NSPOT Trading System Specifications

VERSION 1.3

Release Notice

Document Name	NSPOT Trading System Specifications
Document Type	Functional & Technical Specifications
Version	1.2
Release Date	15 Feb, 2010

Revision History

S.No.	Version	Prepared or Revised By	Reason for preparation/revision	Release Date
1	1.0	Pravin Pillai & Priya Shah		13-Jan-10
2	1.1	Ravindra Sehvade	General Corrections, RMS changes and auto bid functionality	30-Jan-10
3	1.2	Ravindra Shevade	Complete Rewrite. Class and method descriptions have been omitted from this document and will put in a seprate design document. Dealer exposure check included. Market Maker and Broadcast Engine functionality yet to be included.	17-Feb-10
4	1.2	Ravindra Shevade	Second rewrite to reflect changes post implemetation	5-April-2010

TABLE OF CONTENTS

1. Overview

1.1. Company Overview

NCDEX Spot Exchange Ltd. (NSPOT) is a leading electronic Spot Exchange in India. Deriving strength from the expertise of National Commodity and Derivatives Exchange Ltd (NCDEX), NSPOT offers an electronic trading platform for trading in a host of commodities, both agricultural and non-agricultural to various market participants, primary producers including farmers, traders, processors etc. The trading platforms combine technological efficiency and market friendly trading features in a transparent atmosphere to make trading a rich and rewarding experience.

1.2. Project Overview

This project involves building the exchange trading system which comprises of the order routing server for receiving incoming orders and routing them to the appropriate components for further processing, the risk management server to validate the risk against each order, the matching engine to match orders and execute trades, and the broadcast engine to broadcast market data updates to all market participants.

1.3. Acronyms and Abbreviations

Term	Full Form
ORS	Order Routing Server
RMS	Risk Management Server
BE	Broadcast Engine
ME	Matching Engine
FO	Front Office
PFM	Participant's Fund Management
CMS	Commodity Management System
CS tracker	Clearing and Settlement
DNS	Daily Netting Cycle
T2T	Trade to Trade
TM	Trading Member
TAC	Trading Account

PTP	Price Time Priority
MMQ	Market Maker Quote
OTC	Over the Counter
AUC	Auction
IOC	Immediate match or Cancellation
LTP	Last Traded Price
GTS	Good Till Session
AON	All or None
BOV	Buy Order Value
SOV	Sell Order Value
BOQ	Buy Order Quantity
SOQ	Sell Order Quantity
BTV	Buy Trade Value
STV	Sell Trade Value

1.4. Assumptions

Order number generated will be unique for a period of 1 year assuming not more than 9999 order requests per second.

Trade number generated will be unique for a period of 1 year assuming not more than 9999 executions per second.

Matching engine will send the market data directly to the front office in the current phase (and not via broadcast engine)

1.5. Limitations

GTC and GTD orders are not supported in the current phase

Lot validation is not supported in the current phase

Market Maker Quote is not supported in the current phase

Broadcast Engine is not available in the current phase

Fail over, backup and crash recovery is not supported in the current phase

Even though a trading session may span calendar dates, for DNS contracts, session timings should not span calendar dates.

2. Operative Structure

2.1. Segment

Segments are groups of contracts. TMs will have trading permissions to specific segments.

2.2. Contract

Contracts are traded on the exchange. The TM's will be able to place orders (Buy/Sell) for the contracts. Contracts will be a combination of Commodity and Instrument Type. The following parameters will be associated /defined for each contract.

- Contract Symbol
- Commodity
- Instrument Type
- Margin/Stock Contract
- ME Type
- Margins (Buy/Sell)
- Minimum Qty
- Tradeable Lot Size
- Maximum Qty
- Circuit Breaker limits
- Base Price

2.3. Instrument Types

Only one instrument type i.e. spot contract supported. No other instrument types supported. However, certain contracts are termed stock check contracts where a seller needs to have stock prior to putting a sell order. On sell margin contracts, a margin becomes applicable on the sell side orders and trades.

2.4. Contract Settlement Types

2.4.1. T2T

No netting allowed. All trades result into delivery. No netting of positions and orders for margin nor for exposure calculation.

2.4.2. DNS

Daily netting. Net positions at the end of the day result into delivery. For margin and exposure purposes, net positions and square-offstaken into consideration.

2.4.3. Auction

Auctions settled on T2T basis.

2.5. Order Types

2.5.1. Market Order IOC

A market order is an order that would be matched immediately at the available best buy/best sell price for that contract. If the order could not find the match immediately (either partially/fully), the remaining quantity of the order will be cancelled by the trading system.
For example:
Given below is an illustrative order book and its details:

Client	Time	Buy Price	Buy Qty	Sell Price	Sell Qty
A	10:00:05	105	100	-	-
B	10:00:07	-	-	108	200

A Buy Market Order is placed by client C at 10:00:08 with order Qty =100.
The result is that the Market Order will be matched with the order of client B at the price of 108.

2.5.2. Limit Order (with AON and Disclosed Quantity options)

Limit Orders are those orders which will be matched only at or below the limit price in case of buy orders and at or above limit price in case of sell orders. By placing the limit order, the client is limiting the price at which his orders can be traded and also indicating to the matching engine that the order should be placed in the matching queue. For e.g, If a limit buy order is placed at Rs. 100, the client is assured that his order will not be traded at a price above 100 (the limit price). Similarly, if a limit sell order is placed at Rs. 100, the client is assured that his order is not going to be traded below rs.100 (limit price). Modification and cancelation of limit orders is allowed by the exchange. Limit orders are Good Till Session (GTS); i.e. they will be valid only for the trading session and any pending quantity at the end of the session will be cancelled by the exchange.

