MCTradesTSE

Tokyo Stock Exchange Trades/Notices Feed and Order Cancellation

This Document:

MCTradesTSE.pdf – details how to install, configure and run MCTradesTSE.

Revision:

30/07/2012 – vaasuGI – Produced the first version of this manual.

06/09/2021 – C Carroll – Manual Update

Similar MCTrades Products:

Similar MCTrades products exist for other exchanges:- ASX, SFE, HKEX, SGX, TSE
Contact RJE for more details.

1. Running MCTradesTSE:

MCTradesTSE is designed to communicate with different Arrowhead servers/interfaces of Tokyo Stock Exchange. It connects to the DropCopy server to extract notices from the exchange and produce comma-delimited feeds for notices and trades. It connects to OrderNotice server to send the order cancel requests made via web client.

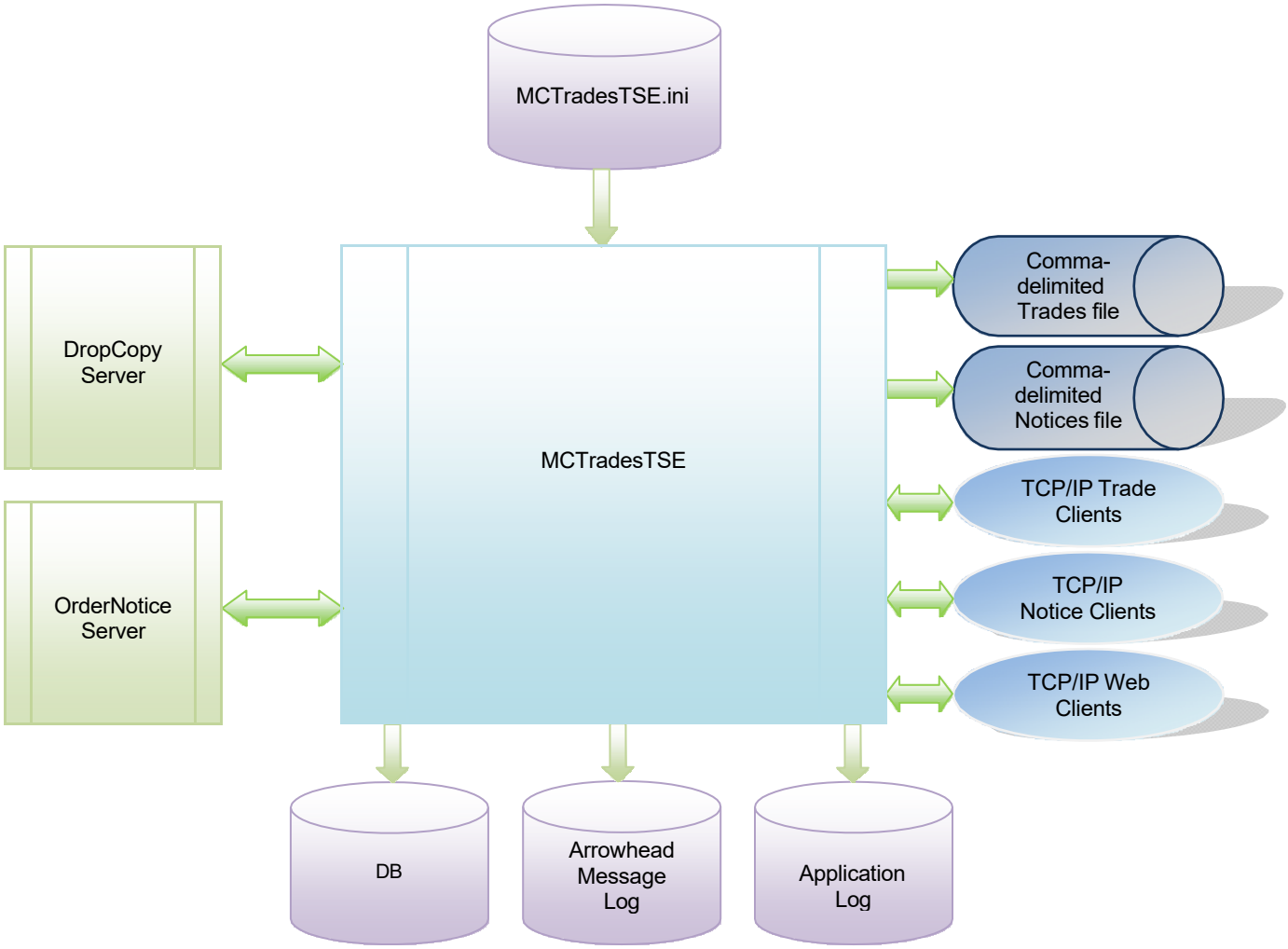


Figure 1. The MCTradesTSE Production System

1.1 Features:

Trade Capturing

This extracts the Trade Reports from the FIX interface and makes it available in the following output forms.

