

Revision:.....	2
Similar MCTrades Products:	2
1. Overview	3
1.1 Features:	4
1.2 GUI Screen:	6
2. Daily Cycle.....	8
3. Installation	8
4 Configuration.....	9
4.1 FIX Connection Parameters:	9
4.2 FIX Logon Parameters:	9
4.3 Optional FIX Parameters:	10
4.4 Trade Feed Parameters:	10
4.5 Order Feed Parameters:	10
4.6 Logging Parameters:	11
4.7 Daily Cycle Parameters:	11
4.8 Feature Filter Parameters:.....	12
4.9 Other Parameters:	12
4.10 Configuration File Example :	12
5 Comma-Delimited Application Development	14
5.1 Comma-Delimited Header:.....	14
5.2 Comma-Delimited Data:.....	14
5.3 Trades File:.....	16
5.4 Orders File:	16
6 Message Sequence Numbers	16
6.1 FIX Message Log:	16
6.2 Missing FIX Message Log:	17
6.3 Specifying a Restart Sequence No:.....	17
7 Features for Testing	18
7.1 Stop and Start:	18
7.2 User set Message Sequence Numbers:	19
8 Command Client:.....	20
8.1 Order Cancellation:	20
8.2 Supported Cancellation Types:.....	20
8.3 Command Clients Parameters:	21
8.4 Cancellation Database Updates:	21
9 Database	21
9.1 Database Tables:.....	21
9.2 Database Parameters:.....	28
9.3 npgsql files:	28
9.4 SQL Script Files:	28

This Document:

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

Revision:

01/03/2012 – S.C. – Produced the first version of this manual.

14/08/2012 – Vaasugi – Made required modifications.

Similar MCTrades Products:

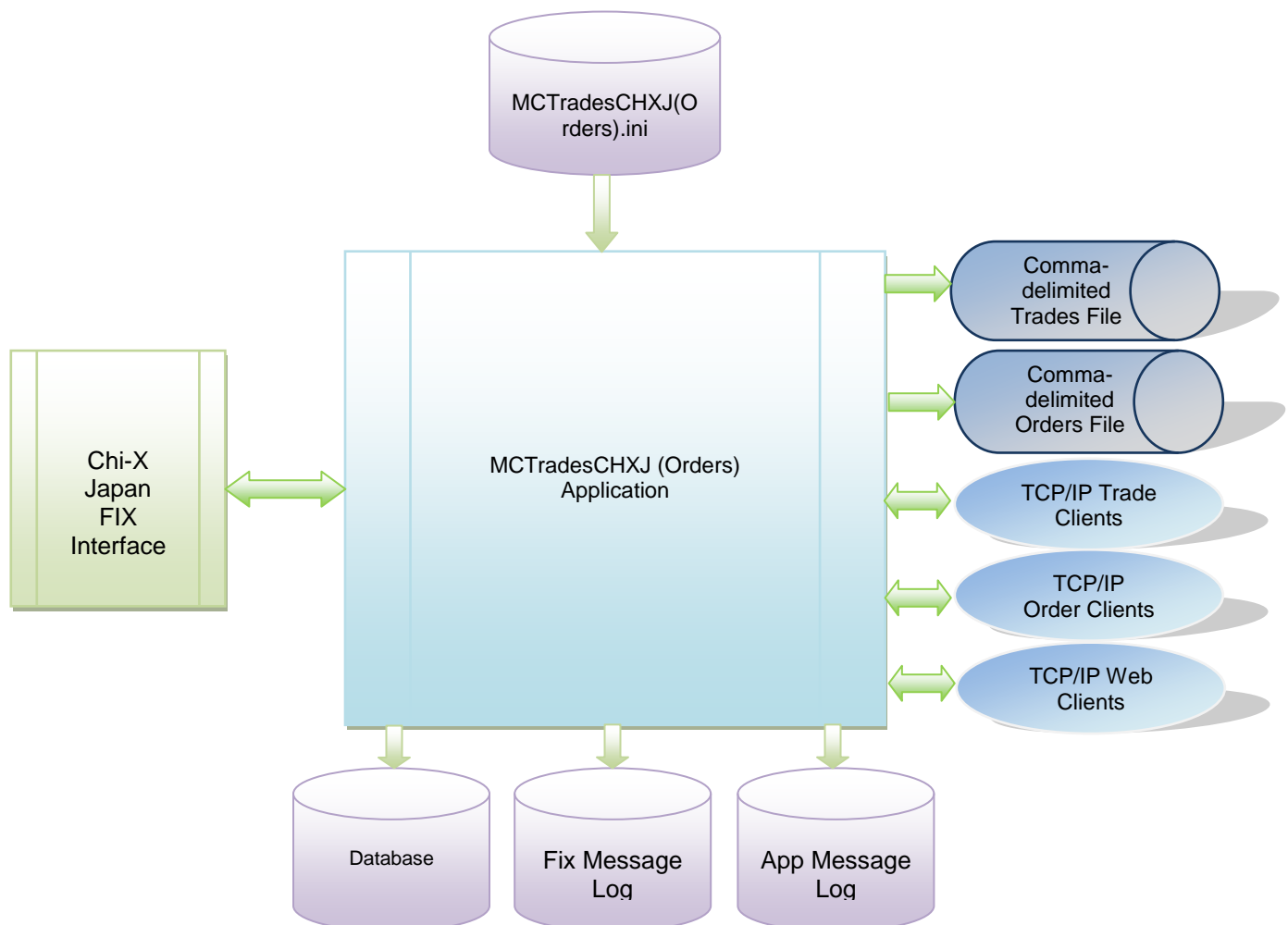
Similar MCOOrders products exist for other exchanges:- ASXTrade, ASX24, Chix Aust, HKex, SGX, TSE, OSE, SBI contact RJE for more details.

1. Overview

MCTradesCHXJ (Orders) application communicates with the FIX Interface of Chi-X Japan 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 MCTradesCHXJ (Orders) production system.





1.1 Features:

Order Placement/Cancellation

Mainly for the purpose of testing, MCTradesCHXJ (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 MCTradesCHXJ (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 Chi-X Japan 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 Chi-X Japan 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 Chi-X Japan Trading System are stored in the database table 'chx_j_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 MCTradesCHXJ(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:



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 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

MCTradesCHXJ (Orders) 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](#):

Note: We currently have no way of detecting Market Close in Chi-X Japan FIX service. A timed shutdown is the only option.

3. Installation

Install MCTradesCHXJ (Orders) as follows:-

<Install Directory> :- MCTradesCHXJ.exe, MCTradesCHXJ(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 MCTradesCHXJ.exe, with the presence of a correctly configured MCTradesCHXJ(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 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 MCTradesCHXJ(Orders).ini

4.1 FIX Connection Parameters:

FIX_SERVER_HOST = Name of Chi-X Japan Fix Server
e.g **FIX_SERVER_HOST**=CHXJ_FIX_GW

FIX_SERVER_PORT = Port to connect Chi-X Japan Fix Server
e.g **FIX_SERVER_PORT**=31312

Chi-X Japan will supply values for these parameter settings.

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

4.2 FIX Logon Parameters:

FIX_SENDER_ID = Part of Fix header, a valid value must be specified.
e.g **FIX_SENDER_ID**= ABN01_TEST

FIX_TARGET_ID = Part of Fix header, a valid value must be specified.
e.g **FIX_TARGET_ID**=CTS_TEST

Chi-X Japan 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 Chi-X Japan 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:

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.

e.g **APP_LOG_FILE**= MCTradesCHXJ

The name of the file e.g MCTradesCHXJ.App.Messages.20120808_093415.log

FIX_LOG_FILE = file base for FIX Message Log; the filename always includes the current date.

