RJE systems

MCTradesSBI (Orders)

SBI Japannext Trading System Order Placement/Cancellation

This Document:

MCTradesSBI (Orders).pdf – details how to install, configure and run MCTradesSBI (Orders).

Revision:

01/03/2012 – S.C. – Produced the first version of this manual.
08/08/2012 – Vaasugi – Made required modifications.

09/08/2021 – C Carroll – Manual Update

Similar MCTrades Products:

Similar MCTrades products exist for other exchanges:- ASXTrade, ASX24, Chix Aust, HKex, SGX, TSE, OSE, Chix Japan contact RJE for more details.

1. Overview

MCTradesSBI (Orders) application communicates with the FIX Interface of SBI Japannext Trading System. It is used to place orders as well as to cancel its own orders in the exchange. It also extracts the Orders/Trades of its own from the FIX Interface and provides them via comma-delimited feeds as well as stores them in the Database.

Cancellation of its own orders in the exchange could be made through the application itself or through an external client which connects and routes its cancel requests through this application.

The following diagram depicts the overall functionality and connectivity of the MCTradesSBI (Orders) production system.

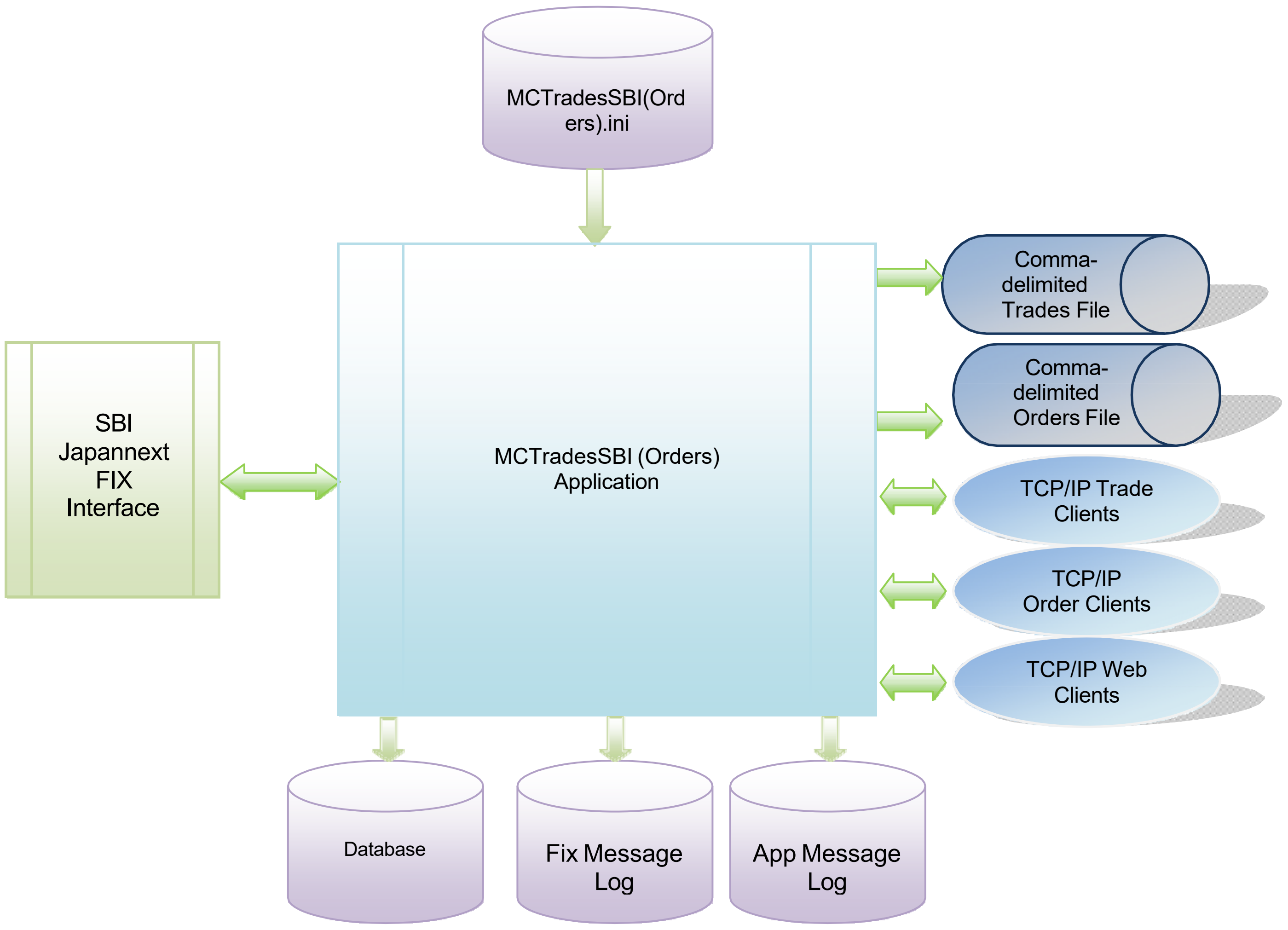


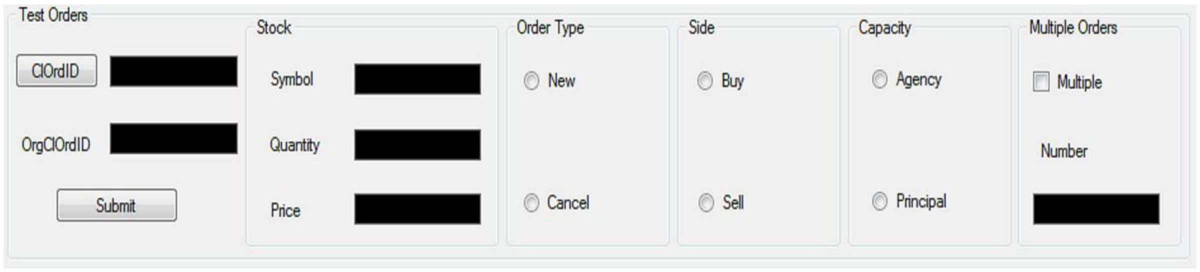
Figure 1. The MCTradesSBI (Orders) Production System

1.1 Features:

Order Placement/Cancellation

Mainly for the purpose of testing, MCTradesSBI (Orders) comes with a Test Orders section, which could be used to place orders or cancel them.

The following shows the Test Orders section of MCTradesSBI (Orders) which enables the user to enter/select the values required for placing or cancelling an order or placing multiple orders at once.



Trade Feed

Trades are available in the following output forms. Trade feed consists of all the Trade Reports extracted from SBI Japannext Trading System.

- Comma-delimited Trade File (single trade side)