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

Notice Capturing

This extracts the Execution Reports from the FIX interface and makes it available in the following output forms. Execution Reports include orders, fills/trades and order cancels acknowledgements. This feature is combined with the next feature 'Order Cancellation'

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

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

There is no support for MC API client connections.

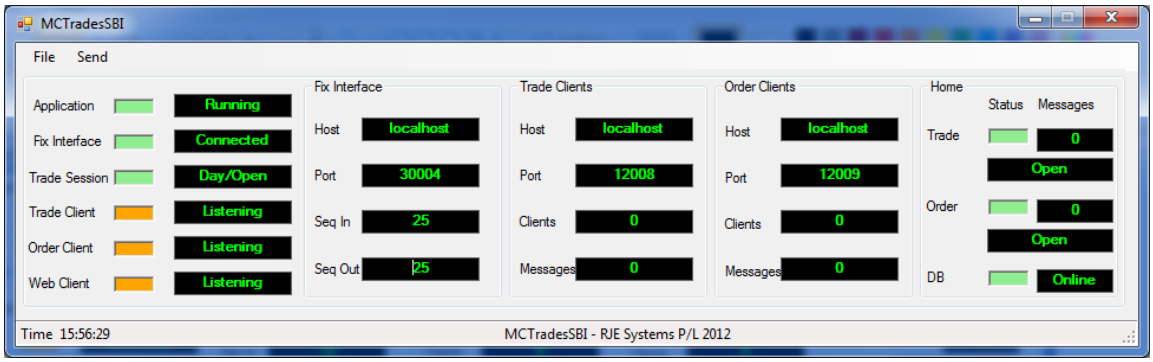
Order Cancellation

A PHP based web client application can be used for viewing and cancelling orders. More details can be found in:- 7. Command Clients Order Cancelation:

An SQL database is required for order cancellation due to the need to ensure a unique ClOrdId for each cancellation transaction. More details can be found in :- 8. SQL Database:

The user can configure the application to have Trade Capture only or Order Capture with Order Cancellation only or both. More details can be found in 4.8 Trade/Order Capture Parameters:

1.2 GUI Screen:



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 my transit other states during stopping and starting.

Application Status:-

- Starting (Orange)
- Running (Green) – normal
- Stopping (Red)
- Waiting (Grey) – between 'Stop' and 'Start'
- Hibernating (Grey) – normal overnight.

FIX Interface Status:-

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

2. Daily Cycle:

MCTradesTSE can be run for multiple days; it shuts down and wakes up at a certain scheduled time each day.

Refer:- 4.7 Daily Cycle Parameters:

WAKE_TIME = time when program wakes up each morning,
SHUT_TIME = time when the program shutdown (hibernation) occurs.

This area functions as per existing MCTrades products.

Note: We currently have no way of detecting Market Closed in a FIX based feed. A timed shutdown is the only option.

3. Installation:

Install MCTradesTSE as follows :-

<Install Directory> :- MCTradesTSE.exe, MCTradesTSE.ini
<Install Directory>/logs :- make a subdirectory for log files.

To run the program, run MCTradesTSE.exe, provided the configuration in the .ini file is correct no other information is needed.

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 program 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 MCTradesTSE.ini

4.1 FIX Connection Parameters:

FIX_SERVER_HOST = Name of TSE Server
Example **FIX_SERVER_HOST**=TSE_JAPAN_GW

FIX_SERVER_PORT = Port to connect to for FIX.
FIX_SERVER_PORT=15007

TSE will supply values for these parameter settings.

4.2 FIX Logon Parameters:

FIX_SENDER_ID = Part of Fix header, a valid value must be specified. TSE will supply. Example **FIX_SENDER_ID**=RTRRJE01

FIX_TARGET_ID = Part of Fix header, a valid value must be specified. TSE will supply. Example **FIX_TARGET_ID**=TDJNX

FIX_USER_ID = user for Fix Logon. Example **FIX_USER_ID**=RJERJE01

FIX_PASSWORD = Password for Fix Logon. Example **FIX_PASSWORD**=Password

Note: We are not aware of any function of automatic password changing in TSE trade feeds.

Note: Since TSE has multiple trading sessions, logons to different sessions need require different sender IDs assigned by TSE.

FIX_TRDSES_1_SENDER_ID=TD12479A
FIX_TRDSES_2_SENDER_ID=TD12479B

4.3 Optional FIX Parameters:

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

Note: You should consult TSE 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 all trades.
Example **TRADES_PORT**=12008

4.5 Order Feed Parameters:

This is the TCP/IP port that applications can connect to receive a feed of execution reports 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:

The application log and FIX log are text files that can be used for trouble shooting.

APP_LOG_FILE = file base for application log, a new log is taken each run; the application log includes the current date and time.

