> Value of computer clock which K2 running in. The format is (HHMMSS).

<sup>11</sup> The field codes listed in `fields.h`.

Name	Length	Type	Description
PriceParamArray	IFS_PRICEPARAM_LEN	string	Price step (tick size) information (see below)
MinQty	IFS_INT_LEN	int	Minimum quantity size requirement for an order

**PriceParamArray ( MMTS I) explanation**

This field defines the price step (tick size) for a given security & board pair. The tick size can be defined in ranges, and additionally it can be defined as an exact value or as a percentage of the order's real price . The price parameter structure is as follows:

L|V|Pdec|n:m[,n:m] ...

Where:

- L is "P" or "Y" and indicates whether the lookup range refers to a price or a yield;
- V is "D", "P" indicating whether the parameter value defaults to price/yield (as per the first parameter), or is a percentage;
- Pdec is a single digit integer defining the number of decimal places (pricedecimals) for price values in general.
- [n:m] are pairs of values that make up the array where n is the starting price/yield of the lookup range and m is the parameter value that applies to the range

The first range always starts at zero and ends at the start of the following range. The last range ends at infinity.

e.g. P|D|2|0:0.01,1.50:0.05

Meaning a price based rule where the parameters for the price ranges are:

Range  $\geq 0$  and  $< 1.50 = 0.01$

Range  $\geq 1.50 = 0.05$

Table 34.18. MMTS I - Order entry record (query)

Note: This table can be used to check the client's own orders entered into the K2.

Name	Length	Type	Description
orderid	IFS_INT_LEN	int	Order id given by the pgw.
OrdNo <sup>12</sup>	IFS_INT_LEN	int	Order number in the MMTS
TrdAccId	IFS_IDS_LEN	ids	Trading account identifier
BuySell	IFS_CHAR_LEN	char	buy or sell indicator for the order
OrderMethodSet	IFS_ORDERMETHOD_LEN	string	(B)uy order or (S)ell order OrderMethods defines the type of the order. OrderMethods is a bit-mask. Each bit in the field indicates an order type.
BoardId	IFS_BOARDID_LEN	string	Board code

---

<sup>12</sup> This is filled only in case of appropriate license.

Name	Length	Type	Description
SecId	IFS_IDS_LEN	ids	Security code
Price	IFS_INT_LEN	int	orderentry price. Price may be zero for a market order. Because Price is an int value, the position of the decimal point is not maintained in the field. The number of decimals is obtained from the decimals field in the secboard table for the order's security
Yield	IFS_INT_LEN	int	orderentry yield
Quantity	IFS_INT_LEN	int	is the disclosed quantity of the order expressed as a number units (shares, bonds, etc).
Hidden	IFS_INT_LEN	int	is the hidden quantity of the order expressed as a number units (shares, bonds, etc).
BrokerRef	IFS_BROKERREF_LEN	string	free-format field that may be entered by the user for each orderentry
ExpDate	IFS_INT_LEN	int	Expiration date
ExpTime	IFS_INT_LEN	int	Expiration time
TriggerPrice	IFS_INT_LEN	int	Trigger price
SettleDate	IFS_INT_LEN	int	Settlement date YYYYMMDD
MinFillQty	IFS_INT_LEN	int	Minimum filled quantity
OpCode	IFS_CHAR_LEN	char	Order Boolean Operators 'A' , 'O'
PopCode	IFS_CHAR_LEN	char	Order Relational Operators '=' , '<' '>'
OUserId	IFS_IDS_LEN	ids	Other user id then the message sender.
FirmId	IFS_IDS_LEN	ids	Firm code
OrderDate	IFS_INT_LEN	int	Date of the orderentry YYYYMMDD
Status	IFS_CHAR_LEN	char	Order status handled by K2. See the Note below the table.
TransactionType <sup>13</sup>	IFS_CHAR_LEN	char	Entry, amendment, withdraw or Tick Up/Tick down. See the Note below the table.
Msg	IFS_MSG_LEN	string	The message in the badreplymsg if that exists.
InternalRef <sup>14</sup>	IFS_BROKERREF_LEN	string	Internal reference for ifs users, never sent to MMTS

Note:

Possible values in the Status field of the order entry (query) table are as follows:

- 'A' The order is accepted by the K2.
- 'C' The order is confirmed by the authorised user (order routing manager).
- 'D' The order is denied by K2 or the the order routing manager.

<sup>13</sup> See Transaction types section

<sup>14</sup> This is filled only in case of appropriate license.

- 'E' The order is successfully entered to the Trading Engine.  
'R' The order is refused by the Trading Engine.  
'U' Trading Engine's answer is unexpected or hasn't arrived yet.

Possible values in the Transaction type field of the order entry (query) table are as follows:

- 'E' A brand new order. +\*/  
'A' An attempt to modify an existing order.  
'W' Remove an existing order.  
'T' TickUpTickDown entry.

**Table 34.19. MMTS I - Market Maker Order Entry record (query)**

Note: This table can be used to check the client's own Market Maker orders entered into the K2.

Name	Length	Type	Description
orderid	IFS_INT_LEN	int	Order id given by the pgw
BuyOrdNo <sup>15</sup>	IFS_INT_LEN	int	System-assigned buy order number
SellOrdNo <sup>16</sup>	IFS_INT_LEN	int	System assigned sell order number
BoardId	IFS_BOARDID_LEN	string	Board identifier
SecId	IFS_IDS_LEN	ids	Security identifier (security code)
BuySell	IFS_CHAR_LEN	char	Buy, Sell or Both Sides
Replace	IFS_CHAR_LEN	char	If true the previous Market Maker order will be withdrawn and replaced. Replace specified by the MMTS settings will prevail against this field.
BuyTrdAccId	IFS_IDS_LEN	ids	Buy side Trading account identifier
BuyOMSet	IFS_ORDERMETHOD_LEN	string	Defines the types of Buy side orders
BuyPrice	IFS_INT_LEN	int	Price of the Buy side order
BuyYield	IFS_INT_LEN	int	Yield of the Buy side order
BuyQuantity	IFS_INT_LEN	int	Quantity of Buy side order
BuyHidden	IFS_INT_LEN	int	Not used (left empty)
BuyBrokerRef	IFS_BROKERREF_LEN	string	Free-format field that may be entered by the user for the Buy side order
SellTrdAccId	IFS_IDS_LEN	ids	Sell side Trading account identifier
SellOMSet	IFS_ORDERMETHOD_LEN	string	Defines the types of Sell side orders
SellPrice	IFS_INT_LEN	int	Price of the Sell side order
SellYield	IFS_INT_LEN	int	Yield of the Sell side order
SellQuantity	IFS_INT_LEN	int	Quantity of Sell side order

---

<sup>15</sup> This is filled only in case of appropriate license.

<sup>16</sup> This is filled only in case of appropriate license.

Name	Length	Type	Description
SellHidden	IFS_INT_LEN	int	Not used (left empty)
SellBrokerRef	IFS_BROKERREF_LEN	string	Free-format field that may be entered by the user for the Buy side order
Status	IFS_CHAR_LEN	char	Order status (Accepted, Entered, Refused by MMTS, Denied by K2, etc.). See the note below the table.
Msg	IFS_MSG_LEN	string	Error message in case of R(efused) or D(enied) status
InternalRef <sup>17</sup>	IFS_BROKERREF_LEN	string	Internal reference for ifs users, never sent to mmts

Note:

Possible values in the Status field of the Market Maker Order entry (query) table are as follows:

- 'A' The order is accepted by the K2.
- 'C' The order is confirmed by the authorised user (order routing manager).
- 'D' The order is denied by K2 or the the order routing manager.
- 'E' The order is successfully entered to the Trading Engine.
- 'R' The order is refused by the Trading Engine.
- 'U' Trading Engine's answer is unexpected or hasn't arrived yet.

Table 34.20. MMTS I - Order add record

This structure is used by client when sending the given transaction type to K2 server

Name	Length	Type	Description
TrdAccId	IFS_IDS_LEN	ids	Trading account identifier
BuySell	IFS_CHAR_LEN	char	buy or sell indicator for the order
OrderMethodSet	IFS_ORDERMETHOD_LEN	string	(B)uy order or (S)ell order OrderMethods defines the type of the order. OrderMethods is a bit-mask. Each bit in the field indicates an order type.
BoardId	IFS_BOARDID_LEN	string	Board code
SecId	IFS_IDS_LEN	ids	Security code

---

<sup>17</sup> This is filled only in case of appropriate license.