- Comma-Delimited TCP/IP Trade Feed (single trade side)

Note: The Comma-Delimited TCP/IP feed is similar to all other MCTrades products.

Order Feed

Orders are available in the following output forms. Order feed consists of all the Execution Reports extracted from SBI Japannext Trading System.

- Comma-delimited order file (single trade side)
- Comma-Delimited TCP/IP order feed (single trade side)

Order Feed comes along with the following.

i) **Data Store**

Orders extracted from SBI Japannext Trading System are stored in the database table 'sbi_orders'. A SQL database is required for this feature. More details can be found in 9. Database

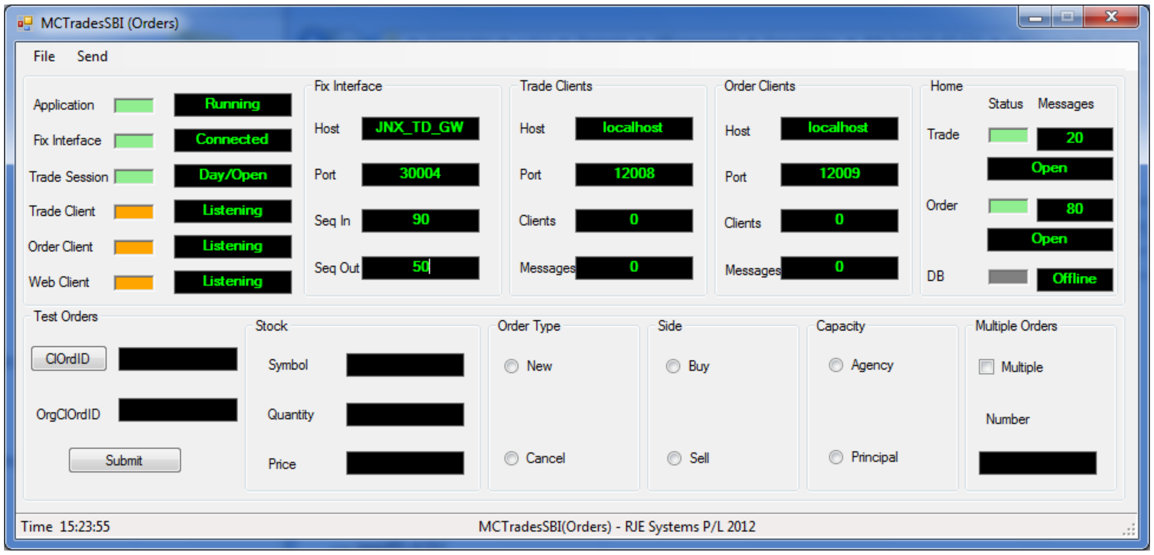
ii) **External Client**

An external GUI client or a PHP based web client can connect and route the cancel requests via MCTradesSBI(Orders). However the application is able to cancel only its own orders.

More details can be found in 8. Command Client

The user can configure the application to enable/disable any of the above features. More details can be found in 4.8 Feature Filter Parameters:

1.2 GUI Screen:



Application Running
Fix Interface Connected
Trade Session Day/Open
Trade Client Listening
Order Client Listening
Web Client Listening

Fix Interface: Host JNX_TD_GW, Port 30004, Seq In 90, Seq Out 50

Trade Clients: Host localhost, Port 12008, Clients 0, Messages 0

Order Clients: Host localhost, Port 12009, Clients 0, Messages 0

Home: Trade 20 Open; Order 80 Open; DB Offline

Test Orders: ClOrdID, OrgClOrdID; Stock: Symbol, Quantity, Price; Order Type: New, Cancel; Side: Buy, Sell; Capacity: Agency, Principal; Multiple Orders: Multiple, Number

Time 15:23:55 MCTradesSBI(Orders) - RJE Systems P/L 2012

The application contains a GUI screen which gives a quick visual indication that everything is working. Typically, good status values are green but status values may transit other states during stopping and starting.