The following are the possible terms available along with the limit orders to the clients for placing orders:

2.5.3. All or None (AON)

If an order is placed with the quantity condition as All or None, then that would mean that either the entire quantity of the order should be traded or all of it should remain pending. The interpretation of all or none is that the entire quantity mus be matched in a signle trade as against in multiple trades at one time. Such orders cannot remain partially traded, at any point in time. Such time condition can only be specified against normal limit orders.

2.5.4. Disclosed Quantity

Orders with Disclosed Quantity are a type of limit orders, the difference being that only part of the total order quantity i.e. metered quantity is disclosed to the market. Metered Quantity is always less than the Total order quantity. Irrespective of the metered quantity, the quantity considered for matching will be the total order quantity.

2.5.5. Market Maker Quote (1-way or 2-way)

Quotes placed by market makers are called as market maker quotes. Market maker can place one way or two way quotes depending on the constraints defined for each of the contracts. Market makers are trading members with rights to participate as market makers in specific contracts as allowed and permitted by the exchange. In quote based matching contracts, orders from non market makers are always matched with quotes on the other side. Modification of market maker quotes is allowed by the exchange. All the market quotes will be only Good Till Session (GTS); i.e. they will be valid only for the trading session. Quotes may also contain zero quantity which implies that the quantity is unlimited. Zero quantity is allowed for all buy orders and sell margin sell orders.

2.5.6. OTC Orders (Market orders and Specified rate orders)

A negotiated Trade/ Order or over the counter orders are bilateral deals on the sale and purchase of a commodity. Prior to the order placement, the buying and selling parties may mutually agree on the price and the quantity before hand. If any of the order details does not

match with the counter order details, then the order will wait in matching engine till end of the session. For an OTC Contract to be matched, both the qty and the price (for specified rate order), apart from counterparty code should match. OTC orders can be modified or cancelled by the initiator before a counter order is received for the same.

There are two types of OTC orders, which will be allowed in the Trading system:

- OTC Market Order
- OTC Specified Rate Order

2.5.7. OTC Market Order

In OTC Market Order, if all the order details of the buying and the selling parties match with each other, the orders will be matched and traded at the Last Traded Price of that contract. If the same is not available, the Order may be matched at the Base Price for the order.

2.5.8. OTC Specified Rate Order

In OTC Specified Rate Order, if all the order details of the buying and the selling parties match with each other, the orders will be matched and traded at the price mentioned in the order.

2.5.9. Auction Order (English/Dutch/Yankee Forward/Reverse)

Auction orders are initiated by the buyer or the seller of the contracts. The initiator of the auction order will have to specify the bidding period start and end time, type of auction as well as the floor price/ceiling price while placing any auction order. Bids for these orders will be accepted during the respective bidding period and only after completion of the same, will the matching be done for each auction order. Cancellation of auction orders by the initiator is allowed by the exchange.

Following are the three types of auctions allowed in the trading system:

2.5.10. English Auction:

In English Auctions there will be only one actrive bid per auction order at a time. In case of forward auctions, if any new bid out-bids prevailing bid, the prevailing bid will be knocked-off. Always one of the bid gets cancelled/rejected since only one can remain in queue. In case of reverse auctions, if any new bid is placed at a price less than the prevailing bid price, the prevailing bid will be knocked-off. At the end of the bidding period the auction bid will be

matched with the auction order and a trade will be generated for the same. For English auctions, there will be only one trade against one auction order.

For example:

Auction Order (Forward Auctions) details are provided in the table below:

Client Code	Buy/Sell	Bidding Period Start Time	Bidding Period End Time	Auction Type	Floor Price	Qty
A	Sell	10:05:00 AM	11:30:00 AM	English	80	500

Auction bid details for the auction order at the end of the session for the order above has been provided below:

At the end of the session, the

Client Code	Auction Bid No	Buy/Sell	Price	Qty
B	10120900001	Buy	90	500

ME will generate a trade for the same @Rs. 90 for qty =500 as in case above.

2.5.11. Yankee Auction

In Yankee Auctions, there can be multiple auction bids for an auction order at different prices. Auction bids will be accepted as long as the total order qty of the auction order can be allocated among the different auction bids based on price, quantity and time priority.

At the end of the session bids will be matched with the auction order and trades will be generated for all the auction bids to which some qty can be allocated at the respective bid prices.

For example:

Auction Order (Forward auction) details are provided in the table below:

Client Code	Buy/Sell	Session Start Time	Session End Time	Auction Type	Floor Price	Qty
A	Sell	10:05:00 AM	11:30:00 AM	Yankee	80	500

At the end of the auction session, auction bid details for the auction order above has been provided below:

Client	Auction Bid	Buy/Sell	Price	Bid	Allocated	Traded	Cumulative	Total

Code	No			Qty	Qty	Price	Allocated Order Qty	Balance Order Qty
B	10120900001	Buy	90	100	100	90	100	400
C	10120900002	Buy	89	200	200	89	300	200
D	10120900003	Buy	88	300	200	88	500	-

At the end of the auction session, the system will match all the auction bids with the auction order such that the total order qty has been traded fully. So, at the end of the session, the following trades will be generated against the auction order:

Client	Trade Price	Trade Qty
B(Buy)	90	100
C(Buy)	89	200
D(Buy)	88	200
A(Sell)	90	100
A(Sell)	89	200
A(Sell)	88	200

2.5.12. Dutch Auction

In Dutch Auctions there can be multiple auction bid for an auction order at different prices. Auction bids will be accepted as long as the total Order Qty of the auction order can be allocated among the different auction bids based on price, quantity and time priority.

At the end of the auction session, auction bids will be matched with the auction order and trades will be generated for all the auction bids to which any Qty is allocated. The trade price for all the forward auction bids will be the lowest price to which any allocation has been done and the highest price for all the reverse auction bids.

