SFEOrders

Related Documents:

SYCOM Implementation of FIX Specification Version 4.0 (SYCOM_fix_spec/pdf)
SYCOM Fix Introduction (SYCOM_fix_introduction.pdf)

Other Information:

Please consult the RJE Web Site www.rje.com.au for the latest information on RJE products.

This Document:

SGXOrders – SYCOM Fix Order Capture - describes how SGXOrders interacts to the SYCOM Fix (AOEI) System to capture details of Execution reports data.

Revision:

15/12/2011 – Web Release – SGXOrders Rev 01_04.
21/12/2009 – Initial Document – SGXOrders Rev 01_01.

1. Overview:

The SFEOrders product has been extended include the following components:-

- Order Cancellation
- SQL Database
- Web Client

1.1 SFEOrders:

The program connects to the SYCOM Fix (AOEI) interface and collects execution reports. The contain order transactions and details of Fills and other trade transactions. So the system could be more accurately called an “Orders + Trades” feed.

The Orders data is available via a TCP/IP port as per other RJE products..

Where possible the field tags used are similar to other RJE products (e.g. MCTradesASX).

The configuration settings for the system are found in SFEOrders.ini

1.2 Order Cancellation:

SFEOrders supports cancellation of orders via:-

- Web Clients
- GUI Clients

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. Command Clients – Order Cancelation:

1.3 SQL Database:

An optional SQL database can be configured to store orders and related information.

The SQL Database is required for Order Cancellation and for the Web Client.

More details can be found in :- 9. SQL Database:

1.4 Web Client:

A PHP based web client application can be used for view / cancel orders.
See :-10. PHP – Web Client:

2. Daily Cycle:

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

Refer:- 3.2.6 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 RJE products.

3. Installation/Configuration:

3.1 Installation:

Simply install SFEorders as follows :-

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

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

You must set the following parameters correctly:-

- 3.2.1 Fix Connection Parameters
- 3.2.2 Fix Logon Parameters

Note: If you which to run the program without a GUI refer:-3.2.7 Other parameters:

3.2 Configuration:

All configuration parameters are stored in SFEOrders.ini

3.2.1 FIX Connection Parameters:

FIX_SERVER_HOST = Name of SYCOM Fix Server.
e.g. FIX_SERVER_HOST=SFEAS210

FIX_SERVER_PORT = Port to connect to for FIX.
e.g. FIX_SERVER_PORT=3160

3.2.2 FIX Logon Parameters:

FIX_SENDER_ID = SFE Assigned Firm Code/Acronym. Part of Fix header, a valid value must be specified..
e.g. FIX_SENDER_ID= FCS

FIX_SENDER_SUB_ID= Trader ID. Part of Fix header, a valid value must be specified.
e.g. FIX_SENDER_SUB_ID=FJC

FIX_USER_ID = user for Fix Logon.
e.g. FIX_USER_ID= FJC

FIX_PASSWORD = Password for Fix Logon.
e.g. FIX_PASSWORD= fjcfjc

3.2.3 Optional FIX Parameters:

FIX_HEARTBEAT=<Heartbeat interval> (Seconds) – default = 60.
e.g. FIX_HEARTBEAT=1 – Should be set to 1 for SYCOM.

FIX_NEW_SESSION=YES – Should always be YES. (Parameter currently ignored)
Refer:-6. FIX Session Sequence Numbers:

3.2.4 Orders Feed Parameters:

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

The format of the data is described in 4. Application Development:.

ORDERS_PORT = TCP/IP port for all Orders.
e.g. ORDERS_PORT=12008

3.2.5 Logging Parameters:

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= SFEOrders
filename= SFEOrders_20080429_150113.log.

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

LOGGING_LEVEL= Set the level of application message logging, can turn on additional diagnostic messages.
LOGGING_LEVEL=9

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

FIX_LOG_FILE = file base for FIX Message Log, the filename always includes the current date.
e.g. FIX_LOG_FILE=fix.messages, filename= fix.socket.20080429.log

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

Refer:-6.1 Fix Message Log:

3.2.6 Daily Cycle Parameters:

Refer:- 2. Daily Cycle:

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

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

3.2.7 Other parameters:

NO_GUI=YES – Specify this value to run without a GUI, e.g. as a Windows NT service.

3.2.8 New parameters – 01.04 – Web Client:

See :-

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

4. Application Development:

The only option for developers is to make a TCP/IP connection to SFEOrders output ports and receive trades data in comma-delimited format.