Application Status:-

- Starting (Orange)
- Running (Green) – normal
- Stopping (Red)
- Hibernating (Grey) – normal overnight.
- Waiting (Grey) – normal when user press ‘stop’

FIX Interface Status:-

- Starting (White)
- Recovering (Yellow)
- Connecting (Orange)
- Connected (Green)
- Closing (Grey)

Trade Session Status:-

- Day/Open (Green)
- Day/Closed (Grey)
- Night/Open (Green)

- Night/Closed (Grey)
- Unknown (Grey)

Trade Clients Status:- (if accepting client connections)

- Listening (Orange) - accepting connections
- Connected (Green) – one or more clients connected
- Stopping (Grey)

Order Clients Status:- (if accepting client connections)

- Listening (Orange) - accepting connections
- Connected (Green) – one or more clients connected
- Stopping (Grey)

Web Clients Status:- (if accepting client connections)

- Listening (Orange) - accepting connections
- Connected (Green) – one or more clients connected
- Stopping (Grey)

Trades File Status:-

- Open (Green)
- Closed (Grey)
- Error (Red)

Orders File Status:-

- Open (Green)
- Closed (Grey)
- Error (Red)

Database Status:-

- Open (Green)
- Not connected (Grey)

2. Daily Cycle

SBI Japannext Trading System has multiple Trading Sessions and while MCTradesSBI (Orders) runs for multiple days, it shuts down and wakes up at a certain scheduled time each day, each Trading Session.

Refer:- 4.7 Daily Cycle Parameters:

Note: We currently have no way of detecting Market Close in SBI Japannext FIX service. A timed shutdown is the only option.

Note: SBI Japannext currently has one Day Trading Session and one Night Trading Session. MCTradesSBI(Orders) uses different Session IDs for these, **TD** is the Trading Session ID for Day Session and **TN** for Night Session.

3. Installation

Install MCTradesSBI (Orders) as follows :-

<Install Directory> :- MCTradesSBI.exe, MCTradesSBI(Orders).ini
<Install Directory>:- Mono.Security.dll, Npgsql.dll
<Install Directory>/logs :- Make a subdirectory for logs, trades and orders files.

To run the program, run MCTradesSBI.exe, with the presence of a correctly configured MCTradesSBI(Orders).ini file and the two dll files Mono.Security.dll and Npgsql.dll.

You must set the following parameters correctly:-

- Parameters – Fix Connection Configuration 4.1 FIX Connection Parameters:
- Parameters – Fix Logon Configuration 4.2 FIX Logon Parameters:

If you wish to run the programs without a GUI refer 4.9 Other parameters:

Note: When upgrade to a new version intra-day you should copy the FIX log file if installing in a new directory.

4 Configuration

All configuration parameters are stored in MCTradesSBI(Orders).ini

4.1 FIX Connection Parameters:

FIX connection parameters should be configured for each Trading Session. Parameter names come with the Trading Session ID.

FIX_TD_SERVER_HOST = Name of SBI Japannext Fix Server for Day Trading Session
e.g **FIX_TD_SERVER_HOST**=JNX_TD_GW

FIX_TN_SERVER_HOST = Name of SBI Japannext Fix Server for Night Trading Session
e.g **FIX_TN_SERVER_HOST**=JNX_TN_GW

FIX_TD_SERVER_PORT = Port to connect SBI Japannext Fix Day Trading Session
e.g **FIX_TD_SERVER_PORT**=30004

FIX_TN_SERVER_PORT = Port to connect SBI Japannext Fix Night Trading Session
e.g **FIX_TN_SERVER_PORT**=25017

SBI Japannext will supply values for these parameter settings.

Note: IP address of the FIX Server Hosts can be entered in the ini file. Alternatively an entry for each Fix Server Host can be made in the windows host file (C:\Windows\System32\drivers\etc\hosts) with a name for example JNX_TD_GW and that name in turn can be used in the ini file.

4.2 FIX Logon Parameters:

FIX logon parameters should be configured for each Trading Session. Parameter names come with the Trading Session ID.

FIX_TD_SENDER_ID = Part of Fix header for Day Trading, a valid value must be specified.
e.g **FIX_TD_SENDER_ID**=TD12479A

FIX_TN_SENDER_ID = Part of Fix header for Night Trading, a valid value must be specified.

e.g **FIX_TN_SENDER_ID**=TN12479A

FIX_TD_TARGET_ID = Part of Fix header for Day Trading, a valid value must be specified.
e.g **FIX_TD_TARGET_ID**=TDJNX

FIX_TN_TARGET_ID = Part of Fix header for Night Trading, a valid value must be specified.
e.g **FIX_TN_TARGET_ID**=TNJNX

SBI Japannext will supply values for these parameter settings.

4.3 Optional FIX Parameters:

FIX_HEARTBEAT=<Heartbeat interval> (Seconds) – default = 30
Example **FIX_HEARTBEAT**=30

Note: You should consult SBI Japannext before setting this parameter, the default of 30 seconds is recommended.

4.4 Trade Feed Parameters:

These are TCP/IP ports that applications can connect to receive a feed of trade data.