For example:

Auction Order (Forward auction) details are provided in the table below:

Client Code	Buy/Sell	Session Start Time	Session End Time	Auction Type	Floor Price	Qty
A	Sell	10:05:00 AM	11:30:00 AM	Dutch	80	500

At the end of the auction session, auction bid details for the auction order above have been shown below:

Client Code	Auction Bid No	Buy/Sell	Price	Bid Qty	Allocated Qty	Traded Price	Cumulative Allocated Order Qty	Total Balance Order Qty
B	10120900001	Buy	90	100	100	88	100	400
C	10120900002	Buy	89	200	200	88	300	200
D	10120900003	Buy	88	300	200	88	500	-

At the end of the auction session, the system will match all the auction bids with the auction order such that the total order qty has been traded fully. So, at the end of the session, the following trades will be generated against the auction order:

Client	Trade Price	Trade Qty
B(Buy)	88	100
C(Buy)	88	200
D(Buy)	88	200
A(Sell)	88	100
A(Sell)	88	200
A(Sell)	88	200

2.5.13. Auction Bid

Auction bids are placed against an auction order. The auction bids will be accepted in the trading system only when the bidding period is in progress.

2.5.14. Auction Bids – Auto

A facility to place auto-bids will be provided in the auction matching engine. Auto bids will have a limit price. Every time a new bid comes in, and it is better than the prevailing bid, auto bid limit will be checked. If auto bid limit is better than incoming bid, a new bid one tick higher than the incoming bid will be generated. Tick size for auto bids is updated by ORS

before sending to the matching engine. RMS validates margin for auto bids at the limit price. Auto bids only support limit price, quantity is fixed.

2.6. Market Timings

2.6.1. Sessions

The market may run on a 24x7 basis.

Each Matching Engine Instance serves a group of contracts identified by ME-Instance ID. Each ME instance follows a consistent session calendar. One ME-Instance may contain multiple segments but will have only one matching engine type. Session timings for each ME-Instance may overlap with other ME-instances and may span multiple calendar dates.

Each trading Session for a matching engine instance will be identified with a unique session ID and will have series of non-overlapping start and end datetimes.

Trading sessions calendar may be changed any time by market watch before start of a trading session.

Irrespective of the sessions, for DNS purposes, calendar date of the trade will be considered for netting of trades. However, netting effects in margin calculations will be restricted to within trading session. This implies that for DNS contracts, there should not be more than one trading session per day and a trading session should not span calendar dates.

2.6.2. Settlement Calendar

Settlement calendar is defined for each commodity group and may not have any relation to the trading session nor to the market segment. Irrespective of the trading sessions, the settlement calendar would state the date for settlement of each commodity segment group in relation to the trade date range.

3. Software Environment

3.1. FIX

The Financial Information eXchange (FIX) protocol is a messaging standard developed specifically for the real-time electronic exchange of securities transactions. FIX is a public-domain specification owned and maintained by FIX Protocol, Ltd (FPL). It has been decided to use FIX 4.2 since this will speed up development and is the most widely used Fix version.

3.2. QuickFIX/J

QuickFIX/J is a full featured messaging engine for the FIX protocol. It is a 100% Java open source implementation of the popular C++ QuickFIX engine.

3.3. CAJO

CAJO is a distributed computing framework that allows any number of Java Virtual Machines to work together literally as one. Applications can evolve dynamically: from all objects being in a single Virtual Machine, to all objects being in separate Virtual Machines, to all objects being in separate physical Machines, even at runtime.

3.4. HSQLDB

HSQLDB (HyperSQL DataBase) is a leading SQL relational database engine with support for nearly full ANSI-92 SQL plus many SQL:2008 enhancements. It offers a small, fast database engine which offers in-memory and disk-based tables and supports embedded and server modes.