e.g **FIX_LOG_FILE**= MCTradesCHXJ

The name of the file e.g MCTradesCHXJ.Fix.Messages.20120423.log

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_TIME = time when program wakes up each morning (hour:min), default 08:00.

e.g **WAKE_TIME**=08:00

SHUT_TIME = time when the program shutdown (hibernation) occurs (hour:min)
default 16:00.

e.g **SHUT_TIME**=16:00

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 Database Parameters:](#)

4.10 Configuration File Example :

```
*****
* APPLICATION VERSION          *
*****
APP_VERSION=MCTradesCHXJ - RJE Systems P/L 2012
*****
* RUN WITHOUT GUI             *
*****
*NO_GUI=YES
*****
* RUN WITHOUT TEST ORDERS    *
*****
*NO_TEST_ORDERS=YES
*****
* FIX Session properties     *
```



MCTradesCHXJ(orders)

Chi-X Japan Trading System Order Placement/Cancellation

```
*****
FIX_SERVER_HOST=CHXJ_FIX_GW
FIX_SERVER_PORT=31312
FIX_SENDER_ID=ABN01_TEST
FIX_TARGET_ID=CTS_TEST
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=MCTradesCHXJ
APP_LOG_DIRECTORY=logs
LOGGING_LEVEL=9
*****
* Fix Log File properties *
*****
FIX_LOG_FILE=MCTradesCHXJ
FIX_LOG_DIRECTORY=logs
*****
* BROKER_LIST To filter own trades*
*****
BROKER_LIST=ABN01
*****
* FEATURE FILTER *
*****
TRADE_FEED=TRUE
ORDER_FEED=TRUE
*****
* WAKE UP/SHUT DOWN *
*****
WAKE_TIME=08:00
SHUT_TIME=16:00
*****
* 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
***** END *****
```

5 Comma-Delimited Application Development

One option for developers is to make a TCP/IP separate connections to MCTradesCHXJ 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 Chi-X 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:2,exchange|S:4,market|S:4,trade_date|D,firm_id|S,trader_id|S,trade_no|N,order Id|S,clOrderId|S,exec_id|N:10,execRefID|N:10,execTransType|N:1,OrdStatus|N:1,LastCapacity|N:1,Account|S,quantity|N:9,sec_code|S,price|N:12.7,value|N:18.2,time_trade|T,utc_timestamp|TS,Side|C,OrderQty|N,AvgPrice|N:12.7,CumQty|N,TransactTime|TS,ExecBroker|S,ClientID|S,ExecType|C,LeavesQty|N:10,ClientCrossRef|S,TransactID|S,TradeLiquidityInd|S:2,SecurityID|S,IDSsource|N:10,SecurityExchange|S,~

Orders

firm_id|S,trader_id|S,order_id|N,cl_order_id|N,exec_id|N,exec_trans_type|N,order_status|N,ord_reject_reason|S,account|S,exchange_code|S,symbol|S,side|S,order_qty|N,price|N,last_shares|N,cum_qty|N,transact_time|T,process_code|C,exec_inst|C,shared|C,shared_group_id|S,shared_trader_id|S,text|S,order_type|N,expire_time|T,commodity|S,month_year|S,month|N,year|N,unique_key|S,~

5.2 Comma-Delimited Data:

Fields that are not relevant are simply empty.

Trades

JAPAN,CHXJ,CHXJ,20120808,CTS_TEST,ABN01_TEST,29,5100,1245000,B20120808140001129-0,,0,2,2,ABN01,400,7203,3200.00,1280000.00,12:48:11,20120808-03:48:11,1,400,3200.00,400,20120808-03:48:11.148,,ABN01,2,0,,A,,,~

Orders

CTS_TEST,ABN01_TEST,5100,1245000,A-1305776,0,0,,ABN01,,7203,1,400,0.00,0,0,20120808-03:45:39.688,,,,,,2,,72,03,,2013,5100|A-1305776|20120808-03:45:39.688,~

Note: Additional examples are available from RJE.