The format of the data is described in 5. Comma Delimited Application Development

TRADES_PORT = TCP/IP port for trades.
e.g **TRADES_PORT**=12008

4.5 Order Feed Parameters:

This is the TCP/IP port that applications can connect to receive a feed of execution report data.

The format of the data is described in 5. Comma Delimited Application Development

ORDERS_PORT = TCP/IP port for all Orders.
e.g. **ORDERS_PORT**=12009

4.6 Logging Parameters:

Application and FIX logs are text files that can be used for trouble shooting. The logs' names contain the Trading Session ID which the logs belong to, making it easy for the user to differentiate logs of one session from others.

APP_LOG_FILE = file base for application log, a new log is taken each run; the application log includes Trading Session ID, current date and time.
e.g **APP_LOG_FILE**= MCTradesSBI
The name of the file e.g MCTradesSBI.TD.App.Messages.20120808_144159.log

FIX_LOG_FILE = file base for FIX Message Log; a new log is taken each trading session; the filename always includes the Trading Session ID and the current date.
e.g **FIX_LOG_FILE**= MCTradesSBI
The name of the file e.g MCTradesSBI.TD.Fix.Messages.20120808

APP_LOG_DIRECTORY=Directory where the application log is stored.
e.g **APP_LOG_DIRECTORY**=logs

FIX_LOG_DIRECTORY=Directory where FIX Message Log is stored.
e.g **FIX_LOG_DIRECTORY**=logs

APP_DATA_DIRECTORY=Directory where the Trades and Orders files are stored.
e.g **APP_DATA_DIRECTORY**=data

Note: APP_DATA_DIRECTORY defaults to APP_LOG_DIRECTORY if not specified.

If you don't specify these settings, defaults will apply.

Note: In this application the FIX Message Log is important see 6.1 FIX Message Log: for more details.

4.7 Daily Cycle Parameters:

Refer:- 2. Daily Cycle

Wake up and Shut down times should be configured for each Trading Session. Parameter names come with the Trading Session ID.

TD_WAKE_TIME = time when program wakes up each morning for day trading session (hour:min), default 08:20.
e.g **TD_WAKE_TIME**=08:20

TD_SHUT_TIME = time when the program shutdown after day trading session and hibernation occurs (hour:min) default 16:30.
e.g **TD_SHUT_TIME**=16:30

TN_WAKE_TIME = time when program wakes up every night for night trading session (hour:min), default 19:00.
e.g **TN_WAKE_TIME**=19:00

TN_SHUT_TIME = time when the program shutdown after night trading session and hibernation occurs (hour:min) default 23:59.
e.g **TN_SHUT_TIME**=23:59

4.8 Feature Filter Parameters:

Refer:- 1.1 Features:

TRADE_FEED=TRUE – set TRUE to enable Trade Feed

ORDER_FEED=TRUE – set TRUE to enable Order Feed

Note: Enabling the Order Feed also enables the database connection as well as the external client connection for order cancellation.

4.9 Other Parameters:

NO_GUI=NO – this should always be set to NO; The GUI is required for order entry/cancellation

NO_TEST_ORDERS=YES – set to enable order entry/cancellation through the application

See also

- 8.3 Command Clients Parameters:
- 9.2 SQL Database Parameters:

4.10 Configuration File Example :

```
*******************************
* APPLICATION VERSION         *
*******************************
APP_VERSION=MCTradesSBI(Orders) - RJE Systems P/L 2012
*******************************
* RUN WITHOUT GUI             *
*******************************
*NO_GUI=YES
*******************************
* RUN WITHOUT TEST ORDERS     *
*******************************
*NO_TEST_ORDERS=YES
*******************************
* FIX Session properties      *
*******************************
FIX_TD_SERVER_HOST=JNX_TD_GW
FIX_TN_SERVER_HOST=JNX_TN_GW
FIX_TD_SERVER_PORT=30004
FIX_TN_SERVER_PORT=25017
FIX_TD_SENDER_ID=TD12479A
FIX_TN_SENDER_ID=TN12479A
FIX_TD_TARGET_ID=TDJNX
FIX_TN_TARGET_ID=TNJNX
FIX_CHANGE_PASS=NO
FIX_HEARTBEAT=30
*******************************
* TCP Clients properties      *
*******************************
TRADES_PORT=12008
ORDERS_PORT=12009
COMMAND_PORT=12010
ORDER_REFRESH=DELETE
***********************************
* Application Log File properties *
***********************************
APP_LOG_FILE=MCTradesSBI
APP_LOG_DIRECTORY=logs
LOGGING_LEVEL=9
*******************************
*  Fix Log File properties    *
*******************************
FIX_LOG_FILE=MCTradesSBI
FIX_LOG_DIRECTORY=logs
***********************************
* BROKER_LIST To filter own trades*
***********************************
BROKER_LIST=ABN01
***********************************
* FEATURE SELECTOR                *
```

```
***********************************
TRADE_FEED=TRUE
ORDER_FEED=TRUE
***********************************
* WAKE UP/SHUT DOWN               *
***********************************
TD_WAKETIME =08:20
TD_SHUTTIME =16:30
TN_WAKETIME =19:00
TN_SHUTTIME =23:59
***********************************
* SQL Database Parameters         *
***********************************
SQL_DATABASE_NAME=webdb
SQL_DATABASE_SERVER=127.0.0.1
SQL_DATABASE_PORT=5432
SQL_USER_ID=postgres
SQL_PASSWORD=rjeadmin
*JUNK=BAD
************ END **************
```

5 Comma-Delimited Application Development

One option for developers is to make a TCP/IP separate connections to MCTradesSBI trade/order feed ports and receive trades/orders data in comma-delimited format separately. Data is simply sent when it is available; there is no need to request data. In this case trades are single sided, all data received from SBI Japannext is included.