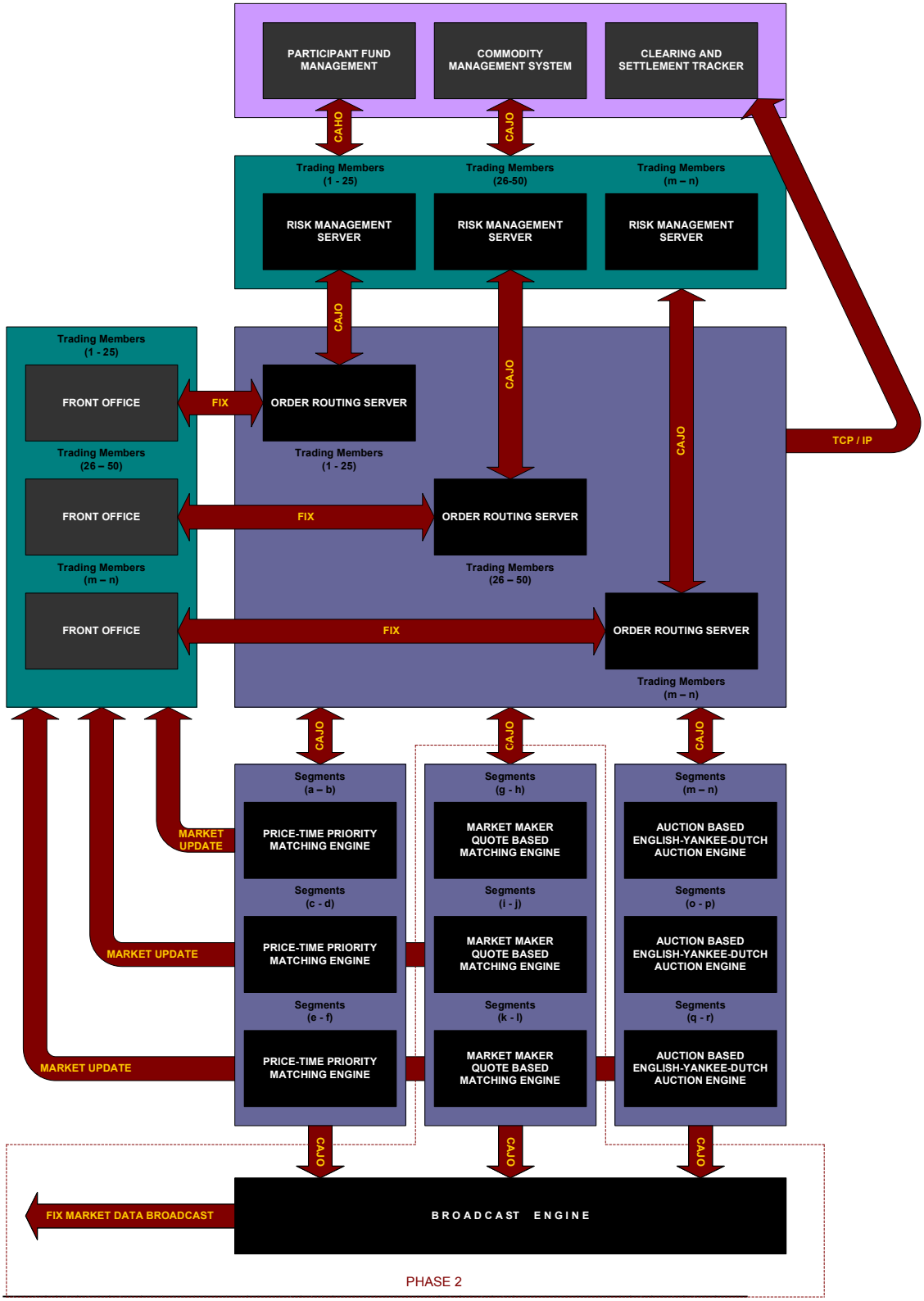
3.5. LOG4J

Log4j will be used for looging..

3.6. Log4FIX

Log4FIX is an open source FIX logger/ message viewer and based on QuickFIX/J. Log4FIX provides a pretty view for FIX messages by utilizing the QuickFIX data dictionary.

4. Architecture


PARTICIPANT FUND MANAGEMENT
COMMODITY MANAGEMENT SYSTEM
CLEARING AND SETTLEMENT TRACKER
CAHO
CAJO
Trading Members (1 - 25)
Trading Members (26-50)
Trading Members (m – n)
RISK MANAGEMENT SERVER
RISK MANAGEMENT SERVER
RISK MANAGEMENT SERVER
CAJO
CAJO
CAJO
TCP / IP
Trading Members (1 - 25)
FRONT OFFICE
FIX
ORDER ROUTING SERVER
Trading Members (1 - 25)
Trading Members (26 – 50)
FRONT OFFICE
FIX
ORDER ROUTING SERVER
Trading Members (26 – 50)
Trading Members (m – n)
FRONT OFFICE
FIX
ORDER ROUTING SERVER
Trading Members (m – n)
CAJO
CAJO
CAJO
Segments (a – b)
PRICE-TIME PRIORITY MATCHING ENGINE
Segments (c - d)
PRICE-TIME PRIORITY MATCHING ENGINE
Segments (e - f)
PRICE-TIME PRIORITY MATCHING ENGINE
Segments (g - h)
MARKET MAKER QUOTE BASED MATCHING ENGINE
Segments (i - j)
MARKET MAKER QUOTE BASED MATCHING ENGINE
Segments (k - l)
MARKET MAKER QUOTE BASED MATCHING ENGINE
Segments (m – n)
AUCTION BASED ENGLISH-YANKEE-DUTCH AUCTION ENGINE
Segments (o - p)
AUCTION BASED ENGLISH-YANKEE-DUTCH AUCTION ENGINE
Segments (q - r)
AUCTION BASED ENGLISH-YANKEE-DUTCH AUCTION ENGINE
MARKET UPDATE
MARKET UPDATE
MARKET UPDATE
CAJO
CAJO
CAJO
FIX MARKET DATA BROADCAST
BROADCAST ENGINE
PHASE 2

4.1. Order Routing

Following steps explain the mechanism of routing an order in the trading system:

FO has a TM-ORS mapping which lets it route order requests for a TM to the appropriate ORS instance.

ORS validates the incoming order request to make sure that the TM corresponding to the request is accessible on that ORS instance. This is done by checking for an entry for the TM in the in-memory TM Master. If no entry exists, retrieve the ORS Instance ID for the TM from the persistent store and check if it matches with the ORS Instance ID of the ORS on which the order was received. If there is a mismatch, reject the order, else load the TM into the TM Master and accept the order.

Store the QuickFIXJ sessionID in the in-memory TM master. This sessionID will help ORS send execution reports back to the FO.

Obtain the ORS Instance ID from the in-memory ORS map and incorporate it into host order id.