5.3 Trades File:

A Trade file is produced for each day 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-day, 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.
e.g MCTradesCHXJ.20120808.trades

5.4 Orders File:

An Order file is produced for each day 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-day, 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.
e.g MCTradesCHXJ.20120808.orders

6 Message Sequence Numbers

Message Sequence Numbers start from 1 each day. By default when reconnecting/restarting mid-day, 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. The version of FIX protocol being used (version 4.2) doesn't support resetting 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 day. 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:](#)

Note: You should never delete the FIX message log, if the rare event that 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.

This can cause problems with the sequence number of the login message we send to the Chi-X Trading System. If the sequence number is less than expected Chi-X 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 correct then could be a problem with the Fix Message log. ***  
*** See MCTradesCHXJ.PDF - 6.2 Missing Fix Message Log. ***  
*** You can run MCTradesCHXJ -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

MCTradesCHXJ -S nnn

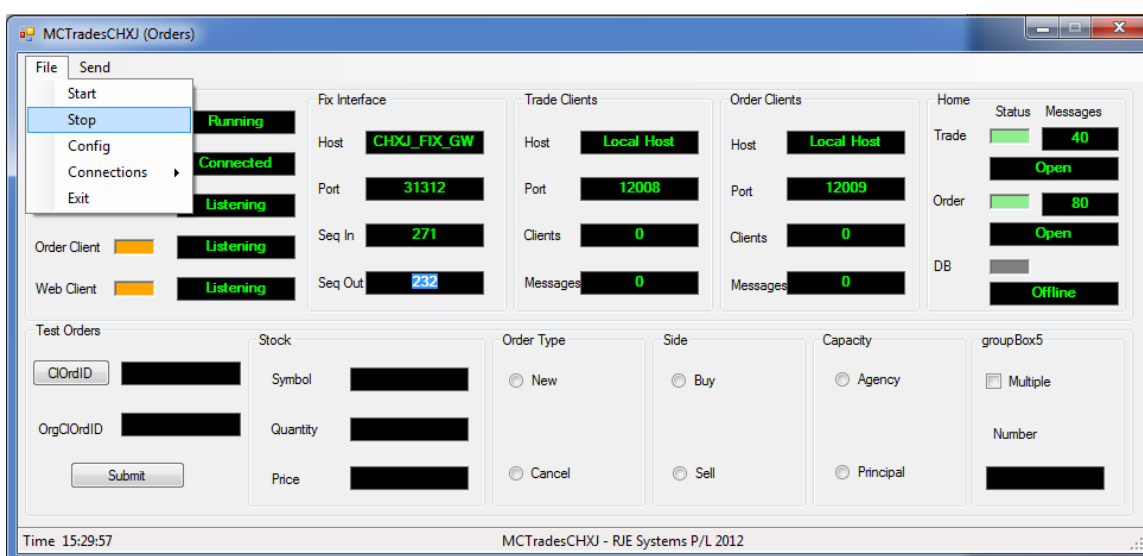
Note: Where nnn is the sequence number.

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 Chi-X administrator or he can reset the FIX session (as the last option). In this mode the application will re-request all trades for the day from Chi-X.