The port for clients connections is configured in :- 4.4 Trade Feed Parameters:

5.1 Comma-Delimited Header:

Most applications would process the header as it gives a list of field names corresponding to field positions.

Trades

Country|S,Exchange|S,Market|S,TradeDate|D,FirmID|S,TraderID|S,TradeNo|N,OrderID|S,ClOrderID|S,ExecID|N,ExecTransType|N,OrdStatus|N,Account|S,Quantity|N,SecCode|S,Price|N,Value|N,TradeTime|T,UTCTimeStamp|TS,Side|C,OrderQty|N,AvgPrice|N,CumQty|N,TransactTime|TS,ExecType|C,LeavesQty|N,~

Orders

Country|S,Exchange|S,Market|S,TradeDate|D,FirmID|S,TraderID|S,MsgSeqNo|N,OrderID|N,ClOrderID|N,ExecID|N,ExecTransType|N,OrdStatus|N,OrderRejectReason|S,Account|S,Symbol|S,TradeTime|T,UTCTimeStamp|TS,Side|S,OrderQty|N,Price|N,LastShares|N,CumQty|N,TransactTime|T,OrderType|N,~

5.2 Comma-Delimited Data:

Fields that are not relevant are simply empty:-

Trades

JAPAN,SBI,SBI,20120808,TDJNX,TDC12479A,90,20120808-57c8,1520000,00115022052B-22,0,2,ABN01,400,7203,3200,1280000.00,15:20:27,20120808-06:20:27,1,400,3200,400,20120808-06:20:27,2,0,~

Orders

JAPAN,SBI,SBI,20120808,TDJNX,TDC12479A,69,20120808-57c8,1520000,001150220519-1,0,0,,ABN01,7203,15:20:09,20120808-06:20:09,1,400,,,0,20120808-06:20:09,2,~

Note: Additional examples are available from RJE.

5.3 Trades File:

A Trade file is produced for each trading session with a comma-delimited header and a comma-delimited trade record for each trade. The contents of this file are identical to the data that would be sent for a trades feed.

On a restart mid-session, the internal copy of the trades is recreated from the FIX Message Log and the old trades file gets replaced by a new trades file.

The trades file's name contains the Trading Session ID which the file belongs to, making it easy for the user to differentiate it from the others.

e.g MCTradesSBI.TD.20120808.trades

5.4 Orders File:

An Order file is produced for each trading session with a comma-delimited header and a comma-delimited execution record for each execution. The contents of this file are identical to the data that would be sent of an orders feed.

On a restart mid-session, the internal copy of the orders is recreated from the FIX Message Log and the old orders file gets replaced by a new orders file.

The orders file's name contains the Trading Session ID which the file belongs to, making it easy for the user to differentiate it from the others.

e.g MCTradesSBI.TD.20120808.orders

6 Message Sequence Numbers

Message Sequence Numbers start from 1 each trading session. By default when reconnecting/restarting mid-session, the sequence numbers at both ends continue on from their previous values and any missing messages are recovered. Hence, on a restart the application reprocesses the FIX Message log to re-establish outbound/inbound sequence numbers.

6.1 FIX Message Log:

Typically the FIX session is continued across runs and there is a single FIX Message log for each trading session. Messages sent/received are recovered from the FIX Messages log at startup. When resuming the FIX session the application only fetches the new messages.

You can specify a filename/directory for this file in 4.6 Logging Parameters:

6.2 Missing FIX Message Log:

A missing FIX message log could be cause by the following things:-

- Running from a different directory or with different .ini settings.
- Deleting or renaming the file.

This can cause problems with the sequence number of the login message we send to the SBI Japannext Trading System. If the sequence number is less than expected the exchange will ignore this message and you will eventually get the following error:-

```
|*************************************************************************************
*** Fatal Error - Exceeded FIX Logon retries - Check Config FIX_SERVER_PORT / FIX_SERVER_HOST.  ***
*** If settings above are corrent then could be a problem with the Fix Message log.              ***
*** See MCTradesSBI.PDF - 6.2 Missing Fix Message Log.                                           ***
*** You can run MCTradesSBI -s 'nnn'.   Where 'nnn' = last message sequence no from our end.     ***
|*************************************************************************************
```

This error could mean the FIX Message log has been deleted or you could simply be connecting to the wrong host/port.

Note: Resetting the message sequence number will solve this error. The version of FIX protocol being used (version 4.2 with some features of 4.4 back ported) supports resetting sequence numbers. For more info see 7.2 User set Message Sequence Numbers:

Note: A message log error can only be a problem if you have successfully connected earlier.