Example **APP_LOG_FILE**= MCTradesTSE
The filename produced will be (MCTradesTSE.App.Messages.20120423_170238.log)

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

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

APP_DATA_DIRECTORY defaults to APP_LOG_DIRECTORY if not specified.

FIX_LOG_FILE = file base for FIX Message Log, the filename always includes the current date. Example **FIX_LOG_FILE**=MCTradesTSE,
The filename produced will be (MCTradesTSE.Fix.Messages.20120423.log)

FIX_LOG_DIRECTORY=Directory where FIX Message Log is stored.
FIX_LOG_DIRECTORY=logs

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_TIME = time when program wakes up each morning (hour:min), default 07:00.
Example **WAKE_TIME**=07:30

SHUT_TIME = time when the program shutdown (hibernation) occurs (hour:min) default 23:30.
Example **SHUT_TIME**=21:00

Note : When MCTradesTSE wakes up for the trading session one, it recovers the trades and orders of the previous night session from the FIX log so as to allow it to contain trades and orders of the current day session as well as the ones of previous night session.

4.8 Trade/Order Capture Parameters:

Refer:- 1.1 Features:

TRADES = TRUE – set to TRUE to capture trade reports
ORDERS = TRUE – set to TRUE to capture execution reports and enable order cancellation via web client

4.9 Other Parameters:

See :-

- 7.3 Command Clients Parameters:
- 8.2 SQL Database Parameters:

4.10 Configuration File Example :

```
******************************
* APPLICATION VERSION         *
******************************
APP_VERSION=MCTradesTSE - RJE Systems P/L 2012
******************************
* RUN WITHOUT GUI             *
******************************
*NO_GUI=YES
******************************
* RUN WITHOUT TEST ORDERS     *
******************************
*NO_TEST_ORDERS=YES
******************************
* FIX Session properties      *
******************************
FIX_SERVER_HOST=localhost
FIX_SERVER_PORT=30004
FIX_TRDSES_1_SENDER_ID =TD12479A
FIX_TRDSES_2_SENDER_ID =TD12479B
FIX_TARGET_ID=TDJNX
FIX_USER_ID=RTRRJE01
```

```
FIX_PASSWORD=[None]
FIX_CHANGE_PASS=NO
FIX_HEARTBEAT=30
*FIX_NEW_SESSION=YES
*
*******************************
* TCP Clients properties      *
*******************************
TRADES_PORT=12008
ORDERS_PORT=12009
COMMAND_PORT=12010
ORDER_REFRESH=DELETE
*
*********************************
* Application Log File properties *
*********************************
*APP_LOG_FILE=MCTradesTSE
APP_LOG_DIRECTORY=logs
LOGGING_LEVEL=9
*
*******************************
*  Fix Log File properties    *
*******************************
*FIX_LOG_FILE=MCTradesTSE
FIX_LOG_DIRECTORY=logs
*
*********************************
* BROKER_LIST To filter own trades*
*********************************
BROKER_LIST=ABN01
*
*********************************
* ORDERS TRADES CAPTURE filter    *
* SET TO TRUE TO ENABLE CAPTURE OF*
* ORDERS or TRADES or BOTH        *
*********************************
TRADES=TRUE
ORDERS=TRUE
*
*********************************
* Order Parameters                *
*********************************
TRDSES_1_CLORD_PREFIX =TD12479A
TRDSES_2_CLORD_PREFIX =TD12479B
*
*********************************
* WAKE UP/SHUT DOWN               *
*********************************
*
```

```
TRDSES_1_WAKE_TIME =08:20
TRDSES_1_SHUT_TIME =16:30
TRDSES_2_WAKE_TIME =19:00
TRDSES_2_SHUT_TIME =23:59
**********************************
* Market Closed ?                 *
**********************************
CLOSE_TIME=18:15
* Check after CLOSE_TIME
**********************************
* 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 MCTradesTSE 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.

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,TSE,TSE,20120710,TDJNX,TD12479A,23,20120710-44,TD12479A1,20120710-92,0,2,ABN01,400,7203,3200,1280000.00,10:49:12,20120710-00:49:12,1,400,3200,400,20120710-00:49:12,2,0,~

Orders

JAPAN,TSE,TSE,20120710,TDJNX,TD12479A,2,20120710-44,TD12479A1,20120710-7d,0,0,,ABN01,7203,10:48:53,20120710-00:48:53,1,400,,,0,20120710-00:48:53,2,~

Note: Additional examples are available from RJE.

5.3 Trades File:

A Trade file is produced each run 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 of all trades.

On a restart the internal copy of the trade is recreated from the FIX Message Log and a new Trades File is produced.

5.4 Orders File:

An Order file is produced each run 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 a orders feed of all orders.

APP_DATA_DIRECTORY=Directory where the Trades file is stored.

APP_DATA_DIRECTORY defaults to APP_LOG_DIRECTORY if not specified.

6 FIX Session Sequence Numbers:

Typically FIX session sequence numbers start from 1 each day.

By default when reconnecting/restarting the sequence numbers at both ends continue on from their previous values and any missing messages are recovered.

This is especially true for TSE as the version of this they are using (4.2) does not support resetting sequence numbers.

Hence, on a restart the application reprocesses the FIX Message log to reestablish from/to sequence numbers.

6.1 FIX Message Log:

But, typically the FIX session is continued across runs and there is a single FIX Message log for each day. In that mode, traffic sent / received (including trades) is recovered from the FIX Messages log at startup.

Hence all trades is rebuilt from the message log each start up.

When resuming the FIX session the application typically only fetches new trades.

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

For MCTradesTSE you should never delete the FIX message log. In a rare event that the log file is corrupted, you should rename the file.

6.2 Missing FIX Message Log:

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

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

For MCTradesTSE this can cause problems with the sequence number of the login message we send to the TSE trading system.

If the sequence number is less than expected TSE 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 MCTradesTSE.PDF - 6.2 Missing Fix Message Log.                                         ***
*** You can run MCTradesTSE -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: 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:- MCTradesTSE –S nnn – 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 TSE administrator or TSE administrator can reset the FIX session (Last option).

In this mode the application will re-request all trades for the day from TSE.

If you specify an incorrect sequence number we will still attempt to recover, but it is strongly recommend you restart with the correct number.

7 Command Clients Order Cancellation:

7.1 Order Cancellation Overview:

MCTradesTSE supports cancellation of orders via:-

- Web Clients – The final production system.
- GUI Clients – TestClient.exe – temporary fallback option

The following FIX messages are related to order cancellation

- OrderCancelRequest
- OrderCancelAcknowledgement
- OrderCancelReject

When an order cancel request succeeds an OrderCancelAcknowledgement message, which is one kind of Execution Report message, is sent by the exchange. An OrderCancelReject is only sent when an order cancel request fails.
MCTradesTSE counts these to provide an estimate of the number of orders cancelled.

7.2 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|~

7.3 Command Clients Parameters:

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

7.4 Cancellation SQL Database Updates:

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

Table :- **tse_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.

8 SQL Database:

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

The MCTradesTSE application uses the “npgsql” .net data provider for PostgreSQL.

The MCTradesTSE calls ProgreSQL Functions (Stored Procedures) for database access and updating.

8.1 SQL Database Tables:

Table - system

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

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 accept order cancellation requests.

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 TSE 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 – tse_orders

This is the main table of interest to Web Clients and other applications which wish to display 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. An archiving process could be developed if required.

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