Name	Length	Type	Description
Price	IFS_INT_LEN	int	order price. Price may be zero for a market order. Because Price is a long value, the position of the decimal point is not maintained in the field. The number of decimals is obtained from the Decimals field in the Securities table for the order's security.
Yield	IFS_INT_LEN	int	order yield
Quantity	IFS_INT_LEN	int	is the disclosed quantity of the order expressed as a number units (shares, bonds, etc).
Hidden	IFS_INT_LEN	int	is the hidden quantity of the order expressed as a number units (shares, bonds, etc).
BrokerRef	IFS_BROKERREF_LEN	string	free-format field that may be entered by the user for each order
ExpDate	IFS_INT_LEN	int	Expiration date.
ExpTime	IFS_INT_LEN	int	Expiration time
TriggerPrice	IFS_INT_LEN	int	Triggered price
SettleDate	IFS_INT_LEN	int	Settlement date
MinFillQty	IFS_INT_LEN	int	Minimum filled quantity
InternalRef	IFS_BROKERREF_LEN	string	Internal reference for ifs users, never sent to MMTS

Table 34.21. MMTS I - Order cancellation record

This structure is used by client when sending the given transaction type to K2 server

Name	Length	Type	Description
OrdNo	IFS_INT_LEN	int	Order number in the MMTS
TrdAccId	IFS_IDS_LEN	ids	Trading account identifier
BuySell	IFS_CHAR_LEN	char	buy or sell indicator for the order (B)uy order or (S)ell order
BoardId	IFS_BOARDID_LEN	string	Board code
SecId	IFS_IDS_LEN	ids	Security code

Name	Length	Type	Description
Price	IFS_INT_LEN	int	order price. Price may be zero for a market order. Because Price is a long value, the position of the decimal point is not maintained in the field. The number of decimals is obtained from the Decimals field in the Securities table for the order's security.
Yield	IFS_INT_LEN	int	order yield
BrokerRef	IFS_BROKERREF_LEN	string	free-format field that may be entered by the user for each order
OpCode	IFS_CHAR_LEN	char	Order Boolean Operators 'A' , 'O'
PopCode	IFS_CHAR_LEN	char	Order Relational Operators '=', '<', '>'
OUserId	IFS_IDS_LEN	ids	Other user id then the message sender.
FirmId	IFS_IDS_LEN	ids	Firm code
OrderDate	IFS_INT_LEN	int	Date of the order YYYYMMDD
InternalRef	IFS_BROKERREF_LEN	string	Internal reference for ifs users, never sent to MMTS

Table 34.22. MMTS I - Order amendment record

This structure is used by client when sending the given transaction type to K2 server

Name	Length	Type	Description
OrdNo	IFS_INT_LEN	int	Order number in the MMTS
TrdAccId	IFS_IDS_LEN	ids	Trading account identifier
OrderMethodSet	IFS_ORDERMETHOD_LEN	string	OrderMethods defines the type of the order. OrderMethods is a bit-mask. Each bit in the field indicates an order type.
Price	IFS_INT_LEN	int	Order price. Price may be zero for a market order. Because Price is a long value, the position of the decimal point is not maintained in the field. The number of decimals is obtained from the Decimals field in the Securities table for the order's security.
Yield	IFS_INT_LEN	int	order yield
Quantity	IFS_INT_LEN	int	is the disclosed quantity of the order expressed as a number units (shares, bonds, etc).

Name	Length	Type	Description
Hidden	IFS_INT_LEN	int	is the hidden quantity of the order expressed as a number units (shares, bonds, etc).
BrokerRef	IFS_BROKERREF_LEN	string	free-format field that may be entered by the user for each order
ExpDate	IFS_INT_LEN	int	Expiration date.
ExpTime	IFS_INT_LEN	int	Expiration time
TriggerPrice	IFS_INT_LEN	int	Triggered price
SettleDate	IFS_INT_LEN	int	Settlement date
MinFillQty	IFS_INT_LEN	int	Minimum filled quantity
OUserIdx	IFS_IDS_LEN	ids	Other user id then the message sender
InternalRef	IFS_BROKERREF_LEN	string	Internal reference for ifs users, never sent to MMTS

Table 34.23. MMTS I - Tick up/tick down

This structure is used by client when sending the given transaction type to K2 server

Name	Length	Type	Description
TrdAccId	IFS_IDS_LEN	ids	Trade account identifier
BuySell	IFS_CHAR_LEN	char	Buy, Sell or Both Sides
BoardId	IFS_BOARDID_LEN	string	Board identifier
SecId	IFS_IDS_LEN	ids	Security identifier (security code)
Tick <sup>18</sup>	IFS_INT_LEN	int	Number of price ticks (number of standard price steps; increments or decrements by a given tick size). Tick sizes may vary for each security or board.
UserId	IFS_IDS_LEN	ids	User identifier
FirmId	IFS_IDS_LEN	ids	Firm identifier
BrokerRef	IFS_BROKERREF_LEN	string	Free-format field that may be entered by the user for each order
InternalRef	IFS_BROKERREF_LEN	string	Internal reference for ifs users, never sent to mmts

Table 34.24. MMTS I - Market Maker Order entry

This structure is used by client when sending the given transaction type to K2 server

Name	Length	Type	Description
BoardId	IFS_BOARDID_LEN	string	Board identifier
SecId	IFS_IDS_LEN	ids	Security identifier (security code)
BuySell	IFS_CHAR_LEN	char	Buy, Sell or Both Sides

<sup>18</sup> If it is negative the price will be decreased by the given number of ticks.

Name	Length	Type	Description
Replace	IFS_CHAR_LEN	char	If true the previous Market Maker order will be withdrawn and replaced. Replace specified by the MMTS settings will prevail against this field.
BuyTrdAccId	IFS_IDS_LEN	ids	Buy side Trading account identifier
BuyOMSet	IFS_ORDERMETHOD_LEN	string	Defines the types of Buy side order
BuyPrice	IFS_INT_LEN	int	Price of the Buy side order
BuyYield	IFS_INT_LEN	int	Yield of the Buy side order
BuyQuantity	IFS_INT_LEN	int	Quantity of Buy side order
BuyHidden	IFS_INT_LEN	int	Not used (left empty)
BuyBrokerRef	IFS_BROKERREF_LEN	string	Free-format field that may be entered by the user for the Buy side order
SellTrdAccId	IFS_IDS_LEN	ids	Sell side Trading account identifier
SellOMSet	IFS_ORDERMETHOD_LEN	string	Defines the types of Sell side order
SellPrice	IFS_INT_LEN	int	Price of the Sell side order
SellYield	IFS_INT_LEN	int	Yield of the Sell side order
SellQuantity	IFS_INT_LEN	int	Quantity of Sell side order
SellHidden	IFS_INT_LEN	int	Not used (left empty)
SellBrokerRef	IFS_BROKERREF_LEN	string	Free-format field that may be entered by the user for the Buy side order
InternalRef	IFS_BROKERREF_LEN	string	Internal reference for ifs users, never sent to mmts

**35. Structure of the records - MMTS II.**

---

Table 35.1. MMTS II - Market record

Name	Length	Type	Description
Id	IFS_IDS_LEN	ids	Market code
Name	IFS_NAME_LEN	string	Name of the market
Status	IFS_ENUM_LEN	enum	current status of the market: (A)ctive, (S)uspended or (D)efunct

Table 35.2. MMTS II - Instrument record

Name	Length	Type	Description
Id	IFS_IDS_LEN	ids	Instrument code
Name	IFS_NAME_LEN	string	Name of the instrument
Status	IFS_ENUM_LEN	enum	current status of the market: (A)ctive, (S)uspended or (D)efunct
SecClassId	IFS_ENUM_LEN	enum	Security class identifier
1. Leg InstrId	IFS_IDS_LEN	ids	Leg identifier for spread
2. Leg InstrId	IFS_IDS_LEN	ids	Leg identifier for spread

Table 35.3. MMTS II - Sector record

Name	Length	Type	Description
Id	IFS_IDS_LEN	ids	Sector identifier.
Name	IFS_NAME_LEN	string	Sector name.

Table 35.4. MMTS II - Board record

Name	Length	Type	Description
Id	IFS_IDS_LEN	string	Board identifier.
Name	IFS_NAME_LEN	string	Board name.
Status	IFS_ENUM_LEN	enum	Board status.
Ordermethods	IFS_ORDERMETHODS_LEN	string	OrderMethods defines the types of orders that will be accepted for the securities. OrderMethods is a bit-mask. Each bit in the field indicates one type of order. If the bit is set to 1, then that type of order is accepted.

Table 35.5. MMTS II - Secboard record

Name	Length	Type	Description
Id	IFS_SECBOARDID_LEN	string	Secboard identifier e.g.
SecName	IFS_SEC_NAME_LEN	string	Security name.

Name	Length	Type	Description
SecShortName	IFS_SEC_SHORTNAME_LEN	string	Security short name
MarketId	IFS_IDS_LEN	ids	Index to the market table
InstrId	IFS_IDS_LEN	ids	Index to the instrument table
SectorId	IFS_IDS_LEN	ids	Index to the sector table.
CurrName	IFS_IDS_LEN	ids	Currency name e.g. HUF, USD
FaceValue	IFS_FIXREAL_LEN	fixreal	Face value of the currency.
PrevEarn	IFS_FIXREAL_LEN	fixreal	Value of the previous earning
Eps	IFS_FIXREAL_LEN	fixreal	PrevEarn / IssuedQty
VolumeTraded	IFS_DOUBLE_LEN	double	Total volume
ValueTraded	IFS_FIXREAL_LEN	fixreal	Total value
OrigIssueQty	IFS_DOUBLE_LEN	double	Original issued size
IssuedQty	IFS_DOUBLE_LEN	double	Issued size. MMTS1: ListedQty
TradeableSize	IFS_DOUBLE_LEN	double	Tradeable size
PriceDecimals	IFS_INT_LEN	int	Price decimals.
LotSize	IFS_INT_LEN	int	Lot size.
IsIndex	IFS_ENUM_LEN	bool	Index Sec
IssuerId	IFS_IDS_LEN	ids	Issuer identifier
Isin	IFS_IDS_LEN	ids	ISIN code
ListingType	IFS_IDS_LEN	ids	Listing type
YieldDecimals	IFS_INT_LEN	int	Yield decimals
SecClassId	IFS_ENUM_LEN	enum	SecClass Id.
NegDealInitiator	IFS_ENUM_LEN	enum	NegDeal initiator
ExpiryDate	IFS_INT_LEN	int	Holds the the last trading date instead
UnderlyingId	IFS_SECBOARDID_LEN	string	Underlying secboard idx
OptionStrikePrice	IFS_FIXREAL_LEN	fixreal	Option strike
OptionVerb	IFS_ENUM_LEN	enum	Option verb
IsUnderlyingToOption	IFS_ENUM_LEN	bool	Is underlying to options
SpreadLeg1SecBoardId	IFS_SECBOARDID_LEN	string	Secboard id of spreadLeg 1.
SpreadLeg2SecBoardId	IFS_SECBOARDID_LEN	string	Secboard id of spreadLeg 2.