6.3 Specifying a Restart Sequence No:

If you know what the outbound FIX sequence number from your end should be, you can specify it as follows.

```
MCTradesSBI –S nnn
```

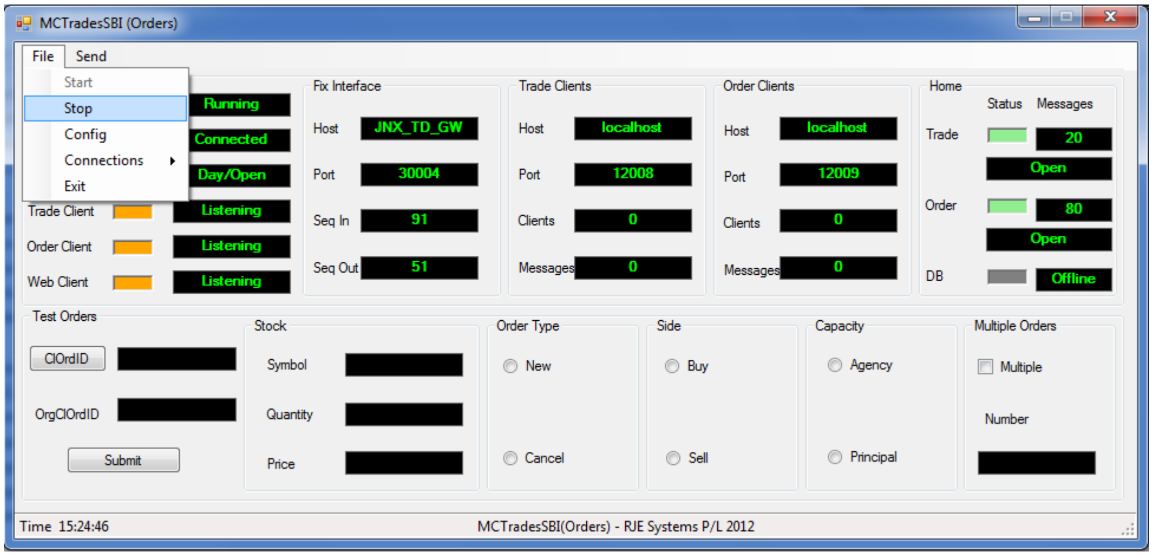
Note: Where nnn is the sequence number.

You should be able to get the number from the previous FIX Message Log. If you don’t know this number you can obtain it from the SBI Japannext administrator or he can reset the FIX session (as the last option). In this mode the application will re-request all trades for the trading session from SBI Japannext.

7 Features for Testing

The following features are included solely for the purpose of facilitating the testing of this application.

7.1 Stop and Start:



Running
Connected
Day/Open
Trade Client Listening
Order Client Listening
Web Client Listening

Fix Interface
Host JNX_TD_GW
Port 30004
Seq In 91
Seq Out 51

Trade Clients
Host localhost
Port 12008
Clients 0
Messages 0

Order Clients
Host localhost
Port 12009
Clients 0
Messages 0

Home
	Status	Messages
Trade	🟩	20
	Open	
Order	🟩	80
	Open	
DB	▨	Offline

Test Orders
ClOrdID
OrgClOrdID

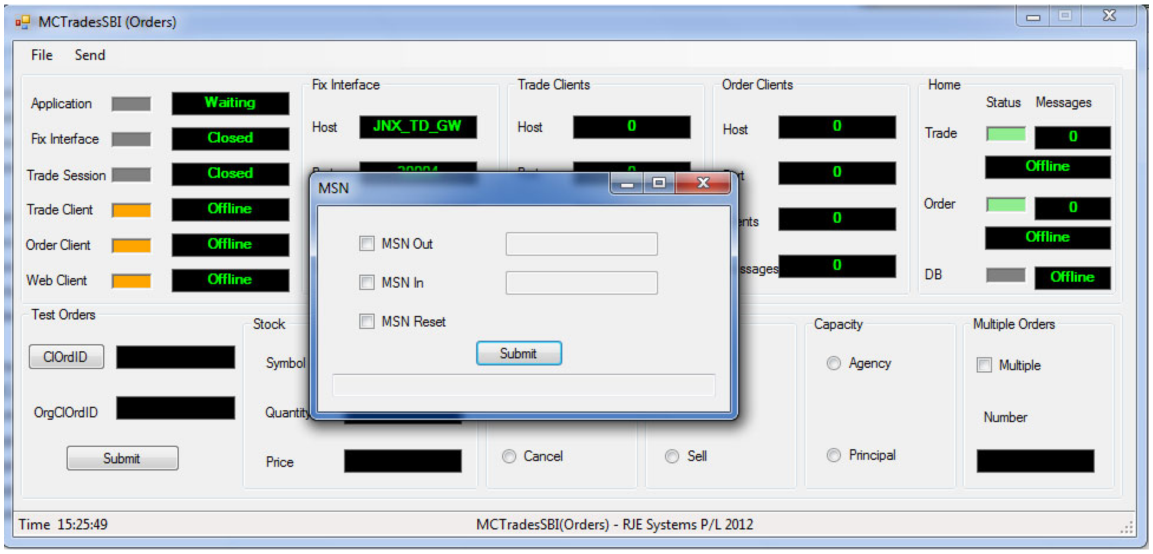
Stock
Symbol
Quantity
Price

Order Type
Side
Capacity
Multiple Orders
Number

Time 15:24:46
MCTradesSBI(Orders) - RJE Systems P/L 2012

The ‘Stop’ and the ‘Start’ submenus provided under the File menu facilitate stopping and restarting the application without exiting from the application.

7.2 User set Message Sequence Numbers:



When restarting the application via Stop/Start menu, the user is given the option of changing the last outbound (MSN Out) and inbound (MSN In) message sequence numbers in MCTradesSBI(Orders) application. He also can reset the message sequence number if he wants. This feature could be used during the conformance tests to see how both parties react (application and exchange) for such changes in the message sequence number as well as to resolve the error when the FIX log file is missing.

See 6.2 Missing FIX Message Log:

8 Command Client:

8.1 Order Cancellation:

An external GUI client or a PHP based web client can connect and route the cancel requests via MCTradesSBI(Orders). However the application is able to cancel only its own orders.

The following FIX messages are related to order cancellation

- Order Cancel Request
- Order Cancel Acknowledgement
- Order Cancel Reject

When an order cancel request succeeds an Order Cancel Acknowledgement message, which is one kind of Execution Report message, is sent by the exchange. An Order Cancel Reject is only sent when an order cancel request fails. MCTradesSBI(Orders) stores the cancel results in 'sbi_order_cancel_result' table in the database.

8.2 Supported Cancellation Types:

Only one type of order cancel is currently supported:-

1. Cancel Individual Order:-
   CANCEL_REQUEST|USER=admin|REQUEST_NO=297|CxlType=F|OrderID=6688077|~

8.3 Command Clients Parameters:

COMMAND_PORT= TCP port command clients connect to.
COMMAND_PORT=12010

8.4 Cancellation Database Updates:

Table :- **sbi_last_clord_id** this table is used to ensure a unique ClOrdId for each cancellation transaction. It is updated after each can cancellation request to ensure each request has a unique id.

Table :- **sbi_trans** web clients create an entry in this table each time they issue a cancellation request. MCTradesSBI updates this table when the cancelation request result is known. It is intended that web clients will archive the contents of this table.

Table :- **sbi_order_cancel_result** this table is updated with the results of each order cancel request. The intention is that this table will be a long term 'audit trail' of cancellation activity.

9 Database

Database design, tables and functions have been developed and tested with a ProsgreSQL database running under Windows and Linux.

The MCTradesSBI(Orders) application uses the "npgsql" .net data provider for PostgreSQL. It calls ProgreSQL Functions (Stored Procedures) for database access and update.

9.1 Database Tables:

Table - system

The purpose of this table is to allocate a unique Guid (uuid) to each system. In this context MCTradesSBI(Orders) is one system. All data of MCTradesSBI(Orders) has the system_id of MCTradesSBI.

Function :- get_system_info() create/retrieve system table information for a particular system.

```
-- Table: "system"
-- DROP TABLE "system";

CREATE TABLE "system"
(
  id bigserial NOT NULL,
  system_id uuid,
  system_name character varying(50),
  exchange character varying(10)
)
WITH (
  OIDS=TRUE
);
ALTER TABLE "system" OWNER TO postgres;
GRANT ALL ON TABLE "system" TO postgres;
GRANT ALL ON TABLE "system" TO public;
```

Table – system_state

This table shows the current state of a system indicating if the system is ready to store the data.

Currently defined system states are:-

```
enum SessionState : int
{
    Connecting = 10,
    Connected  = 20,
    Ready      = 30,
    Closed     = 90
}
```

The table is also updated periodically to provide `memory_trans` and `database_trans` counters. These provide feedback of whether orders table updating is keeping with the rate execution reports are being sent by the SBI Japannext Trading System.

Function :- update_system_state() - updates the system_state table.

```
-- Table: system_state

-- DROP TABLE system_state;

CREATE TABLE system_state
(
  id bigserial NOT NULL,
  system_id uuid,
  system_state integer,
  host_name character varying(10),
  port_no integer,
  memory_trans integer,
  database_trans integer,
  last_update timestamp without time zone
)
WITH (
  OIDS=TRUE
);
ALTER TABLE system_state OWNER TO postgres;
GRANT ALL ON TABLE system_state TO postgres;
GRANT ALL ON TABLE system_state TO public;
```

Table – sbi_orders

This is the main table of interest which stores orders' data. As execution reports occur the current state of the database is updated to reflect the current state of the order. When the field order_active='Y' the order is an active order which is a candidate for cancellation. As orders trade out or are cancelled order_active is set to 'N'.

The orders information is kept in the DB indefinitely as it may be useful.