7 Features for Testing

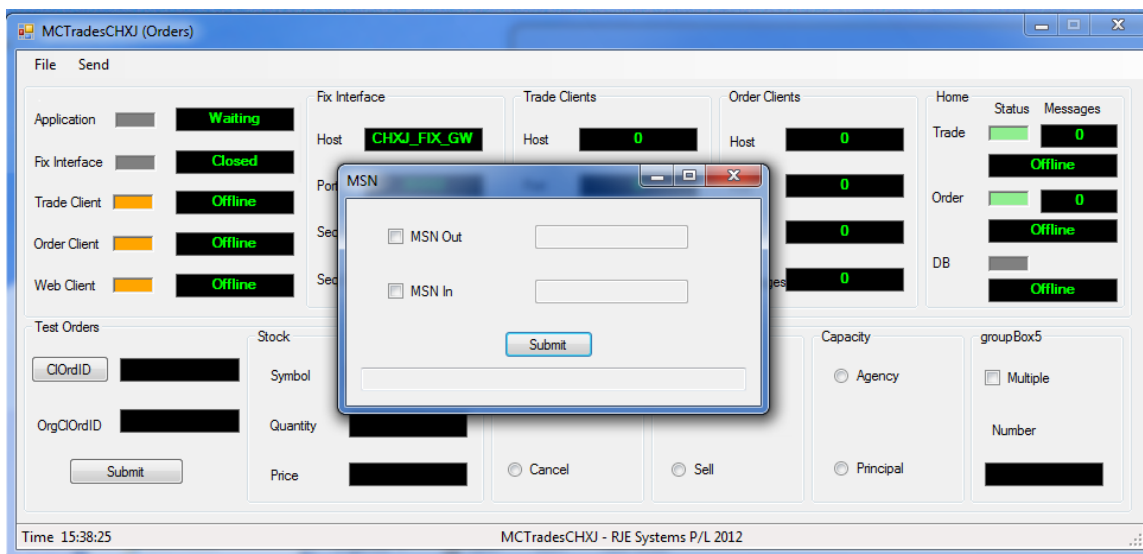
The following features are included solely for the purpose of facilitating the testing of this application.

7.1 Stop and Start:



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 MCTradesCHXJ(Orders) application. 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.

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 MCTradesCHXJ(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. MCTradesCHXJ(Orders) stores the cancel results in 'chx_j_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 :- **chx_j_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.

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

Table :- **chx_j_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 PostgreSQL database running under Windows and Linux.

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

9.1 Database Tables:

Table - system

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

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 Chi-X Japan 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 – chx_j_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 :- `chx_j_update_order()` – Updates the orders table for each execution report transaction.

```
-- Table: chx_j_orders
-- DROP TABLE chx_j_orders;
CREATE TABLE chx_j_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 chx_j_order_pkey PRIMARY KEY (system_id, order_id)
)
WITH (
    OIDS=FALSE
);
ALTER TABLE chx_j_orders OWNER TO postgres;
GRANT ALL ON TABLE chx_j_orders TO postgres;

```

Table – chx_j_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:- chx_j_get_cl_ord() and chx_j_update_cl_ord()

```
-- Table: chx_j_last_clord_id
```



```
-- DROP TABLE chx_j_last_clord_id;

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

Table – chx_j_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 :- chx_j_update_order_cancel() - updates this table and the chx_j_trans table with the results of each order cancel request.

```
-- Table: chx_j_order_cancel_result
-- DROP TABLE chx_j_order_cancel_result;

CREATE TABLE chx_j_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 chx_j_order_cancel_result OWNER TO postgres;
GRANT ALL ON TABLE chx_j_order_cancel_result TO postgres;
```

Table – chx_j_trans

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

```
-- Table: chx_j_trans

-- DROP TABLE chx_j_trans;

CREATE TABLE chx_j_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 chx_j_trans OWNER TO postgres;
GRANT ALL ON TABLE chx_j_trans TO postgres;

-- Index: "chxjIndOrderID"

-- DROP INDEX "chxjIndOrderID";

CREATE INDEX "chxjIndOrderID"
    ON chx_j_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=rjlinuxlap

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=rjxxxxxx

9.3 npgsql files:

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

- 1)CREATE TABLE system
- 2)CREATE TABLE system_state
- 3)CREATE TABLE chx_j_last_clord_id
- 4)CREATE TABLE chx_j_orders
- 5)CREATE TABLE chx_j_trans
- 6)CREATE TABLE chx_j_order_cancel_result

The following files create database functions:-

- 1)FUNCTION get_system_info
- 2)FUNCTION update_system_state
- 3)FUNCTION chx_j_get_cl_ord
- 4)FUNCTION chx_j_update_cl_ord
- 5)FUNCTION chx_j_update_order
- 6)FUNCTION chx_j_update_order_cancel