Changeable fields

Name	Length	Type	Description
Remarks	IFS_SEC_REMARK_LEN	string	Remarks about the security
SecState	IFS_CHAR_LEN	char	Security state, active or suspended.
BidPrice	IFS_FIXREAL_LEN	fixreal	Bid price
BidDepth	IFS_INT_LEN	int	Bid depth
BidDepthT	IFS_INT_LEN	int	Best bid depth
BidN	IFS_INT_LEN	int	Number of bid orders
OfferPrice	IFS_FIXREAL_LEN	fixreal	Best offer price
OfferDepth	IFS_INT_LEN	int	Offer depth
OfferDepthT	IFS_INT_LEN	int	Best offer depth
OfferN	IFS_INT_LEN	int	Number of offer orders
openPrice	IFS_FIXREAL_LEN	fixreal	Open price
highPrice	IFS_FIXREAL_LEN	fixreal	High price
lowPrice	IFS_FIXREAL_LEN	fixreal	Low price
lastTradedPrice	IFS_FIXREAL_LEN	fixreal	Last traded price
lastOffMktPrice	IFS_FIXREAL_LEN	fixreal	Last off market price
changePrice	IFS_FIXREAL_LEN	fixreal	Price change
Qty	IFS_INT_LEN	int	Last quantity
Time	IFS_INT_LEN	int	Last time
volumeToday	IFS_DOUBLE_LEN	double	Daily traded volume
valueToday	IFS_FIXREAL_LEN	fixreal	Daily traded value

Name	Length	Type	Description
OrderMethods	IFS_ORDERMETHOD_LEN	string	OrderMethods defines the types of orders that will be accepted for the security. OrderMethods is a bit-mask. Each bit in the field indicates one type of order. If the bit is set to 1, then that type of order is accepted.
changeLTP	IFS_FIXREAL_LEN	fixreal	Last trade price change
TradeDate	IFS_INT_LEN	int	TradeDate as integer in YYYYMMDD format. (not operating system date!)
PrevDate	IFS_INT_LEN	int	last date on which the secboard traded before today
PrevPrice	IFS_FIXREAL_LEN	fixreal	last price at which the security traded before today. This is separately maintained for each security on each board
SessionName	IFS_NAME_LEN	string	Name of the current session
Value	IFS_FIXREAL_LEN	fixreal	Last value
lastTradedYield	IFS_FIXREAL_LEN	fixreal	Last traded yield
qtyOffMkt	IFS_INT_LEN	int	Off market last quantity
RefPrice	IFS_FIXREAL_LEN	fixreal	Reference price
NumTrades	IFS_INT_LEN	int	Number of trades
NumOrders	IFS_INT_LEN	int	Number of orders
NumOpenOrders	IFS_INT_LEN	int	Number of open orders
changePricePct	IFS_FIXREAL_LEN	fixreal	Price change in percentage
WAPrice	IFS_FIXREAL_LEN	fixreal	Weighted average price
ChangeStateTime	IFS_DATETIME_LEN	datetime	Secboard change time
AnnounceInd	IFS_ENUM_LEN	Bool	Has news
UpperPriceLimit	IFS_FIXREAL_LEN	fixreal	Upper price limit
LowerPriceLimit	IFS_FIXREAL_LEN	fixreal	Lower price limit
CBLimitUpper	IFS_FIXREAL_LEN	fixreal	Upper circuit breaker limit
CBLimitLower	IFS_FIXREAL_LEN	fixreal	Lower circuit breaker limit
ClosePrice	IFS_FIXREAL_LEN	fixreal	Closing price
VolOffMktToday	IFS_DOUBLE_LEN	double	Daily traded off market volume
ValOffMktToday	IFS_FIXREAL_LEN	fixreal	Daily traded off market value
OpenYield	IFS_FIXREAL_LEN	fixreal	Open yield
HighYield	IFS_FIXREAL_LEN	fixreal	High yield
LowYield	IFS_FIXREAL_LEN	fixreal	Low yield
CloseYield	IFS_FIXREAL_LEN	fixreal	Close yield
RefYield	IFS_FIXREAL_LEN	fixreal	Reference yield
ChangeYield	IFS_FIXREAL_LEN	fixreal	Change yield
ChangeLty	IFS_FIXREAL_LEN	fixreal	Change last traded yield
LastOffMktYield	IFS_FIXREAL_LEN	fixreal	Last off market yield
BidYield	IFS_FIXREAL_LEN	fixreal	Bid yield
OfferYield	IFS_FIXREAL_LEN	fixreal	Offer yield
WAYield	IFS_FIXREAL_LEN	fixreal	Weighted average yield
PrevYield	IFS_FIXREAL_LEN	fixreal	Previous yield
AccruedInterest	IFS_FIXREAL_LEN	fixreal	current accrued interest per unit for fixed interest securities
QuoteBases	IFS_QUOTEBASES_LEN	string	quote bases field for a security
InheritedStatus	IFS_ENUM_LEN	enum	Inherited status
ImpliedBidDepth	IFS_INT_LEN	int	Implied bid depth
ImpliedOfferDepth	IFS_INT_LEN	int	Implied offer depth

Name	Length	Type	Description
cbRefPrice	IFS_FIXREAL_LEN	fixreal	Circuit breaker reference price
cbRefYield	IFS_FIXREAL_LEN	fixreal	Circuit breaker reference yield
OpenInterest	IFS_DOUBLE_LEN	double	Open interest
StrikeValueToday	IFS_FIXREAL_LEN	fixreal	Strike value today
SettlementPrice	IFS_FIXREAL_LEN	fixreal	Settlement price
SettlementPriceType	IFS_ENUM_LEN	enum	Type of the settlement price
IndicativePrice	IFS_FIXREAL_LEN	fixreal	Indicative price
IndicativeVolume	IFS_DOUBLE_LEN	double	Indicative volume
MarketCap	IFS_FIXREAL_LEN	fixreal	Last traded price * issued qty
UnderlyingLastPrice	IFS_FIXREAL_LEN	fixreal	Underlying price
FixingPrice	IFS_FIXREAL_LEN	fixreal	Fixing price

Table 35.6. MMTS II - User record

Name	Length	Type	Description
Id	IFS_IDS_LEN	ids	User identifier
Name	IFS_NAME_LEN	string	User name (real)
FirmId	IFS_IDS_LEN	ids	Firm identifier
DefaultRoleId	IFS_IDS_LEN	ids	Default role identifier
Status	IFS_ENUM_LEN	enum	Status of the user
ApproveOrders	IFS_ENUM_LEN	bool	His orders must be approved
ContactDetail	IFS_NAME_LEN	string	Contact detail
FreeText	IFS_FREE_TEXT_LEN	string	Text of the free
IPGateway	IFS_IPADDR_LEN	string	network address of the Gateway through which the user has logged into the Trading Engine. This address is automatically detected by the Trading Engine when the user logs into the System. For a user who is connected directly to the Trading Engine and does not route through a Gateway, <b>IPGateway</b> is the actual network address from which the user has logged on, and <b>IPClient</b> will be zero
IPClient	IFS_IPADDR_LEN	string	network address from which the user has logged into the Trading Engine (if the user is currently logged in). This address is automatically detected by the Trading Engine when the user logs into the System. For a user who is connected directly to the Trading Engine and does not route through a Gateway, <b>IPClient</b> will be zero and <b>IPGateway</b> is the network address from which the user has logged on.
IsLoggedIn	IFS_ENUM_LEN	bool	Logged flag
Today buy order			
SingleValue	IFS_FIXREAL_LEN	fixreal	Highest value (price*quantity) for a single order (open or matched)
TotalValue	IFS_FIXREAL_LEN	fixreal	Total value (price*quantity) for all orders (open or matched)



Today sell order			
SingleValue	IFS_FIXREAL_LEN	fixreal	Highest value (price*quantity) for a single order (open or matched)
TotalValue	IFS_FIXREAL_LEN	fixreal	Total value (price*quantity) for all orders (open or matched)
Today buy trade			
SingleValue	IFS_FIXREAL_LEN	fixreal	Highest value (price*quantity) for a single trade
TotalValue	IFS_FIXREAL_LEN	fixreal	Total value (price*quantity) for all trades
Today sell trade			
SingleValue	IFS_FIXREAL_LEN	fixreal	Highest value (price*quantity) for a single trade
TotalValue	IFS_FIXREAL_LEN	fixreal	Total value (price*quantity) for all trades