Function :- sbi_update_order() – Updates the orders table for each execution report transaction.

```
-- Table: sbi_orders

-- DROP TABLE sbi_orders;

CREATE TABLE sbi_orders
(
  id bigserial NOT NULL,
  system_id uuid NOT NULL,
  order_id character varying(50) NOT NULL,
  message_no integer,
  order_active character(1),
  order_status character(1),
  order_type character varying(4),
  order_instructions character varying(4),
  order_ref character varying(50),
  order_modified_time timestamp without time zone,
  firm_id character varying(30),
  trader_id character varying(30),
  account character varying(30),
  symbol character varying(50),
  side character(1),
  price numeric(18,4),
  order_qty numeric,
  qty_filled numeric,
  no_fills smallint,
  last_fill numeric,
  last_filltime timestamp without time zone,
  fill_transact_id character varying(50),
  transact_type character varying(4),
  client_name character varying(30),
  customer_ref character varying(50),
  order_expire_time character varying(20),
  commodity character varying(10),
  month_year character varying(10),
  mm integer,
  yyyy integer,
  exchange character varying(10),
  currency character varying(10),
  isin character varying(50),
  product_group character varying(10),
  strike_price numeric,
  expiration_date date,
  last_update timestamp without time zone,
  CONSTRAINT sbi_order_pkey PRIMARY KEY (system_id, order_id)
)
WITH (
  OIDS=FALSE
);
ALTER TABLE sbi_orders OWNER TO postgres;
GRANT ALL ON TABLE sbi_orders TO postgres;
```

Table – sbi_last_clord_id

This table is used to ensure a unique ClOrdId for each cancellation transaction. It is updated after each cancellation request to ensure each request has a unique id.

Fucntions:- sbi_get_cl_ord() and sbi_update_cl_ord()

```
-- Table: sbi_last_clord_id

-- DROP TABLE sbi_last_clord_id;

CREATE TABLE sbi_last_clord_id
(
  id bigserial NOT NULL,
  system_id uuid NOT NULL,
  last_clord_id integer,
  CONSTRAINT sbi_last_clord_id_pkey PRIMARY KEY (system_id)
)
WITH (
  OIDS=TRUE
);
ALTER TABLE sbi_last_clord_id OWNER TO postgres;
GRANT ALL ON TABLE sbi_last_clord_id TO postgres;
GRANT SELECT(system_id), UPDATE(system_id), INSERT(system_id),
REFERENCES(system_id) ON sbi_last_clord_id TO public;
```

Table – sbi_order_cancel_result

This table is updated with the results of each order cancel request. The intention is that this table will be a long term 'audit trail' of cancellation activity.

Function :- sbi_update_order_cancel() - updates this table and the sbi_trans table with the results of each order cancel request.

```
-- Table: sbi_order_cancel_result

-- DROP TABLE sbi_order_cancel_result;

CREATE TABLE sbi_order_cancel_result
(
  id bigserial NOT NULL,
  system_id uuid,
  order_ref character varying(50),
  user_name character varying(30),
  request_id character varying(40),
  no_orders_cancelled integer,
  error_text character varying(256),
  error_id character varying(10),
  cancel_type character varying(10),
  account character varying(30),
  symbol character varying(50),
  side character(1),
  price numeric,
  order_id character varying(50),
  client_name character varying(30),
  customer_ref character varying(50),
  cancel_time timestamp without time zone
)
WITH (
  OIDS=TRUE
);
ALTER TABLE sbi_order_cancel_result OWNER TO postgres;
GRANT ALL ON TABLE sbi_order_cancel_result TO postgres;
```

Table – sbi_trans

Web clients create an entry in this table each time they issue a cancellation request. MCTradesSBI updates this table when the cancelation request result is known. It is intended that web clients will archive the contents of this table.

```
-- Table: sbi_trans

-- DROP TABLE sbi_trans;

CREATE TABLE sbi_trans
(
  id bigserial NOT NULL,
  uid character(128),
  account character varying(30),
  order_id character varying(50),
  status character(1),
  ip character varying(256),
  "timestamp" time without time zone DEFAULT now()
)
WITH (
  OIDS=FALSE
);
ALTER TABLE sbi_trans OWNER TO postgres;
GRANT ALL ON TABLE sbi_trans TO postgres;

-- Index: "sbiIndOrderID"

-- DROP INDEX "sbiIndOrderID";

CREATE INDEX "sbiIndOrderID"
  ON sbi_trans
  USING btree
  (order_id);
```

9.2 Database Parameters:

SQL_DATABASE_NAME=Name of the database to access.
e.g SQL_DATABASE_NAME=webdb

SQL_DATABASE_SERVER=The machine which is the PostgreSQL database server.
e.g SQL_DATABASE_SERVER=rjelinuxlap

SQL_DATABASE_PORT=Port for the PostgreSQL database.

e.g SQL_DATABASE_PORT=5432

SQL_USER_ID=PostgreSQL database user.
e.g SQL_USER_ID=postgres

SQL_PASSWORD= PostgreSQL database user password*
e.g SQL_PASSWORD=rjexxxxxx

9.3 npgsql files:

The following files should reside in the same directory as MCTradesSBI.exe:-

```
	Mono.Security.dll
	Npgsql.dll
```

These files are the "npgsql" .net data provider for PostgreSQL.

9.4 SQL Script Files:

The following files create database tables:-

create_sbi_last_clord_id.sql
create_sbi_order_cancel_result.sql
create_sbi_orders.sql
create_system.sql
create_system_state.sql
create_sbi_trans.sql

The following files create database functions:-

func_sbi_get_cl_ord.sql
func_sbi_update_cl_ord.sql
func_sbi_update_orders.sql
func_sbi_update_order_cancel.sql
func_get_system_info.sql
func_update_system_state.sql