```
-- Table: tse_orders

-- DROP TABLE tse_orders;

CREATE TABLE tse_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 tse_order_pkey PRIMARY KEY (system_id, order_id)
)
WITH (
  OIDS=FALSE
);
ALTER TABLE tse_orders OWNER TO postgres;
GRANT ALL ON TABLE tse_orders TO postgres;
```

Table – tse_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 :- tse_update_order_cancel() - updates this table and the tse_trans table with the results of each order cancel request.

Table – tse_trans

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

Table – user

This table is used by web clients for access control. This area of the system is still be finalized. This table is not accessed by MCTradesTSE.

Table – tse_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.

Fucntions:- tse_get_cl_ord() and tseupdate_cl_ord()

8.2 SQL Database Parameters:

The use of an SQL Database is optional but Order Cancelation and the Web Client facility will not work properly if it is not enabled.

All SQL Database access occurs in a separate the thread it should not affect the performance of the rest of the MCTradesTSE application.

If we detect that database updating cannot keep up with the rate the TSE's data, the program can be enhanced with multiple database updating threads.

SQL_DATABASE_NAME=Name of the database to access. The presence of this parameter turns on database processing. All tables and functions mentioned in
SQL_DATABASE_NAME=webdb

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

SQL_DATABASE_PORT=Port for the PostgreSQL database.
SQL_DATABASE_PORT=5432

SQL_USER_ID=PostgreSQL database user.
SQL_USER_ID=postgres

SQL_PASSWORD= PostgreSQL database user password*
SQL_PASSWORD=rjexxxxxx

8.3 npgsql files

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

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

These files are the “npgsql” .net data provider for PostgreSQL.

8.4 SQL Script Files

The following files create database tables:-

create_tse_last_clord_id.sql
create_tse_order_cancel_result.sql
create_tse_orders.sql
create_system.sql
create_system_state.sql
create_tse_trans.sql
create_user.sql

The following files create database functions:-

func_tse_get_cl_ord.sql
func_tse_update_cl_ord.sql
func_tse_update_orders.sql
func_tse_update_order_cancel.sql
func_get_system_info.sql
func_update_system_state.sql