Table 35.7. MMTS II - Firm record

Name	Length	Type	Description
Id	IFS_IDS_LEN	ids	Firm identifier
Name	IFS_NAME_LEN	string	Firm real name.
Status	IFS_ENUM_LEN	enum	Status of the firm. Active or suspended
Type	IFS_IDS_LEN	ids	Firm type
ContactDetail	IFS_NAME_LEN	sting	Contact detail
FreeText	IFS_FREE_TEXT_LEN	sting	Free text
FirmClass	IFS_ENUM_LEN	enum	Firm class computed from type
UsersLoggedIn	IFS_INT_LEN	int	Number of users logged in
CloseOutOnly	IFS_ENUM_LEN	enum	CloseOut Status
ClearingFirmId	IFS_IDS_LEN	ids	Clearing Firm Name
ClearingCode	IFS_CLEARINGCODE_LEN	string	Clearing Code

Today buy order			
SingleValue	IFS_FIXREAL_LEN	fixreal	Highest value (price*quantity) for a single order (open or matched)
TotalValue	IFS_FIXREAL_LEN	fixreal	Total value (price*quantity) for all orders (open or matched)
Today sell order			
SingleValue	IFS_FIXREAL_LEN	fixreal	Highest value (price*quantity) for a single order (open or matched)
TotalValue	IFS_FIXREAL_LEN	fixreal	Total value (price*quantity) for all orders (open or matched)
Today buy trade			
SingleValue	IFS_FIXREAL_LEN	fixreal	Highest value (price*quantity) for a single trade
TotalValue	IFS_FIXREAL_LEN	fixreal	Total value (price*quantity) for all trades

Today sell trade			
SingleValue	IFS_FIXREAL_LEN	fixreal	Highest value (price*quantity) for a single trade
TotalValue	IFS_FIXREAL_LEN	fixreal	Total value (price*quantity) for all trades

Table 35.8. MMTS II - Order record

Name	Length	Type	Description
OrdNo	IFS_ORDERNO_LEN	string	Order number
OrdNoSpeedIdx	IFS_INT_LEN	int	Order Idx in TE. Speed up order delete and amend
OrderTime	IFS_DATETIME_MSEC_LEN	datetime msec	Date and time of the order entry, extended with milliseconds
OrderStatus	IFS_ENUM_LEN	enum	Order status in MMTS. Open, Matched, Withdrawn etc.
BuySell	IFS_ENUM_LEN	enum	Buy or Sell
BrokerRef	IFS_BROKERREF_LEN	string	Broker reference, free text. Upto 40 character long but truncated to 20 ch overnight for expiries longer than one day
UserId	IFS_IDS_LEN	ids	User index of trader
FirmId	IFS_IDS_LEN	ids	Firm index of trader
SecBoardId	IFS_SECBOARDID_LEN	string	Secboard index
InstrId	IFS_IDS_LEN	ids	Intrument identifier
TrdAccId	IFS_IDS_LEN	ids	Trading account identifier
ExecutionId	IFS_USER_MIFIDID_LEN	string	The ID of the executor (user or algorithm)
Price	IFS_FIXREAL_LEN	fixreal	Price
Yield	IFS_FIXREAL_LEN	fixreal	Yield
TotalQuantity	IFS_INT_LEN	int	Total quantity = visible qty + hidden.
VisibleQuantity	IFS_INT_LEN	int	Visible quantity
Balance	IFS_INT_LEN	int	Balance quantity. Quantity - Matched quantity
Value	IFS_FIXREAL_LEN	fixreal	Value. Quantity * Price
Type	IFS_ENUM_LEN	enum	From OrderMethods
FillType	IFS_ENUM_LEN	enum	From OrderMethods
Duration	IFS_ENUM_LEN	enum	From OrderMethods
PurgeOnLogoff	IFS_ENUM_LEN	bool	From OrderMethods
IsMarketMaker	IFS_ENUM_LEN	bool	From OrderMethods
PrevOrdNo	IFS_ORDERNO_LEN	string	Previous order number. (In case of amendment.)

Name	Length	Type	Description
OriginalOrderId	IFS_ORDERNO_LEN	string	Head of the prevlinkorder list.
ExpTime	IFS_DATETIME_LEN	datetime	Expiration date
TriggerPrice	IFS_FIXREAL_LEN	fixreal	Trigger price
ValueMatched	IFS_FIXREAL_LEN	fixreal	Matched value
MinFillQty	IFS_INT_LEN	int	Minimal fill quantity
AveragePrice	IFS_FIXREAL_LEN	fixreal	ValueMatched / (Qty + Hidden - balance)
PositionType	IFS_ENUM_LEN	enum	Type of the position OpeningTrade ClosingTrade DayTrade
TradeReference	IFS_NAME_LEN	string	Reference text of the trade
AccruedInterest	IFS_FIXREAL_LEN	fixreal	Accrued Interest of the order
MultilegRatio	IFS_INT_LEN	int	Multileg orders quantity ratio
MultilegOrdNo	IFS_ORDERNO_LEN	string	Multileg orders number
MultilegSpeedIdx	IFS_INT_LEN	int	Idx of multilegid in mmts

Table 35.9. MMTS II - Trade record

Name	Length	Type	Description
TrdNo	IFS_TRADENO_LEN	string	Trade number
BuyOrdNo	IFS_ORDERNO_LEN	string	Buy order number. This field is completed only for own orders (own trades).
SellOrdNo	IFS_ORDERNO_LEN	string	Sell order number. This field is completed only for own orders (own trades).
TradeTime	IFS_DATETIME_MSEC_LEN	datetime msec	Time of the trade, extended with milliseconds.
AmendTime	IFS_DATETIME_MSEC_LEN	datetime msec	time at which the trade was cancelled (if it is cancelled) , extended with milliseconds.
BuyBrokerRef	IFS_BROKERREF_LEN	string	Buy broker reference, free text. Upto 40 character long but may be truncated to 20 ch if the trade is generated from an order older than one day

Name	Length	Type	Description
SellBrokerRef	IFS_BROKERREF_LEN	string	Sell broker reference, free text. Upto 40 character long but may be truncated to 20 ch if the trade is generated from an order older than one day
BuyUserId	IFS_IDS_LEN	ids	Buy user index
SellUserId	IFS_IDS_LEN	ids	Sell user index
BuyFirmId	IFS_IDS_LEN	ids	Buy firm index
SellFirmId	IFS_IDS_LEN	ids	Sell firm index
BuyTrdAccId	IFS_IDS_LEN	ids	Buy trading account
SellTrdAccId	IFS_IDS_LEN	ids	Sell trading account
BuyExecutionId	IFS_USER_MIFIDID_LEN	string	The ID of the buy executor (user or algorithm)
SellExecutionId	IFS_USER_MIFIDID_LEN	string	The ID of the sell executor (user or algorithm)
SecBoardId	IFS_SECBOARDID_LEN	string	Secboard identifier
InstrId	IFS_IDS_LEN	ids	Instrument identifier
Price	IFS_FIXREAL_LEN	fixreal	trade price. The number of decimals is obtained from the Decimals field in the SecBoard table for the trade's security.
Yield	IFS_FIXREAL_LEN	fixreal	trade value expressed as yield
Quantity	IFS_INT_LEN	int	trade quantity expressed as a number units (shares, bonds, etc).
Value	IFS_FIXREAL_LEN	fixreal	Value of the trade
StrikeValue	IFS_FIXREAL_LEN	fixreal	Value of the strike trade
BuyTax	IFS_FIXREAL_LEN	fixreal	tax for the buy side of trade
SellTax	IFS_FIXREAL_LEN	fixreal	tax for the sell side of trade
BuyFee	IFS_FIXREAL_LEN	fixreal	Fee of the buy
SellFee	IFS_FIXREAL_LEN	fixreal	Fee of the sell
AccruedInterest	IFS_FIXREAL_LEN	fixreal	Accrued interest
PriceChange	IFS_FIXREAL_LEN	fixreal	Change of the price
TradeStatus	IFS_ENUM_LEN	enum	current status of the trade
TradeSource	IFS_CHAR_LEN	char	A or F automated or fixed.
DissemTime	IFS_DATETIME_LEN	datetime	Negdeal Disseminate time
DissemStatus	IFS_ENUM_LEN	enum	Negdeal Disseminate status

Table 35.10. MMTS II - Negotiated deal record

Name	Length	Type	Description
NegDealNo	IFS_ORDERNO_LEN	string	Order Id
CPNegDealNo	IFS_ORDERNO_LEN	string	CounterParty order Id
Time	IFS_DATETIME_LEN	datetime	Entry Time
OrderStatus	IFS_ENUM_LEN	enum	current status of the order in MMTS (e.g. Unconfirmed, Matched, Withdrawn, etc)
BuySell	IFS_ENUM_LEN	enum	buy or sell indicator for the order (B)uy order or (S)ell order
BrokerRef	IFS_BROKERREF_LEN	string	free-format field that may be entered by the user for each order
UserId	IFS_IDS_LEN	ids	User code
FirmId	IFS_IDS_LEN	ids	Firm code
CPUserId	IFS_IDS_LEN	ids	Counter Party user code
CPFirmId	IFS_IDS_LEN	ids	Counter Party firm code
TrdAccId	IFS_IDS_LEN	ids	Trading account identifier
SecBoardId	IFS_SECBOARDID_LEN	string	secboard code
InstrId	IFS_IDS_LEN	ids	
Price	IFS_FIXREAL_LEN	fixreal	Order price. The number of decimals is obtained from the Decimals field in the Secboard table for the order's security.
Quantity	IFS_INT_LEN	int	is the quantity of the order expressed as a number units (shares, bonds, etc).
SettleDate	IFS_INT_LEN	int	Settlement date
Timeout	IFS_INT_LEN	int	Time in seconds that is allowed for negotiated deals to be confirmed. After this time, the users involved in the deal will be sent messages from the Trading System to remind them that their deal is unconfirmed
Yield	IFS_FIXREAL_LEN	fixreal	Yield of the order
DissemTime	IFS_DATETIME_LEN	datetime	Disseminate time
DissemStatus	IFS_ENUM_LEN	enum	Disseminate status (e.g. Delayed, Sent, Released, etc.)