Data is simply sent when it is available; there is no need to request data.

This function is similar to other RJE Products (e.g. MCTradesASX).

4.1 Comma-Delimited Header:

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

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,~

4.2 Comma-delimited Data:

The same format is used for all orders.

Fields that are not relevant are simply empty:-

FCS,FJC,735918,208,2,D,1,,FCS11H,SFE,IRZ0,1,500,92000,50,50,20091217-06:09:39,,,,,,,,IR,Z0,12,2010,735918|2|20091217-06:09:39,~

Additional examples are available from RJE.

Notes:

1. Fields for “commodity” onwards are extracted / derived by RJE.

2. The "account" field will should provide the basis of recognising which Fortis customer has originated the Order.
3. The "Unique Key" field consists of OrderID+ExecID+TransactTime. You should check for duplicate order transactions using this key. Refer 7. Recovery Strategy.
4. When the ExecID field is present it means this transaction is a Trade (Order fill).

4.3 Orders File:

An Orders file is produced each 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 of a trades feed of all trades.

* Refer to 7. Recovery Strategy below; if we lose connection to SYCOM we will close the orders file and open a new copy.

5. Password Changing:

SFEOrders does not support automatic password changing; we don't believe password changing is supported by the SYCOM Fix interface.

6. FIX Session Sequence Numbers:

The SYCOM rules for sequence numbers are as follows:-

The session number (FIX Tag 5006) will only increment when the MFWS application on the AOEI is stopped / started.

Thus Sequence numbers do not start from 1 each day, however they will begin at 1 from the first login made on a Monday morning, and will continue to increment throughout the week. However they will be reset back to 1 if the AOEI is rebooted.

Because of these rules and the limited time frame we have adopted a simplified recovery strategy detail in 7. Recovery Strategy

6.1 Fix Message Log:

A new FIX message log is written each run.

This can be used to diagnose Fix traffic problems.

7. Recovery Strategy:

Each order (/trade) transaction has a unique key:- OrderID+ExecID+TransactTime

This assumes that we could get multiple fills with the same timestamp, but manual order transactions can't have the same timestamp.

Trade modify records can also be sent, but these should have a different timestamp.

In a program restart scenario applications will need to check if they have already stored the record, they should the unique key field, for this check.

When the link to SYCOM drops out, SFEOrders performs the following recovery sequence:-

After some consideration the easiest way for us to handle this is:-

1. Wipe all order data in memory.
2. Close all client connections (kicking off, connected applications).
3. Open a new.orders file.
4. Recover the link then re-download all order data from SYCOM.

In other words the only difference between a between the SYCOM link dropping out and a clean restart is that the program keeps running and attempts to recover.

The other advantage of this scenario is that each restart is a clean restart if things seem to be going wrong you can just restart.

If SFEOrders detects any serious problem, it will just stop running.

8. Command Clients – Order Cancellation:

8.1 Order Cancellation Overview:

SFEOrders supports cancellation of orders via:-

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

Order Cancellation is the following Fix messages:-

- `MsgOrderCancelRequest`
- `MsgOrderCancelReject`

A MsgOrderCancelReject is only sent when an order cancel request fails.

When an order cancel requests succeeds all that is sent is a number of updated Execution Reports. SFEOrders attempts to count these to provide an estimate of the number of orders cancelled.

8.2 Supported Cancellation Types:

Two types of order cancel are currently supported:-

1. Cancel by Account :-
   CANCEL_REQUEST|USER=admin|REQUEST_NO=279|CxlType=8|Account=CLIENT32C|~
2. Cancel Individual Order:-
   CANCEL_REQUEST|USER=admin|REQUEST_NO=297|CxlType=F|OrderID=6688077|~

Notes:-

1. When cancelling by Account sometimes not all orders are cancelled. There database reflects the current state of the orders. If you attempt to cancel an individual order. You can find out why that order was not cancelled typically :- "Cancel - Contract not trading"

8.3 Command Clients Parameters:

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

ORDER_REFRESH=What to do with existing in memory orders when a client requests a refresh.
ORDER_REFRESH=[NO|KEEP|DELETE]

Order Refresh is covered in more detail in :- 8.5 Order Refresh Recovery Considerations:

8.4 Cancellation SQL Database Updates:

Table :- **sfe_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. This table is SFE specific.

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

Table :- **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.