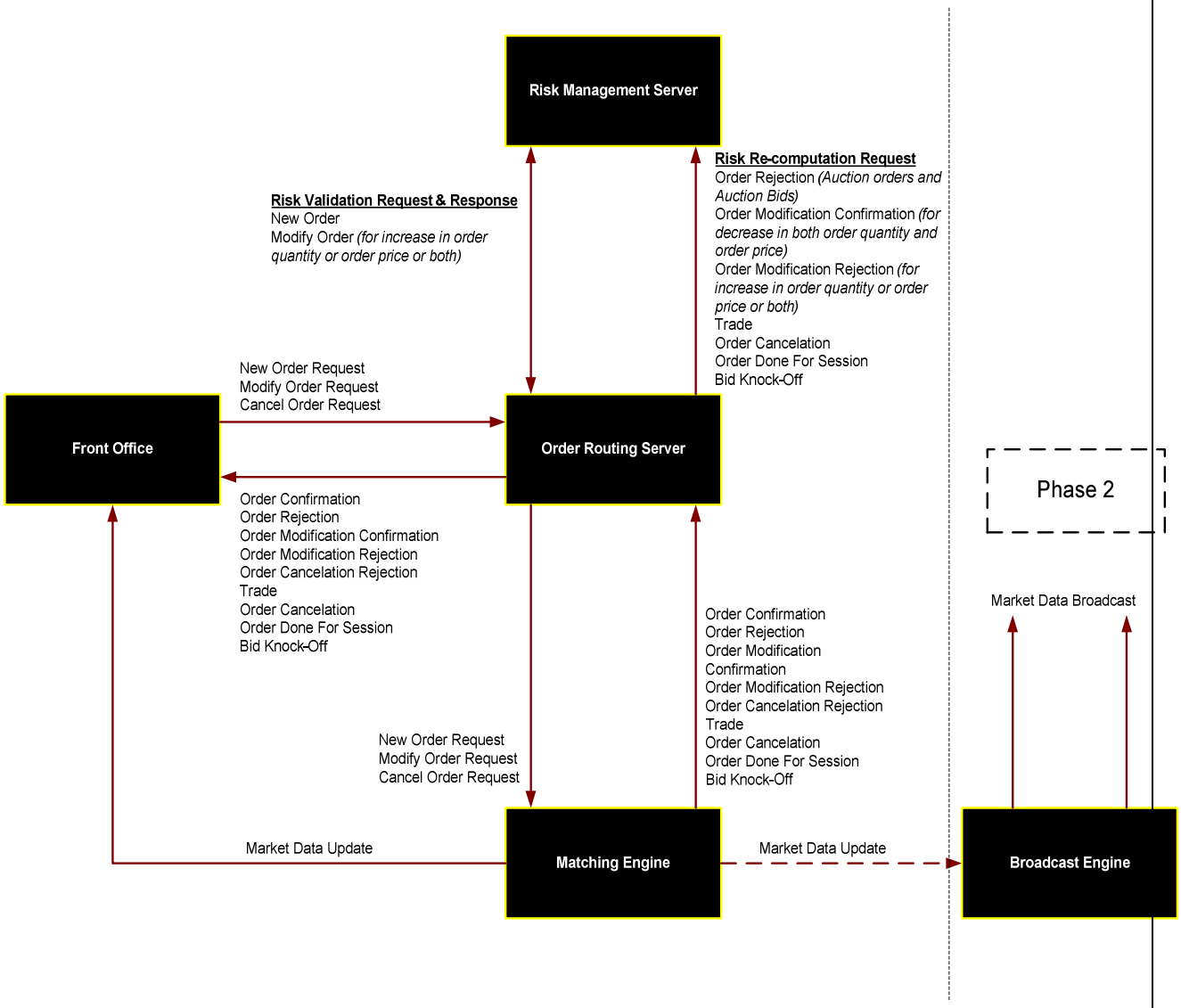
ORS checks the RMS Instance ID for the TM from the in-memory TM master. It obtains the IP address and port for that instance id from the in-memory RMS map. The order request is routed to the appropriate RMS using this information. RMS routing is in synchronous mode.

ORS checks the ME Instance ID for the contract from the in-memory Contract Master. It obtains the IP address and port for that instance id from the in-memory ME map. The order request is routed to the appropriate ME using this information. ME routing is in asynchronous mode.

ORS sends the execution report to the appropriate FO using the sessionID from the TM Master.

ORS sends the execution report to the appropriate RMS using the RMS Instance ID for the TM from the TM Master as above.

4.2. Order Flow


Risk Management Server
Risk Validation Request & Response
New Order
Modify Order (for increase in order quantity or order price or both)
Risk Re-computation Request
Order Rejection (Auction orders and Auction Bids)
Order Modification Confirmation (for decrease in both order quantity and order price)
Order Modification Rejection (for increase in order quantity or order price or both)
Trade
Order Cancelation
Order Done For Session
Bid Knock-Off
New Order Request
Modify Order Request
Cancel Order Request
Front Office
Order Routing Server
Phase 2
Order Confirmation
Order Rejection
Order Modification Confirmation
Order Modification Rejection
Order Cancelation Rejection
Trade
Order Cancelation
Order Done For Session
Bid Knock-Off
Market Data Broadcast
Order Confirmation
Order Rejection
Order Modification Confirmation
Order Modification Rejection
Order Cancelation Rejection
Trade
Order Cancelation
Order Done For Session
Bid Knock-Off
New Order Request
Modify Order Request
Cancel Order Request
Market Data Update
Matching Engine
Market Data Update
Broadcast Engine

4.3. Message Structure (QuickFIXJ Field Mappings)

For incoming and outgoing FIX messages and their mapping with internal data structures, refer to Fix 4.2 Message Structure in Appendix II.

5. Order Routing Server

5.1. Overview

Order routing server (ORS) serves as the connecting broker/router handling the communication between the various components of the exchange trading system viz. Front Office (FO), Risk Management Server (RMS) and Matching Engine (ME). Internal communication between the RMS and ME is also handled via ORS.

5.2. Functions

Loads frequently accessed data from the persistent store and configuration files, and maintains in-memory artifacts (as detailed in the section **In-Memory Artifacts**) for high-speed routing and order validation.

5.2.1. Accept FIX messages and Convert to Application Order Object

Accepts the following request types from FO:

- Login
- New Order
- Modify Order
- Cancel Order
- Market data request
- Resend Orders / Trades Request

Converts FIX order request to application order request structure

5.2.2. Validate TM and Update Session

Validates TM and manages

Checks if TM exists in local TM Map. If not attempts to load from persistent store with ORS instance ID filter. If found updates the local TM Map. If the message is a login message, creates session for the dealer. If TM Map entry does not exists, rejects the order/login message.

5.2.3. Generate Order ID

Generates unique host order id for the order for use within the exchange using ORS instance id as prefix, to enable easy routing from ME to ORS

5.2.4. Basic Order Validations

Performs basic order validations as detailed in the section "**Basic Order Validations".** On failure of basic order validations, sends order rejection message to FO.

5.2.5. Update Order Data

Adds circuit breaker upper limit price to the order in case of market order for risk validations.
Adds LTP as price field on the order in case of OTC market order; if LTP not available uses base price.

5.2.6. RMS and ME Routing

On success of validations and update of order data, puts order to RMS synchronously for risk validation and updation of risk parameters for the following request type "New Order" and request type "Modify Order".
On order risk validation failure response from RMS, sends order rejection message to FO
On order risk validation success response from RMS, puts order request asynchronously to ME. Request type "Cancel Order" is sent directly to ME without sending it to RMS.

5.2.7. Listening to ME

Listens for the following from ME:

- Order confirmation
- Order rejection (auction order and auction bids)
- Modification confirmation
- Modification Rejection
- Cancellation Rejection
- Trade
- Cancellation Confirmation
- Order expiry due to end of session. (for orders pending on the queue at the end of the trading session and auction order quantity pending on the queue at the end of auction bidding period)
- Bid knock-off

5.2.8. Order Confirmation from ME

On order confirmation from ME, sends execution report to FO, inserts the order asynchronously in the persistent store.

5.2.9. Order Rejection from ME

On order rejection from ME, sends execution report to FO, puts the rejection to RMS for re-computation of used margins and stocks.

5.2.10. Modification Confirmation from ME

On order modification confirmation from ME, sends execution report to FO, updates the order asynchronously in the persistent store, puts the confirmation to RMS for re-computation of used margins and stocks.

5.2.11. Modification Rejection from ME

On order modification rejection from ME, sends execution report to FO, puts the rejection to RMS for re-computation of used margins and blocks

5.2.12. Cancellation Rejection from ME

On order cancellation rejection from ME, sends execution report to FO.

5.2.13. Trade Message from ME

On trade message from ME, sends trade to FO, puts trade to RMS for re-computation of used margin and stock, stores LTP in contract master and persists trades in the persistent store asynchronously.

5.2.14. Cancellation Confirmation from ME

On order cancellation confirmation from ME, sends execution report to FO, updates the order asynchronously in the persistent store, puts the cancellation to RMS for re-computation of margins and stock.

5.2.15. Order Expiry Message from ME

On order expiry message from ME, sends execution report to FO, updates the order asynchronously in the persistent store

5.2.16. Bid Knock-off from ME

On bid knock-off from ME, sends execution report to FO, updates the order asynchronously in the persistent store, puts the confirmation to RMS for re-computation of risk parameters.

5.2.17. Other Functions

Maintains active user session information
Resets order volume counter for TMs after every configured interval of time
Provides API for suspending / resuming trading in a contract
Provides API for suspending / resuming trading permissions for a TM
Provides API to modify base price and circuit breaker limits for a contract.

5.3. Configuration

Following are the configuration files and parameters required for ORS:

5.3.1. ORS Configuration

ORS_INSTANCE_ID=01
ORS_IP_ADDRESS=172.12.25.3
ORS_PORT=2024
ORDER_VOLUME_CONTROL_RESET_INTERVAL=30
DB_CONN_STRING = jdbc:oracle://172.12.25.10:3333/NSPOT_DB
IN_MEM_DB_CONN_STRING = <embedded database info or connstring>
DB_USERNAME = NSPOTUSR
DB_PASSWORD = NSPOTPW
IN_MEM_DB_USERNAME = NSPOTUSR
IN_MEM_DB_PASSWORD = NSPOTPW

5.3.2. RMS Instance Info

RMS instance information is read for the first time from persistent state when required for the first time and is then kept loaded in-memory. This consists of RMS_INSTANCE_ID, RMS_IP_Address, RMS_Port.

5.3.3. ME Instance Info

ME instance information is read for the first time from persistent state when required for the first time and is then kept loaded in-memory. This consists of
ME_INSTANCE_ID, ME_IP_ADDRESS, ME_PORT, ME_Type

5.3.4. QuickFIXJ Configuration

(sample QuickFIXJ configuration file)

[default]

FileStorePath=data/exchange

ConnectionType=acceptor

StartTime=00:00:00

EndTime=00:00:00

HeartBtInt=600

ValidOrderTypes=1,2 ?? needs clarification

SenderCompID=NSPOT_EXCHANGE

TargetCompID=NSPOT_FO Should we have this here? This can be determined dynamically.

UseDataDictionary=Y

ResetOnLogon=N

[session]

BeginString=FIX.4.2

SocketAcceptPort=9881

5.4. In-memory artifacts

5.4.1. ORS Map

ORS Map is loaded from ORS.conf on startup and consists of the following information:

- ORS Instance ID
- IP Address
- Port
- Order Request Volume Monitoring Interval

5.4.2. RMS Map

RMS Map is loaded from persistent store when required and then maintained in memory and consists of the following information:

- RMS Instance ID
- IP Address and Port

5.4.3. Matching Engine MAP

This is loaded from loaded from persistent store when required and then maintained in memory and consists of the following information:

- ME Instance ID
- IP Address and Port
-

5.4.4. Matching Type-Order Type Map

Matching Type to Order Type Mapping is loaded from persistent store on startup and consists of the following information:

- Matching Type
- Order Types List (comma-separated)

5.4.5. Order Type-Request Type Map

Order Type to Request Type Mapping is loaded from persistent store on startup and consists of the following information:

- Order Type
- Request Types List (comma-separated)

5.4.6. Contract Master

Contract Master is fully loaded on startup from persistent store and consists of the following information:

- Contract Code
- Segment Code
- Base Price
- Circuit Breaker Upper Limit
- Circuit Breaker Lower Limit
- LTP
- Trading Lot
- Minimum qty
- Limit Quantity (value of 0 indicates no limit)
- Price Conversion Factor

- Tick Size
- Matching Type (PTP, MMQ, AUC)
- IsTradingOn Flag (set to true on receiving trading session start notification and set to false on receiving trading session end notification)
- isFrozen Flag
- ME Instance ID

5.4.7. TM Master

TM Master template structure is loaded on startup and is populated on demand from the persistent store. For every incoming order, a check is made to see if the TM is already present in the TM Master. If no, it is loaded into the TM Master from persistent store. It contains the following information for each TM:

- TM ID
- Max order requests (per t seconds)
- Order request counter (to be reset every t seconds)
- RMS Instance ID
- Segment List (comma-separated)
- QuickFIXJ Session ID
- isSuspended Flag

5.4.8. User Session

User Session template structure is loaded on startup and is populated as and when login requests come in.

- TM ID
- Dealer ID
- QuickFixJ SessionID
- Start Time
- Last Updated Time
- TargetCompID

5.5. Startup Process

Sets up an executor service to handle concurrent execution of tasks

Sets up a scheduled executor service to handle scheduled tasks

Loads the in-memory artifacts as detailed in the section **In-memory artifacts**.
Sets up a listener for receiving order request confirmation/rejection/cancelation responses and trades from ME, trading session start/end notifications from ME and administrative notifications/updates from the exchange administrator
Sets up an order counter reset timer using the scheduled executor service
Sets up a query processor to handle user session related queries from the administrative interface
Sets up a FIX interface in acceptor mode to receive incoming FIX request

5.6. Trading Session Startup Process

Initiated on receiving the trading session start notification from ME
Sets the isTradingOn flag to *true* for all contracts associated with the ME instance id. This would be used for validating the incoming orders to check whether the trading session is in progress for the contract corresponding to the order.

5.7. Trading Session Shutdown Processs

Initiated on receiving the trading session end notification from ME
Sets the isTradingOn flag to *false* for all contracts associated with the ME instance id. Also informs all loaded RMS instances of trading session shutdown notification.

5.8. Shutdown Process

Send logout notification to all active FO sessions
Stop accepting order requests
Shutdown the executor service
Shutdown the scheduled executor service

5.9. Order Validations

Following are the order validations in sequence as performed by the ORS for incoming orders:

5.9.1. Contract and Trading Session Validation

Order must be for a contract for which the trading session is in progress as can be validated using the isTradingOn flag on the contract from the contract master.

Order must be for a contract which is not under suspension as can be validated using the isSuspended flag on the contract from the contract master.
If validation fails, order is rejected.
While validating contract ORS reads the segment for the contract.

5.9.2. TM and Segment Validation

TM must be present in TMMap, must not be suspended and must be allowed to operate on the contract segment.
If TM is not found TMMap, an attempt is made to load TM from persistent store with current ORS instance ID as the filter. If found, it is loaded in TMMap and then the above validations are done.
If validation fails, order is rejected.

5.9.3. Request and Order Type

Request type and order types must be compatible as per Request-Order Type Map. If not, order is rejected.

5.9.4. Volume Control

Order request volume must not exceed the volume limit allowed for the TM. If limit is exceeded, order is rejected. This validation is done while creating the host order number.

5.9.5. Mandatory Fields

Order must have the mandatory fields populated as applicable to the specific order type which can be validated as per the section **Message Structure.**

5.9.6. Matching Type and Order Type

Order type must be valid for the particular contract as can validated by obtaining the matching type for that contract from the contract master and checking for an entry for that order type in the Matching Type to Order Type mapping.

5.9.7. Quantity Validation

Order quantity must be at a minimum equal to the minimum quantity and in multiples of trading lot size and must not exceed the maximum quantity as can be validated from the minimum quantity and maximum quantity fields in the contract master for that contract.

For Partially Disclosed orders, disclosed quantity must be at a minimum equal to the minimum quantity and in multiples of trading lot size and must be less than the order quantity.

5.9.8. Price Validation

Order price must be a multiple of the tick size and must be between the circuit breaker limits as can be validated using the tick size, circuit breaker upper limit and circuit breaker lower limit fields in the contract master for that contract.

5.10. Order Number Generation Logic

The order number is a 15 character string structured as shown below and is guaranteed to be unique for a period of one year. The assumption here is that the order rate per ORS instance will not exceed 9999 per second.

The initial 2 digits will be the ORS Instance ID of the ORS which receives the order and will range from 01-99.

The next 3 digit will be the day of the year which will range from 001-366.

The next 6 digits will be the time stamp when the order is received by the ORS. The format for the same would be hhmmss and can range from 000000 to 235959.

The last 4 digits will be the sequence no which will range from 0001-9999 assuming not more than 9999 orders per second. When the counter reaches 9999, it is reset to 1.

9	9	3	6	5	2	3	5	9	5	9	9	9	9	9
ORS Instance Id		Day of year			Time in hh:mm:ss format						Sequence No			
(01-99)		(001-365)			(00:00:00 – 23:59:59)						(0001 – 9999)			

5.11. New Order Request

ORS performs the following activities for a new order request:

Perform basic order validations as detailed in the section **Basic Order Validations**. If basic validations are successful, the ORS would send the order to RMS for RMS validation. If RMS validations are successful, ORS would send the order to the ME for Matching. If basic order validations fail, then an order rejection message will be sent to the FO by the ORS. IF RMS validations fail, then an order rejection message will be sent to the FO by the ORS.