Name	Length	Type	Description
TradeSource	IFS_CHAR_LEN	char	F for negotiated deals (fix orders).
WarningTimeout	IFS_INT_LEN	int	If not an unconfirmed neg. deal without a reminder, where neg. deals do have a confirmation warningTimeout
OrderMethods	IFS_ORDERMETHOD_LEN	string	OrderMethods field defines the type of the order. OrderMethods is a bit-mask. Each bit in the field indicates one type of order.
PositionType	IFS_ENUM_LEN	enum	Type of the position opening trade, closing trade, daytrade
TradeReference	IFS_NAME_LEN	string	Free format field that can be used for trade reference in case of close or daytrade PositionType (allocation)

Table 35.11. MMTS II - Audit Event record

Name	Length	Type	Description
EventNum	IFS_INT_LEN	int	Sequence no. of the event message
EventGroup	IFS_CAPTION_LEN	string	Event Group (e.g. Market Maker group)
EventType	IFS_CAPTION_LEN	string	Event type (e.g. breach of obligation)
Time	IFS_DATETIME_LEN	datetime	Time of sending
DispMode	IFS_ENUM_LEN	enum	Mode of display (e.g. popup or status line)
ToUserId	IFS_IDS_LEN	ids	Target UserId of the message
ToFirmId	IFS_IDS_LEN	ids	Target FirmId of the message
FromUserId	IFS_IDS_LEN	ids	UserId of sender (empty in case of automatic messages sent by MMTS II)
ToWhat	IFS_ENUM_LEN	enum	Kind of the entity the event is related to (e.g. security, board or secboard)

Name	Length	Type	Description
ToWhatId	IFS_SECBOARDID_LEN	int	Identifier of the entity the event is related to
ToWhom	IFS_ENUM_LEN	enum	Kind of the target entity (e.g. firm or user)
ToWhomId	IFS_IDS_LEN	ids	Identifier of the target user or firm
EventText	IFS_MSG_LEN	string	Text of the message

Table 35.12. MMTS II - Order book by order record

Name	Length	Type	Description
SecBoardId	IFS_SECBOARDID_LEN	string	Security board of the orderbook. Each security board may have only one orderbook
Occupied <sup>19</sup>	IFS_CHAR_LEN	char	If Y then this orderbook is available for usage. If N then the order book is not on the monitoring list.
NumBuys	IFS_INT_LEN	int	Number of rows on the buy side
NumSells	IFS_INT_LEN	int	Number of rows on the sell side
OrderId <sup>20</sup>	IFS_ORDERNO_LEN	string	MMTS order no.
BuySell	IFS_ENUM_LEN	enum	buy or sell indicator for the order (B)uy order or (S)ell order
Price	IFS_FIXREAL_LEN	fixreal	Price of the order
Yield	IFS_FIXREAL_LEN	fixreal	Yield of the order
Qty	IFS_INT_LEN	int	Balance of the order. (Open volume.)
FirmId	IFS_IDS_LEN	ids	Firm identifier
UserId	IFS_IDS_LEN	ids	User identifier
Implied	IFS_ENUM_LEN	bool	If it is an implied order
Hidden	IFS_ENUM_LEN	bool	If the order has Hidden part
MarketMaker	IFS_ENUM_LEN	bool	If it is a Market Maker order
OrderId <sup>21</sup>	IFS_ORDERNO_LEN	string	MMTS order no.

<sup>19</sup> K2 observing status: 'Y'<sup>20</sup> part repeated **NumBuys** times.<sup>21</sup> part repeated **NumSells** times.

Name	Length	Type	Description
BuySell	IFS_ENUM_LEN	enum	buy or sell indicator for the order (B)uy order or (S)ell order
Price	IFS_FIXREAL_LEN	fixreal	Price of the order
Yield	IFS_FIXREAL_LEN	fixreal	Yield of the order
Qty	IFS_INT_LEN	int	Balance of the order. (Open volume.)
FirmId	IFS_IDS_LEN	ids	Firm identifier
UserId	IFS_IDS_LEN	ids	User identifier
Implied	IFS_ENUM_LEN	bool	If it is an implied order
Hidden	IFS_ENUM_LEN	bool	If the order has Hidden part
MarketMaker	IFS_ENUM_LEN	bool	If it is a Market Maker order

Table 35.13. MMTS II - Order book by price record

Name	Length	Type	Description
SecBoardId	IFS_SECBOARDID_LEN	string	Security board of the orderbook. Each security board may have only one orderbook
Occupied <sup>18</sup>	IFS_CHAR_LEN	char	If Y then this orderbook is available for usage. If N then the order book is not on the monitoring list.
OtherNOrder	IFS_INT_LEN	int	Number of orders on other boards
NumBuys	IFS_INT_LEN	int	Number of rows on the buy side
NumSells	IFS_INT_LEN	int	Number of rows on the sell side
Price <sup>24</sup>	IFS_FIXREAL_LEN	fixreal	Buy price level of row
Yield	IFS_FIXREAL_LEN	fixreal	Buy yield level of row
Qty	IFS_INT_LEN	int	Buy Balance on the price level row. (Open volume.)
UserQty	IFS_INT_LEN	int	Buy Quantity of the current user on the row.
VOrders	IFS_INT_LEN	int	Number of visible buy orders on the row
VFirms	IFS_INT_LEN	int	No of firms placing Visible Buy orders on the given row
MM	IFS_ENUM_LEN	bool	'M' if there is marker maker order on the row
Hidden	IFS_ENUM_LEN	bool	Hidden flag. It is 'H' if any order on the row has hidden part.



Name	Length	Type	Description
Flag	IFS_ENUM_LEN	bool	User has order in row = '*', User has no order in row = 'N', User has best order in row = 'I'
Price <sup>25</sup>	IFS_FIXREAL_LEN	fixreal	Sell price level of row
Yield	IFS_FIXREAL_LEN	fixreal	Sell yield level of row
Qty	IFS_INT_LEN	int	Sell Balance on the price level row. (Open volume.)
UserQty	IFS_INT_LEN	int	Sell Quantity of the current user on the row.
VOrders	IFS_INT_LEN	int	Number of visible sell orders on the row
VFirms	IFS_INT_LEN	int	No of firms placing Visible Sell orders on the given row
MM	IFS_ENUM_LEN	bool	'M' if there is marker maker order on the row
Hidden	IFS_ENUM_LEN	bool	Hidden flag. It is 'H' if any order on the row has hidden part.
Flag	IFS_ENUM_LEN	bool	User has order in row = '*', User has no order in row = 'N', User has best order in row = 'I'

Table 35.14. MMTS II - Order book list

Name	Length	Type	Description
SecBoardId <sup>22</sup>	IFS_SECBOARDID_LEN	string	Secboard identifier

Table 35.15. MMTS II - System time

Name	Length	Type	Description
TradeDate	IFS_INT_LEN	int	TradeDate as integer in YYYYMMDD format. (not operating system date!) +*/
TETime <sup>23</sup>	IFS_INT_LEN	int	TradeTime as integer in HHMMSS format. (not operating system time!)
K2Date	IFS_INT_LEN	int	K2 system date

<sup>22</sup> Variable length record, this field can be repeated several times.<sup>23</sup> Value of trading system clock in (HHMMSS) format.

Name	Length	Type	Description
K2Time <sup>24</sup>	IFS_INT_LEN	int	K2 system time
K2QueryTimeOffset	IFS_INT_LEN	int	Time offset to the Trading engine (seconds since the Epoch). TE time = System Time + Offset Time +*/
PgwState	IFS_INT_LEN	int	state of the Pgw process. 0 is start state, 1 is termination.

Table 35.16. MMTS II - Field changes of secboard table

Name	Length	Type	Description
Id	IFS_SECBOARDID_LEN	string	Secboard identifier
FieldId <sup>25</sup> + FieldData	1 + IFS_???_LEN		One byte FieldId followed by Field's data content (This is repeated several times according to the number of fields changed.)

Note:

The field changes of secboard table can be queried by the **ifsc\_get\_next\_field\_chg** function.

```
int ifsc_get_next_field_chg(IFSC_HDL handle,
                           char ** rec_buf,
                           int * len,
                           int table_code,
                           int * seqno);
```

The function returns the length of the received string in the len parameter. By decrementing this returned length value after reading each field by (1+IFS\_???\_LEN) programmers can determine if there are any remaining field changes in the received string.

Table 35.17. MMTS II – Secboard additional information

This table is used to query additional static information on the given secboard. The data are static and they will not change during the trading day, but they may change from one trading day to another.

Name	Length	Type	Description
Id	IFS_SECBOARDID_LEN	string	Secboard identifier
PriceParamArray	IFS_PRICEPARAM_LEN	string	Price step (tick size) information (see below the table)
MinQty	IFS_INT_LEN	int	Minimum quantity size requirement for an order
PointValue	IFS_INT_LEN	int	Contract size

<sup>24</sup> Value of computer clock which K2 running in. The format is (HHMMSS).

<sup>25</sup> The field codes listed in `fields.h`.

**PriceParamArray (MMTS II) explanation**

This field defines the price step (tick size) for a given security & board pair. The tick size can be defined in ranges, and additionally it can be defined as an exact value or as a percentage of the order's real price . The price parameter structure is as follows:

L|V|Pdec|n:m[,n:m] ...

Where:

- L is "P" or "Y" and indicates whether the lookup range refers to a price or a yield;
- V is "D", "P" or "Q", indicating whether the parameter value defaults to price/yield (as per the first parameter), or is a percentage or quantity;
- Pdec is a single digit integer defining the number of decimal places (pricedecimals) for price values in general.
- [n:m] are pairs of values that make up the array where n is the starting price/yield of the lookup range and m is the parameter value that applies to the range

The first range always starts at zero and ends at the start of the following range. The last range ends at infinity.

e.g. P|D|2|0:0.01,1.50:0.05

Meaning a price based rule where the parameters for the price ranges are:

Range  $\geq 0$  and  $< 1.50 = 0.01$

Range  $\geq 1.50 = 0.05$

**Table 35.18. MMTS II - Order entry record (query)**

Note: This table can be used to check the client's own orders entered into the K2.

Name	Length	Type	Description
orderid	IFS_INT_LEN	int	Order id given by the pgw
OrdNo <sup>26</sup>	IFS_ORDERNO_LEN	string	Order number in the mmts
OrdNoSpeedIdx	IFS_INT_LEN	int	OrderIdx in the mmts
TrdAccId	IFS_IDS_LEN	ids	Trading Account identifier
ExecutionId	IFS_USER_MIFIDID_LEN	string	The ID of the executor (user or algorithm)
BuySell	IFS_ENUM_LEN	enum	buy or sell indicator for the order (B)uy order or (S)ell order
OrderType	IFS_ENUM_LEN	enum	Limit or market

---

<sup>26</sup> This is filled only in case of appropriate license.

Name	Length	Type	Description
Duration	IFS_ENUM_LEN	enum	immediate, session, day, goodtilldate, goodtilltime or goodtillcancelled
PurgeOnLogoff	IFS_ENUM_LEN	bool	Purge order or not when the K2 user is logged off
AllowSoftQtyLimit	IFS_ENUM_LEN	bool	Allow Soft Quantity Limit
AllowSoftPriceLimit	IFS_ENUM_LEN	bool	Allow Soft Price Limit
PositionType	IFS_ENUM_LEN	enum	opening trade, closing trade, daytrade
IsPrivate	IFS_ENUM_LEN	bool	True for private orders sent to the POB. Only MMTS_FALSE is allowed.
BoardId	IFS_BOARDID_LEN	string	Board identifier
SecId	IFS_SEC_CODE_LEN	string	Security identifier (security code)
Price	IFS_FIXREAL_LEN	fixreal	Price
Yield	IFS_FIXREAL_LEN	fixreal	Yield
Quantity	IFS_INT_LEN	int	Total quantity = visible qty + hidden.
VisibleQty	IFS_INT_LEN	int	Visible quantity
BrokerRef	IFS_BROKERREF_LEN	string	free-format field that may be entered by the user for each order
ExpTime	IFS_DATETIME_LEN	datetime	Order Expiry time
TriggerPrice	IFS_FIXREAL_LEN	fixreal	Triggered price
TradeRef	IFS_NAME_LEN	string	Trade reference text for allocation
MinFillQty	IFS_INT_LEN	int	Minimum Fill Quantity
OpCode	IFS_ENUM_LEN	enum	And, Or, Not
PopCode	IFS_ENUM_LEN	enum	EQ, NT, GT, GE, LT, LE
MultiLegOrdNo	IFS_ORDERNO_LEN	string	OrderNo of multilegid
MultilegSpeedIdx	IFS_INT_LEN	int	Idx of multilegid in mmts
InstrId	IFS_IDS_LEN	ids	Instrument identifier
UserId	IFS_IDS_LEN	ids	Trader user id
FirmId	IFS_IDS_LEN	ids	Firm identifier
OrderDate	IFS_DATETIME_LEN	datetime	Date and time of the order entry
MarketMaker	IFS_ENUM_LEN	bool	If it is a Market Maker order
Status	IFS_CHAR_LEN	char	Order status handled by K2. See the Note below the table.
TransactionType <sup>27</sup>	IFS_CHAR_LEN	char	Entry, amendment, withdraw or Tick Up/Tick down. See the Note below the table.

---

<sup>27</sup> See Transaction types section

Name	Length	Type	Description
Msg	IFS_MSG_LEN	string	Points to the message in the badreplymsg if that exists
InternalRef <sup>28</sup>	IFS_BROKERREF_LEN	string	Internal reference for ifs users, never sent to mmts

Note:

Possible values in the Status field of the order entry (query) table are as follows:

'A'	The order is accepted by the K2.
'C'	The order is confirmed by the authorised user (order routing manager).
'D'	The order is denied by K2 or the the order routing manager.
'E'	The order is successfully entered to the Trading Engine.
'R'	The order is refused by the Trading Engine.
'U'	Trading Engine's answer is unexpected or hasn't arrived yet.

Possible values in the Transaction type field of the order entry (query) table are as follows:

'E'	A brand new order. +*/
'A'	An attempt to modify an existing order.
'W'	Remove an existing order.
'T'	TickUpTickDown entry.

**Table 35.19. MMTS II – Market Maker Order entry record (query)**

Note: This table can be used to check the client's own Market Maker orders entered into the K2.

Name	Length	Type	Description
orderid	IFS_INT_LEN	int	Order id given by the pgw
BuyOrdNo	IFS_ORDERNO_LEN	string	System-assigned buy order number
SellOrdNo	IFS_ORDERNO_LEN	string	System assigned sell order number
BoardId	IFS_BOARDID_LEN	string	Board identifier
SecId	IFS_SEC_CODE_LEN	string	Security identifier (security code)
OrderType	IFS_ENUM_LEN	enum	Type of order. Only limit order is possible
Duration	IFS_ENUM_LEN	enum	Only session or day order is possible
PurgeOnLogoff	IFS_ENUM_LEN	bool	If true the order will be withdrawn on K2 (PGW) exit
AllowSoftQtyLimit	IFS_ENUM_LEN	bool	Must be set to true
AllowSoftPriceLimit	IFS_ENUM_LEN	bool	Must be set to true

---

<sup>28</sup> This is filled only in case of appropriate license.

Name	Length	Type	Description
IsPrivate	IFS_ENUM_LEN	bool	True for private orders sent to the POB. Only MMTS_FALSE is allowed.
ExpTime	IFS_ENUM_LEN	datetime	Set to 0, because only session or day expiry is possible
BuySell	IFS_ENUM_LEN	enum	Buy, Sell or Both Sides (0,1,2)
Replace	IFS_ENUM_LEN	bool	If true the previous Market Maker order will be withdrawn and replaced. Replace specified by the MMTS settings will prevail against this field.
ExecutionId	IFS_USER_MIFIDID_LEN	string	The ID of the executor (user or algorithm)
BuyTrdAccId	IFS_IDS_LEN	ids	Buy side Trading account identifier
BuyPrice	IFS_FIXREAL_LEN	fixreal	Price of the Buy side order
BuyYield	IFS_FIXREAL_LEN	fixreal	Yield of the Buy side order
BuyQuantity	IFS_INT_LEN	int	Quantity of Buy side the order
BuyVisibleQuantity	IFS_INT_LEN	int	Not used (left empty)
BuyBrokerRef	IFS_BROKERREF_LEN	string	Free-format field that may be entered by the user for the Buy side order
BuyPositionType	IFS_ENUM_LEN	enum	Position type (open, close, daytrade) of the Buy side order (0,1,2)
BuyTradeRef	IFS_NAME_LEN	string	Free format field that can be used for trade reference in case of close or daytrade
SellTrdAccId	IFS_IDS_LEN	ids	BuyPositionType Sell side Trading account identifier
SellPrice	IFS_FIXREAL_LEN	fixreal	Price of the Sell side order
SellYield	IFS_FIXREAL_LEN	fixreal	Yield of the Sell side order
SellQuantity	IFS_INT_LEN	int	Quantity of Sell side the order

Name	Length	Type	Description
SellVisibleQuantity	IFS_INT_LEN	int	Not used (left empty)
SellBrokerRef	IFS_BROKERREF_LEN	string	Free-format field that may be entered by the user for the Sell side order
SellPositionType	IFS_ENUM_LEN	enum	Position type (open, close, daytrade) of the Sell side order (0,1,2)
SellTradeRef	IFS_NAME_LEN	string	Free format field that can be used for trade reference in case of close or daytrade
Status	IFS_CHAR_LEN	char	SellPositionType Order status (Accepted, Entered, Refused by MMTS, Denied by K2, etc.). See the note below the table.
Msg	IFS_MSG_LEN	string	Error message in case of R(efused) or D(enied) status
InternalRef	IFS_BROKERREF_LEN	string	Internal reference for ifs users, never sent to mmts

Note:

Possible values in the Status field of the Market Maker Order entry (query) table are as follows:

'A'	The order is accepted by the K2.
'C'	The order is confirmed by the authorised user (order routing manager).
'D'	The order is denied by K2 or the the order routing manager.
'E'	The order is successfully entered to the Trading Engine.
'R'	The order is refused by the Trading Engine.
'U'	Trading Engine's answer is unexpected or hasn't arrived yet.

Table 35.20. MMTS II - Order add record

This structure is used by client when sending the given transaction type to K2 server

Name	Length	Type	Description
TrdAccId	IFS_IDS_LEN	ids	TradeAccount identifier
ExecutionId	IFS_USER_MIFIDID_LEN	string	The ID of the executor (user or algorithm)
BuySell	IFS_ENUM_LEN	enum	BuySell indicator
OrderType	IFS_ENUM_LEN	enum	Type of the order (Limit ,Market)
Duration	IFS_ENUM_LEN	enum	Type of the duration

Name	Length	Type	Description
PurgeOnLogoff	IFS_ENUM_LEN	bool	Purge order when the K2 user is logged off or not
AllowSoftQtyLimit	IFS_ENUM_LEN	bool	Allow soft quantity limit. Must be true.
AllowSoftPriceLimit	IFS_ENUM_LEN	bool	Allow soft price limit. Must be true
PositionType	IFS_ENUM_LEN	enum	Type of the position
IsPrivate	IFS_ENUM_LEN	bool	Is Private order or not. Only MMTS_FALSE is allowed.
BoardId	IFS_BOARDID_LEN	string	Board identifier
SecId	IFS_SEC_CODE_LEN	string	Security identifier
Price	IFS_FIXREAL_LEN	fixreal	Price of the order
Yield	IFS_FIXREAL_LEN	fixreal	Yield of the order
Quantity	IFS_INT_LEN	int	Quantity
VisibleQty	IFS_INT_LEN	int	Visible quantity
BrokerRef	IFS_BROKERREF_LEN	string	free-format field that may be entered by the user for each order
ExpTime	IFS_DATETIME_LEN	datetime	Expiration time
TriggerPrice	IFS_FIXREAL_LEN	fixreal	Triggered price
TradeRef	IFS_NAME_LEN	string	Trade reference
MinFillQty	IFS_INT_LEN	int	minimum filled quantity
InternalRef	IFS_BROKERREF_LEN	string	Internal reference for ifs users, never sent to mmts

Table 35.21. MMTS II - Order cancellation record

This structure is used by client when sending the given transaction type to K2 server

Name	Length	Type	Description
OrdNo	IFS_ORDERNO_LEN	string	MMTS order number
OrdNoSpeedIdx	IFS_INT_LEN	int	MMTS order index number
TrdAccId	IFS_IDS_LEN	ids	Trading account identifier
BuySell	IFS_ENUM_LEN	enum	BuySell indicator
BoardId	IFS_BOARDID_LEN	string	Board identifier
InstrId	IFS_IDS_LEN	ids	Instrument identifier
SecId	IFS_SEC_CODE_LEN	string	Security identifier
Price	IFS_FIXREAL_LEN	fixreal	price
Yield	IFS_FIXREAL_LEN	fixreal	Yield
BrokerRef	IFS_BROKERREF_LEN	string	free-format field that may be entered by the user for each order
OpCode	IFS_ENUM_LEN	enum	And, Or, Not
PopCode	IFS_ENUM_LEN	enum	EQ, NT, GT, GE, LT, LE



Name	Length	Type	Description
MultilegOrdNo	IFS_ORDERNO_LEN	string	Modified order number, this value can be found in the order table MultilegOrdNo fields
MultilegSpeedIdx	IFS_INT_LEN	int	MMTS order index number
UserId	IFS_IDS_LEN	ids	User identifier
FirmId	IFS_IDS_LEN	ids	Firm identifier
OrderDate	IFS_DATETIME_LEN	datetime	MMTS date type
MarketMaker	IFS_ENUM_LEN	bool	Market maker
InternalRef	IFS_BROKERREF_LEN	string	Internal reference for ifs users, never sent to mmts

Table 35.22. MMTS II - Order amendment record

This structure is used by client when sending the given transaction type to K2 server

Name	Length	Type	Description
OrdNo	IFS_ORDERNO_LEN	string	Order number
OrdNoSpeedIdx	IFS_INT_LEN	int	Order Idx in mmts. Speed up order delete and amend
TrdAccId	IFS_IDS_LEN	ids	Trade account identifier
ExecutionId	IFS_USER_MIFIDID_LEN	string	The ID of the executor (user or algorithm)
Duration	IFS_ENUM_LEN	enum	Type of the duration
AllowSoftQtyLimit	IFS_ENUM_LEN	bool	Allow soft quantity limit
AllowSoftPriceLimit	IFS_ENUM_LEN	bool	Allow soft price limit
PositionType	IFS_ENUM_LEN	enum	Type of the position opening trade, closing trade, daytrade
Price	IFS_FIXREAL_LEN	fixreal	Price of the order
Yield	IFS_FIXREAL_LEN	fixreal	Yield of the order
Quantity	IFS_INT_LEN	int	Quantity of the order
VisibleQty	IFS_INT_LEN	int	Visible quantity
BrokerRef	IFS_BROKERREF_LEN	string	free-format field that may be entered by the user for each order
ExpTime	IFS_DATETIME_LEN	datetime	Expiration time
TriggerPrice	IFS_FIXREAL_LEN	fixreal	Triggered price

Name	Length	Type	Description
MinFillQty	IFS_INT_LEN	int	Minimum quantity required to fill the order
TradeRef	IFS_NAME_LEN	string	Reference of the trade use for the allocation
InternalRef	IFS_BROKERREF_LEN	string	Internal reference for ifs users, never sent to mmts

Table 35.23. MMTS II - Tick up/Tick down

This structure is used by client when sending the given transaction type to K2 server

Name	Length	Type	Description
OrdNo	IFS_ORDERNO_LEN	string	System-assigned order number of the order. Within one trading day order numbers are assigned sequentially by the System as each order is validated and placed in the order book. The 12 digit order numbers are prefixed with YYYYMMDD- date in the string. The next trading day the order numbers are started from 1 again.
OrdNoSpeedIdx	IFS_INT_LEN	int	Order Idx in mmts. Speeds up order delete and amend
TrdAccId	IFS_IDS_LEN	ids	Trade account identifier
ExecutionId	IFS_USER_MIFIDID_LEN	string	The ID of the executor (user or algorithm)
BuySell	IFS_ENUM_LEN	enum	Buy, Sell or Both Sides
BoardId	IFS_BOARDID_LEN	string	Board identifier
SecId	IFS_SEC_CODE_LEN	string	Security identifier (security code)

Name	Length	Type	Description
Tick <sup>29</sup>	IFS_INT_LEN	int	Number of price ticks (number of standard price steps; increments or decrements by a given tick size). Tick sizes may vary for each security or board.
UserId	IFS_IDS_LEN	ids	User identifier
FirmId	IFS_IDS_LEN	ids	Firm identifier
BrokerRef	IFS_BROKERREF_LEN	string	Free-format field that may be entered by the user for each order
InternalRef	IFS_BROKERREF_LEN	string	Internal reference for ifs users, never sent to mmts

Table 35.24. MMTS II – Market Maker Order entry record

This structure is used by client when sending the given transaction type to K2 server

Name	Length	Type	Description
BoardId	IFS_BOARDID_LEN	string	Board identifier
SecId	IFS_SEC_CODE_LEN	string	Security identifier (security code)
OrderType	IFS_ENUM_LEN	enum	Type of order. Only limit order is possible
Duration	IFS_ENUM_LEN	enum	Only session or day order is possible
PurgeOnLogoff	IFS_ENUM_LEN	bool	If true the order will be withdrawn on K2 (PGW) exit
AllowSoftQtyLimit	IFS_ENUM_LEN	bool	Must be set to true
AllowSoftPriceLimit	IFS_ENUM_LEN	bool	Must be set to true
IsPrivate	IFS_ENUM_LEN	bool	True for private orders sent to the POB. Only MMTS_FALSE is allowed.
ExpTime	IFS_DATETIME_LEN	datetime	Set to 0, because only session or day expiry is possible
BuySell	IFS_ENUM_LEN	enum	Buy, Sell or Both Sides

---

<sup>29</sup> The number of ticks can be negative, in that case the price will be decreased by the given number of ticks

Name	Length	Type	Description
Replace	IFS_ENUM_LEN	bool	If true the previous Market Maker order will be withdrawn and replaced. Replace specified by the MMTS settings will prevail against this field.
ExecutionId	IFS_USER_MIFIDID_LEN	string	The ID of the executor (user or algorithm)
BuyTrdAccId	IFS_IDS_LEN	ids	Buy side Trading account identifier
BuyPrice	IFS_FIXREAL_LEN	fixreal	Price of the Buy side order
BuyYield	IFS_FIXREAL_LEN	fixreal	Yield of the Buy side order
BuyQuantity	IFS_INT_LEN	int	Quantity of Buy side the order
BuyVisibleQuantity	IFS_INT_LEN	int	Not used (left empty)
BuyBrokerRef	IFS_BROKERREF_LEN	string	Free-format field that may be entered by the user for the Buy side order
BuyPositionType	IFS_ENUM_LEN	enum	Position type (open, close, daytrade) of the Buy side order
BuyTradeRef	IFS_NAME_LEN	string	Free format field that can be used for trade reference in case of close or daytrade
SellTrdAccId	IFS_IDS_LEN	ids	BuyPositionType Sell side Trading account identifier
SellPrice	IFS_FIXREAL_LEN	fixreal	Price of the Sell side order
SellYield	IFS_FIXREAL_LEN	fixreal	Yield of the Sell side order
SellQuantity	IFS_INT_LEN	int	Quantity of Sell side the order
SellVisibleQuantity	IFS_INT_LEN	int	Not used (left empty)
SellBrokerRef	IFS_BROKERREF_LEN	string	Free-format field that may be entered by the user for the Sell side order

Name	Length	Type	Description
SellPositionType	IFS_ENUM_LEN	enum	Position type (open, close, daytrade) of the Sell side order
SellTradeRef	IFS_NAME_LEN	string	Free format field that can be used for trade reference in case of close or daytrade
InternalRef	IFS_BROKERREF_LEN	string	SellPositionType Internal reference for ifs users, never sent to mmts

### **36. Error codes**

---

The ifsc functions return with the following error codes:

**IFS\_OK**

No error, 0.

**IFS\_ERROR**

Internal error.

**IFS\_UNKNOWNMSG**

Unknown message type.

**IFS\_NOUSER**

Unknown user.

**IFS\_NOLOGIN**

User is not logged in.

**IFS\_NOACTIVE**

User is not active.

**IFS\_INVPWD**

Invalid password.

**IFS\_NOQUERYPRIV**

User does not have query privilege.

**IFS\_UNKNOWNTABLE**

Unknown table code.

**IFS\_NOSECBOARD**

Unknown secboardid.

**IFS\_NOOB**

The secboard is not on the order book list.

**IFS\_NOENTRYPRIV**

User does not have order entry privilege.

**IFS\_NOSPACE**

No more free space.

**IFS\_UNKNOWNTRANS**

Unknown transaction type.

**IFS\_NOCONFIRMPRIV**

User does not have confirm privilege.

**IFS\_NOORDERENTRY**

Order entry record not found.

**IFS\_UNKNOWNSTATUS**

Unknown order entry status code.

**IFS\_NOCONFIGPRIV**

User does not have privilege to insert / delete order book.

**IFS\_ALREADYWATCH**

The order book is on the list already.

**IFS\_UNKNOWNSWITCH**

Unknown toggle code.

**IFS\_COMERROR**

Communication error.

**IFS\_MSGERROR**

Message format error.

**IFS\_OENOTCSTRING**

Order entry record is not terminated with zero.

**IFS\_OETOO LONG**

Order entry record is too long.

**IFS\_UNCHANGESTATUS**

State of the order entry record is not allowed to change.

**IFS\_TRACEFILEERR**

Trace file open error.

**IFS\_OEDENIED**

Order entry record is denied.

**IFS\_RECONDIFFSRV**

Reconnected to different server, disconnecting.

**IFS\_NOMBP**

No market by price watch for secboard.

**IFS\_MBPALREADYWATCH**

Market by price already watched.

**IFS\_NOLICENSE**

This functions has not been licensed.

**IFS\_CLIENTLICEXCEED**

The number of clients exceeds the licensed limit.

**IFS\_MSGPROTVERDIFF**

The IFS MSG PROTOCOL VERSION of the ifss differs from the version of the ifsc.

**IFS\_ALREADYCONNECT**

This handle has been already connected.

**IFS\_NOCONNECT**

This handle has not been connected.

**IFS\_NULLHANDLE**

The handle is null pointer!

**IFS\_MMONOTCSTRING**

Market Maker Order Entry record is not terminated with 0.

**IFS\_MMOTOOLONG**

Market Maker Order Entry record is too long.

**IFS\_NOMMORDER**

Market Maker Order Entry record can not be found.

**IFS\_UNCHANGEMMSTATUS**

The status of the Market Maker Order Entry record can not be changed.

### **37. Ifs field coding functions**

---

The ifs\_set\_XXX field coding functions help to set up the order entry record to be delivered for the ifsc. These functions insert data given in XXX format to the given buffer in string format.

```
int ifs_set_int(    char * buffer,
                   int data);

int ifs_set_ids    (char * buffer,
                   char * data);

int ifs_set_double (char * buffer,
                   double data);

int ifs_set_code   (char * buffer,
                   int data);

int ifs_set_char   (char * buffer,
                   char data);

int ifs_set_string (char * buffer,
                   int len_in_buffer,
                   char * data);

int ifs_set_enum   (char * buffer,
                   char data);

struct fix_real
{
    double value;
    int decimals;
}

int ifs_set_fix_real(char * buffer,
                    struct fix_real data);
```

#### **int ifs\_set\_XXX**

The length of the field including the terminating zero. Using this function it is possible to set the position to the next field.

#### **char \* buffer**

It is a pointer to the buffer, where the field will be written.

#### **int len\_int\_buffer**

The length of the field in the buffer including the terminating zero.

#### **XXX data**

The data to be written.

Example:

```
int s, len, orderid;
char *buf;

char trdaccid[15], buysell, om[33];
int price;

.
len += ifs_set_ids(buf, trdaccid);
len += ifs_set_char(buf + len, buysell);
len += ifs_set_string(buf + len, IFS_ORDERMETHOD_LEN, om);
.
```



```
len += ifs_set_int(buf + len, price);
.
if ((s = ifsc_orderentry(IFS_ACTION_ORDER_WITHDRAW,
                        buf,
                        len,
                        &orderid)) != IFS_OK) {
    error_handler(...
}
```

### ***38. Ifs field decoding functions***

---

The ifs get XXX functions help to read the fields delivered by the ifs. These functions convert the value of the field from XXX format to string format.

```
int ifs_get_int (int * target,
                char * buffer);

int ifs_get_ids (char * target,
                int len_in_target,
                char * buffer);

int ifs_get_double(double * target,
                  char * buffer);

int ifs_get_code (int * target,
                  char * buffer);

int ifs_get_char (char * target,
                  char * buffer);

int ifs_get_string(char * target,
                  int len_in_target,
                  char * buffer,
                  int len_in_buffer);

int ifs_get_enum (char * target,
                  char * buffer);

struct fix_real
{
    double value;
    int decimals;
}

int ifs_get_fix_real(struct fix_real * target,
                    char * buffer);

int ifs_is_empty_str(char * buffer,
                    int len);
```

#### **int ifs\_get XXX**

The length of the field including the terminating zero. It makes possible to set the position to the next field.

**XXX \* target**

Pointer to the variable, where the value of the field is to be written to.

**int len\_int\_target**

The length of the target string without the terminating zero.

**char \* buffer**

Pointer to the field.

**int len\_int\_buffer**

The length of the field in the buffer including the terminating zero.

**int ifs\_is\_empty\_str**

Return value is 1, if there is an empty string in the head of the buffer, 0 otherwise. The string in the head of the buffer is: The first len byte or if that contains a null byte, then the bytes to the null byte. The empty string has zero length or that contains only spaces (' ').

**int len**

Len of field.

Example:

```
int s, len, seqno;
char *buf;

int orderid, ordno;
char trdaccid[15], buysell, om[33];
.
if ((s = ifsc_get_next_record(&buf,
                             &len,
                             IFS_T_ORDERENTRY,
                             &seqno)) == IFS_OK) {
    len = ifs_get_int(&orderid, buf);
    len += ifs_get_int(&ordno, buf + len);
    len += ifs_get_ids(trdaccid, 14, buf + len);
    len += ifs_get_char(&buysell, buf + len);
    len += ifs_get_string(om, 32, buf + len,
                          IFS_ORDERMETHOD_LEN);
    .
}
```

### ***39. Ifs field step function***

---

The ifs get next function simplifies the reading of the field changes supplied by the ifsc. It takes the code and the value of the next changed field from the string containing the field changes (field code, value pairs).

```
int ifs_get_next_field(char * buffer,
                      int buf_len,
                      int * pos,
                      char *field_code,
                      char **data,
                      int * data_len);
```

**int ifs\_get\_next\_field**

Error code. It is zero if there are any changed fields left, IFS\_NOMORE if there are not.

**int \* buffer**

Pointer to the string containing the field changes.

**int buf\_len**

Length of the string containing the field changes.

**int \* pos**

Index of the field code, value pair waiting for processing.

**char \* field\_code**

Code of the current field.

**char \*\* data**

The changed value of the current field.

**int \* data\_len**

The length of the value of the current field (byte).

Example:

```
for (i = IFS_OK; i == IFS_OK;) {
    i = ifs_get_next_field(msg, len, &pos, &field_code,
                          &data, &data_len);

    if (i == IFS_OK) {
        switch(field_code) {
            case SA_SB_REMARKS:
                ifs_get_string(remarks, IFS_SEC_REMARK_LEN,
                              data, IFS_SEC_REMARK_LEN);
                break;
            case SA_SB_STATUS:
                ifs_get_char(&status, data);
                .
        }
    }
}
```