8.5 Order Refresh Recovery Considerations:

During the initial development of the application we chose the recovery strategy detailed in :- 7. Recovery Strategy:. What this means is that if a web client requests a download of orders/trades during a session, duplicate execution reports could be sent to existing Orders TCP/IP stream clients if ORDER_REFRESH=KEEP is chosen.

ORDER_REFRESH defaults to NO as the database should not get out of sync during a session that remains online. Refreshing the database contents should not be necessary.

If it is decided that refreshing of the database is required ORDER_REFRESH=DELETE is recommended. This had the side effect of knocking order TCP/IP stream clients offline hence access to that function should be restricted.

If it becomes necessary we should be able to develop a better solution to this problem. However is of this facility should not be required, we have included it as it was part of the original requirement.

9. SQL Database:

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

The SFEOrders program uses the “npgsql” .net data provider for PostgreSQL.

The SFEOrders program typically calls ProgreSQL Functions (Stored Procedures) for database access and updating. This approach was chose as we believe it will deliver the best performance.

9.1 SQL Database Tables:

9.1.1 Table - system

The purpose of this table is to allocate a unique Guid (uuid) to each system which will be storing orders data.

In this context SFEOrders is one system all data for SFEOrders has the system_id for SFEOrders.

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 sfe;
GRANT ALL ON TABLE "system" TO sfe;
```

9.1.2 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 SFE 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=FALSE
);
ALTER TABLE system_state OWNER TO sfe;
GRANT ALL ON TABLE system_state TO sfe;
GRANT ALL ON TABLE system_state TO public;
```

9.1.3 Table – orders

This is the main table of interest to Web Clients and other application 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'.

On a restart/recovery or Order Refresh order_active is set to 'U'. After recovery any orders which remains as order_active='U' must have been deleted or purged while the system was offline.

We are keeping orders information as it may be useful, if we continue with this approach are archiving process must be developed.

Function :- sfe_update_order() – Updates the orders table for each execution report transaction. Note: It is likely that this function will be specific to the SFE other systems will use the same database table but may have a slightly different update function.

Function :- wipe_current_orders() – Sets order_active to 'U' on restart/recovery. Other system may not have this requirement.

```
CREATE TABLE 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),
....
)
```

9.1.4 Table – 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 :- sfe_update_order_cancel() - updates this table and the trans table with the results of each order cancel request.

9.1.5 Table – trans

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

9.1.6 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 SFEOrders.

9.1.7 Table – sfe_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. This table is SFE specific. We are adhering strictly to the SFE requirements in this regard even if this is not so strictly necessary for order cancellations as by definition a order cancellation transaction cannot span multiple days like an order can.

As a result of implement this approach a SQL Database is required for SFEOrders cancellations.

Fucntions:- get_cl_ord() and update_cl_ord()

9.2 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 SFEOrders application.

If we detect that database updating cannot keep up with the rate the SFE is sending 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=sfe

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=sfe

SQL_PASSWORD= PostgreSQL database user password*
SQL_PASSWORD=rjexxxxxx

* There may be a better way of controlling access to the PostgreSQL database. We have chosen the user/password model to simplify the initial development.

9.3 npgsql files:

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

```
07/07/2007  12:09 AM           282,624 Mono.Security.dll
28/09/2011  08:55 PM           365,568 Npgsql.dll
```

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

9.4 SQL Script files:

The following files create database tables:-

```
15/12/2011  02:46 PM             1,488 create_orders.sql
15/12/2011  02:46 PM               764 create_order_cancel_result.sql
15/12/2011  02:39 PM               467 create_sfe_last_clord_id.sql
15/12/2011  02:36 PM               280 create_system.sql
15/12/2011  02:37 PM               444 create_system_state.sql
15/12/2011  02:38 PM               490 create_trans.sql
15/12/2011  02:37 PM               523 create_user.sql
```

The following files create database functions:-

```
15/12/2011  02:53 PM               568 func_get_cl_ord.sql
15/12/2011  02:52 PM               704 func_get_system_info.sql
15/12/2011  02:54 PM             8,546 func_sfe_update_order.sql
15/12/2011  02:54 PM             3,162 func_sfe_update_order_cancel.sql
15/12/2011  02:55 PM               593 func_update_cl_ord.sql
15/12/2011  02:55 PM             1,427 func_update_system_state.sql
15/12/2011  02:56 PM               711 func_wipe_current_orders.sql
```

10. PHP – Web Client: