MCTradesCHXJ (DropCopy)

Chi-X Japan Trading System Trade/Order Feeds

This Document:

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

Revision:

01/03/2012 – S.C. – Produced the first version of this manual.
14/08/2012 – Vaasugi – Made required modifications.

Similar MCTrades Products:

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

1. Overview

MCTradesCHXJ (DropCopy) application communicates with the DropCopy Interface of Chi-X Japan Trading System via FIX protocol. It extracts Orders/Trades from the consolidated feed of FIX messages supplied by the DropCopy Service. It then provides the Orders/Trades via comma-delimited feeds as well as stores them in the Database.

The following diagram depicts the overall functionality and connectivity of the MCTradesCHXJ (DropCopy) production system.

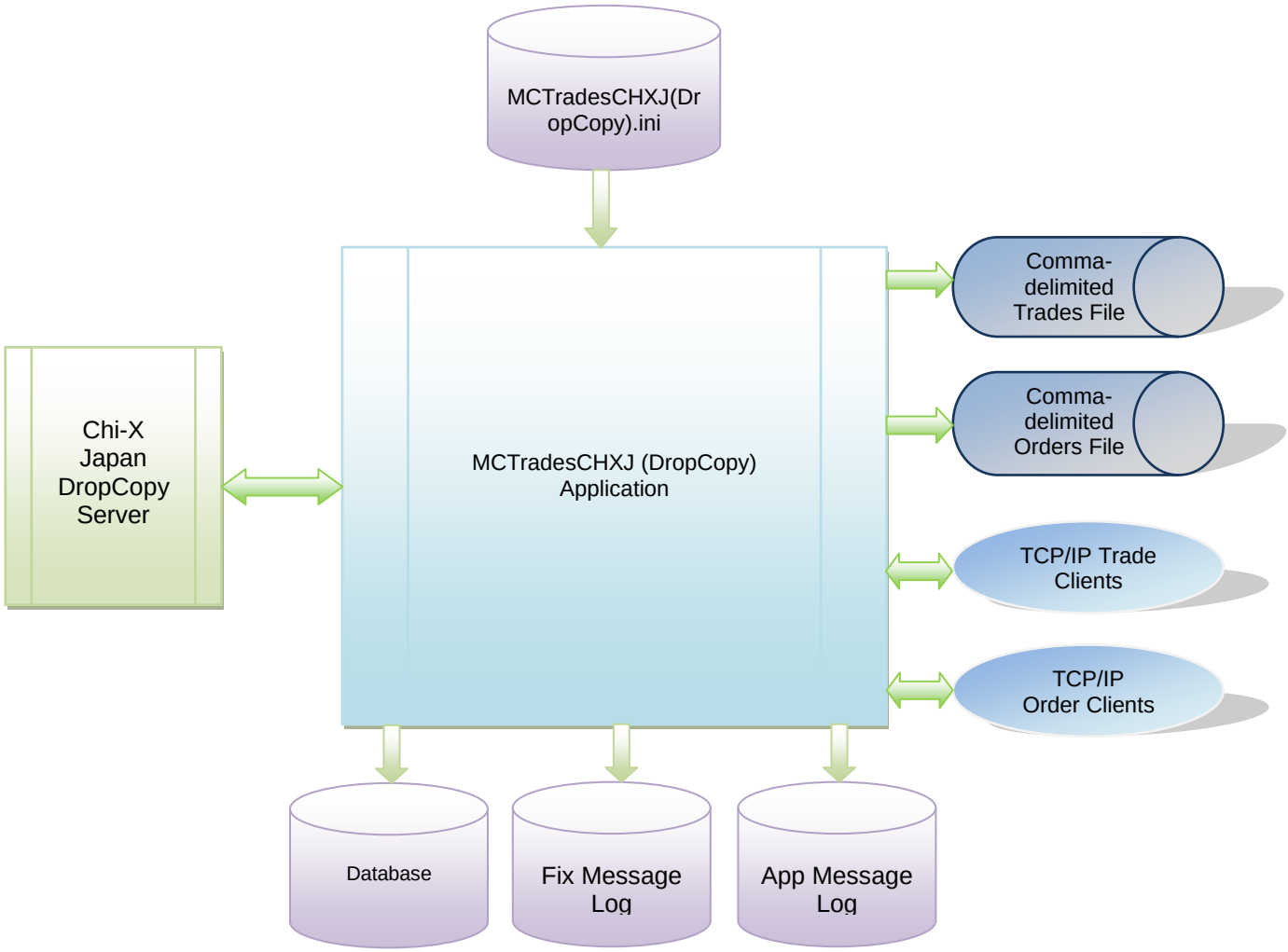


Figure 1. The MCTradesCHXJ (DropCopy) Production System

1.1 Features:

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)

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)

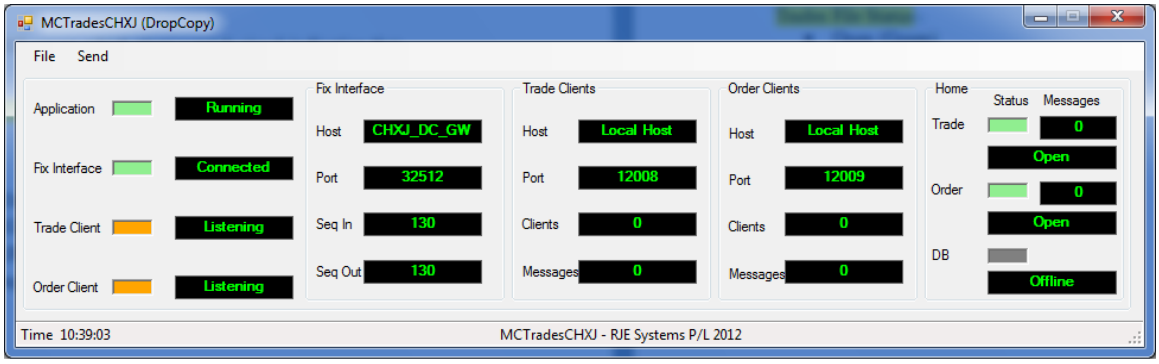
Note: The Comma-Delimited TCP/IP feed is similar to all other MCTrades products.

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 8. Database

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)

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 (DropCopy) 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 DropCopy service. A timed shutdown is the only option.

3. Installation

Install MCTradesCHXJ (DropCopy) as follows :-

<Install Directory> :- MCTradesCHXJ.exe, MCTradesCHXJ(DropCopy).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(DropCopy).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(DropCopy).ini

4.1 FIX Connection Parameters:

FIX_SERVER_HOST = Name of Chi-X Japan DropCopy Server
e.g **FIX_SERVER_HOST**=CHXJ_DC_GW

FIX_SERVER_PORT = Port to connect Chi-X Japan DropCopy Server
e.g **FIX_SERVER_PORT**=32512

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

Note: IP address of the Drop Copy Server Host can be entered in the ini file. Alternatively an entry for the Drop Copy Server Host can be made in the windows host file (C:\Windows\System32\drivers\etc\hosts) with a name for example CHXJ_DC_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**=ABND1

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

DB_CONNECTION=TRUE – set TRUE to enable database connection and storing of orders in the database

4.9 Other Parameters:

NO_GUI=YES – set to enable the application running with no GUI

NO_TEST_ORDERS=YES – this should always be set to YES

See also

- 8.2 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      *
*********************************
FIX_SERVER_HOST=CHXJ_DC_GW
FIX_SERVER_PORT=32512
FIX_SENDER_ID=ABND1
FIX_TARGET_ID=CTS_TEST
FIX_HEARTBEAT=30
*********************************
* TCP Clients properties      *
*********************************
TRADES_PORT=12008
ORDERS_PORT=12009
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
DB_CONNECTION=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,orderId|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,TradeLiquidInd|S:2,SecurityID|S,IDSource|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 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 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 corrent 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

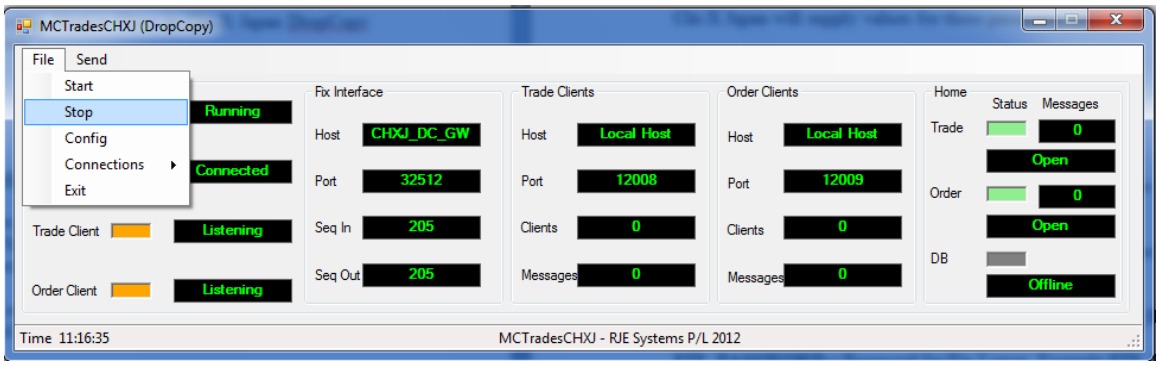
Note: Where nnn is the sequence number.

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

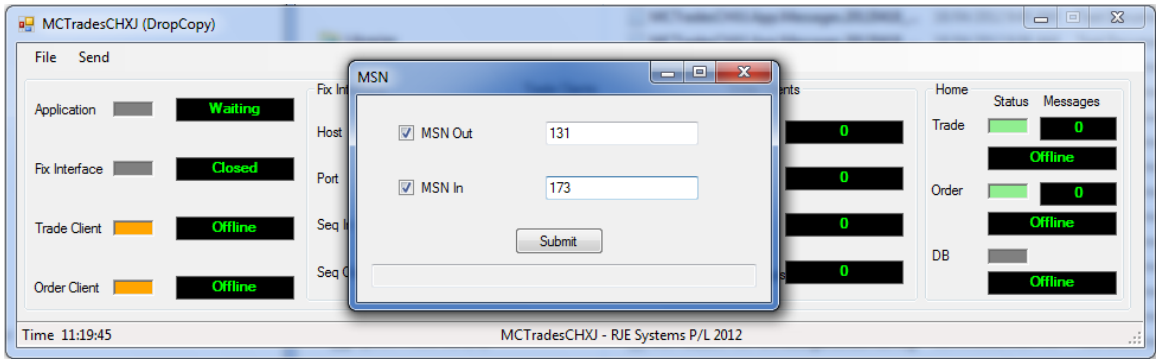
The following features are included solely for the purpose of facilitating the testing of this application.

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(DropCopy) 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 Database

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

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

8.1 Database Tables:

Table - system

The purpose of this table is to allocate a unique Guid (uuid) to each system. In this context MCTradesCHXJ(DropCopy) is one system. All data of MCTradesCHXJ(DropCopy) 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;
```

8.2 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

8.3 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.

8.4 SQL Script Files:

The following files create database tables:-

1)CREATE TABLE system
2)CREATE TABLE system_state
3)CREATE TABLE chx_j_orders

The following files create database functions:-

1)FUNCTION get_system_info
2)FUNCTION update_system_state
3)FUNCTION chx_j_update_order

Appendix 1

Order Feed:

Field Number	Field Name in Output	Presence	Fix Field Name	Fix Tag
1	Country	{Always}	(Internal)	-
2	Exchange	{Always}	(Internal)	-
3	Market	{Always}	(Internal)	-
4	FirmID	{Always}	SenderCompID	49
5	TraderID	{Always}	TargetCompID	56
6	ClientID	{Not Always}	ClientID	109
7	MsgSeqNo	{Always}	MessageSeqNo	34
8	OrderID	{Always}	OrderID	37
9	ClOrdID	{Not Always}	ClOrdID	11
10	OrigClOrdID	{Cancel/Replace}	OrigClOrdID	41
11	ExecID	{Always}	ExecID	17
12	ExecRefID	{Trade Cancel}	ExecRefID	19
13	ExecInst	{Not Always}	ExecInst	18
14	ExecType	{Always}	ExecType	150
15	ExecBroker	{Not Always}	ExecBroker	76
16	Account	{Not Always}	Account	1
17	Symbol	{Always}	Symbol	55
18	Side	{Always}	Side	54
19	Price	{Always}	Price	44
20	OrderQty	{Always}	OrderQty	38
21	OrderType	{Not Always}	OrdType	40
22	OrderCapacity	{Not Always}	OrderCapacity	47
23	OrderStatus	{Always}	OrdStatus	39
24	ExecTransType	{Always}	ExecTransType	20
25	Description	{Not Always}	Text	58
26	AvgPrice	{Always}	AvgPx	6
27	LastPrice	{Trade }	LastPx	31

28	LastShares	{Trade }	LastShares	32
29	LastCapacity	{Trade}	LastCapacity	29
30	CumQty	{Always}	CumQty	14
31	LeavesQty	{Always}	LeavesQty	151
32	TimeInForce	{Not Always}	TimeInForce	59
33	ExpireTime	{Not Always}	ExpireTime	126
34	MaxFloor	{Not Always}	MaxFloor	111
35	PegDifference	{Not Always}	PegDifference	211
36	PostOnly	{Not Always}	PostOnly	8021
37	TradeLiqIndicator	{Trade}	TradeLiquidityIndicator	9882
38	ExecRestatementReason	{Peg Order Resume}	ExecRestatementReason	378
39	TransactTime (in UTC)	{Always}	TransactTime	60
40	ExecutedDate (Local)	{Always}	Date(TransactTime)	-
41	ExecutedTime (Local)	{Always}	Time(TransactTime)	-
42	TimeStamp (UTC)	{Always}	TransactTime	-

Trade Feed:

Field Number	Field Name in Output	Presence	Fix Field Name	Fix Tag
1	Country	{Always}	(Internal)	-
2	Exchange	{Always}	(Internal)	-
3	Market	{Always}	(Internal)	-
4	FirmID	{Always}	SenderCompID	49
5	TraderID	{Always}	TargetCompID	56
6	ClientID	{Not Always}	ClientID	109
7	MsgSeqNo	{Always}	MessageSeqNo	34
8	OrderID	{Always}	OrderID	37
9	ClOrdID	{Not Always}	ClOrdID	11
10	ExecID	{Always}	ExecID	17
11	ExecRefID	{Trade Cancel}	ExecRefID	19
12	ExecType	{Always}	ExecType	150

13	ExecBroker	{Not Always}	ExecBroker	76
14	Account	{Not Always}	Account	1
15	Symbol	{Always}	Symbol	55
16	Side	{Always}	Side	54
17	OrderQty	{Always}	OrderQty	38
18	OrderType	{Not Always}	OrdType	40
19	OrderStatus	{Always}	OrdStatus	39
20	ExecTransType	{Always}	ExecTransType	20
21	AvgPrice	{Always}	AvgPx	6
22	LastPrice	{Always}	LastPx	31
23	LastShares	{Always}	LastShares	32
24	LastCapacity	{Not Always}	LastCapacity	29
25	TradeValue	{Always}	(Derived)	-
26	CumQty	{Always}	CumQty	14
27	LeavesQty	{Always}	LeavesQty	151
28	TradeLiqIndicator	{Not Always}	TradeLiquidityIndicator	9882
29	TransactTime (in UTC)	{Always}	TransactTime	60
30	ExecutedDate (Local)	{Always}	Date(TransactTime)	-
31	ExecutedTime (Local)	{Always}	Time(TransactTime)	-
32	TimeStamp (UTC)	{Always}	TransactTime	-

Order Table:

Field Number	Colum Name	Fix Field Name	Fix Tag
1	country	(Internal)	-
2	exchange	(Internal)	-
3	market	(Internal)	-
4	firm_id	SenderCompID	49
5	trader_id	TargetCompID	56
6	client_id	ClientID	109
7	message_no	MessageSeqNo	34
8	order_active	(Derived)	-

9	order_id	OrderID	37
10	clord_id	ClOrdID	11
11	exec_id	ExecID	17
12	exec_type	ExecType	150
13	account	Account	1
14	symbol	Symbol	55
15	side	Side	54
16	price	Price	44
17	order_qty	OrderQty	38
18	order_type	OrdType	40
19	order_capacity	Capacity	47
20	order_status	OrdStatus	39
21	exec_trans_type	ExecTransType	20
22	description	Text	58
23	avg_price	AvgPx	6
24	last_price	LastPx	31
25	last_fill	LastShares	32
26	no_of_fills	(Derived)	-
27	cum_qty	CumQty	14
28	leaves_qty	LeavesQty	151
29	time_in_force	TimeInForce	59
30	min_qty	MinQty	110
31	max_floor	MaxFloor	111
32	transact_time	TransactTime	60
33	trade_date	TradeDate	75
34	time_stamp	(Database Update Time)	-