5.12. Modification Order Request

ORS performs the following activities for a modification order request:

The ORS would validate, if modification is allowed for the order type as per the table below:

	Limit Order	Market Order	OTC Specified Rate Order	OTC Market Order	Auction Order	Auction Bids
Modification Request Allowed	Y	N	Y	Y	N	N

If allowed, the ORS would only validate the order quantity, order price and metered quantity (for limit Order) as per the basic order validations. If the modified order quantity < previous quantity and price < previous price, the ORS would put the order to ME. If Modified order quantity > previous order quantity, or price > previous price the ORS would put the order to RMS for risk validation. If RM validations are successful, the ORS would put the order to the ME for Matching. If basic order validations fail, then an order rejection message will be sent to the FO by the ORS. If RM validations fail, then an order rejection message will be sent to the FO by the ORS.

5.13. Order Cancellation Request

ORS performs the following activities for a cancelation order request:

The ORS would validate if cancelation is allowed for the specific order type as per the table below:

	Limit Order	Market Order	OTC Specified Rate Order	OTC Market Order	Auction Order	Auction Bids
Cancelation Request Allowed	Y	N	Y	Y	Y	N

If cancelation would be allowed, then the ORS would send the order to the ME. On receiving order cancelation requests from the ORS, the ME would either send an order cancelation execution or order cancelation rejection message to the ORS.

On receiving order cancelation from ME, the ORS would send the same to the RM for re-computation of margins, stocks, and exposure limits and would also asynchronously update the order status in the persistent database.

For both order cancelation and order cancelation rejection, the ORS would send the same to FO.

5.14. Order Cancellation by the ME

If the Market order could not be traded fully, the remaining quantity will be canceled by ME. ME would send the order cancelation message to ORS. ORS would send the same to the RMS for re-computation of blocked margins, exposures and stocks. The ORS would also send the message to the FO intimating the same. The ORS would also asynchronously update the order status in the persistent database.

5.15. Bid knock-off by ME

For auction bids knocked off by ME, it would send the bid knock-off message to ORS.

ORS would send the same to the RMS for re-computation of blocked margins, exposures and stocks. The ORS would also send the message to the FO intimating the same. The ORS would also asynchronously update the order status in the persistent database.

5.16. Order Expiry from ME

For all orders pending on the queue at the end of the session and for auction order quantity pending on the queue at the end of bidding period, the ME will send order done for session message to ORS.

ORS would send the same to the RMS for re-computation of blocked margins, exposures and stocks in case of auction orders. The ORS would also send the message to the FO intimating the same.

The ORS would also asynchronously update the order status in the persistent database.

5.17. ORS Implementation

5.17.1. Order Routing Server

Order Routing Server is the main class that controls the overall life-cycle of the order routing server and is responsible for setting up, operating and shutting down of the various components of the order routing server. During the operating phase, it sets itself up in QuickFIXJ acceptor mode for receiving incoming FIX order requests.

5.17.2. Executor Service

An Executor Service for concurrent execution of tasks is set up at start-up by OrderRoutingServer.

5.17.3. Scheduled Executor Service

A Scheduled Executor Service for handling scheduled tasks is set up at start-up by OrderRoutingServer.

5.17.4. Exchange FIX Interface

ExchFIXIntf is implemented using the QuickFIXJ FIX engine and is responsible for processing the session messages (logon, heartbeat, logout) as well as the application messages (new order, order modification, order cancelation) coming from FO. ExchFIXIntf submits an OrdReqProc thread to the executor service to handle each order request.

5.17.5. Order Request Processor

OrdReqProc performs the basic order validations, puts the order request to RMS for risk validation as applicable, sends order rejection response to FO in case of validation failure, puts the order request to ME in case of validation success.

5.17.6. ORS Listener

ORSListener sets itself up as a CAJO server to listen to order request confirmation/rejection/cancelation responses and trades from ME, trading session start/end notifications from ME and administrative notifications/updates from administrative interface. ORSListener submits an OrdRespProc task to the executor service for handling each order request confirmation/rejection/cancelation response from ME.

5.17.7. Execution Response Processor

Execution Response Processor sends the execution report corresponding to the response to FO, forwards the response to RMS for risk re-computation as applicable, submits a Persistor

task to the executor service as applicable to update the order asynchronously. ORSListener submits a TradeProc task to the executor service for handling each trade from ME.

5.17.8. Trade Processor

Trade Processor sends the execution report corresponding to the trade to FO, forwards the trade to RMS for risk re-computation and submits a Persistor task to the executor service to persist the trade asynchronously.

5.17.9. Persistor

Persistor task puts orders and trades in a persistent store as required asynchronously.

5.17.10. Exchange Admin Process

ExchAdminProc updates the *isTradingOn* flag in the contract master and provies an API for real time administrative updates and session queires.

5.17.11. Exchange Fix Adapter

ExchFIXAdapter provides conversion from FIX order request to internal order request format and from internal execution response format to FIX execution report.

5.17.12. Volume Control Timer

Order Volume Ccontrol Reset Timer is set up using the scheduled executor service and resets the order counter for the TMs for every *t* seconds as configured in the ORS configuration file and rejects orders for a TM when his order request volume limit for the *t* seconds is crossed.

6. Risk Management Server

6.1. Overview

Risk Management Server (RMS) serves risk validation requests from the ORS for the order requests that are placed on the exchange. Depending on the order side and contract type (stock/margin), RMS validates a combination of the risk parameters – exposure, margin, stock. On risk validation success, RMS blocks the required exposure, margin, stock as applicable. On receiving order request confirmation/rejection/cancelation and trades from the ORS, RMS does a re-computation of these parameters as applicable.

6.2. Risk Parameters and Definitions

6.2.1. Margin Limit

All new order requests (except for stock check sell orders) and modification requests, will be validated against margin limit. Margin limit is defined at the TM level. Blocked margin is calculated Trading account and contract wise and then summed up for validation at the TM level. If margin limit is likely to be exceeded due to the incoming order, then the same will be rejected. For order confirmation/rejection/cancellation and trade responses from the ORS, blocked Margin will be re-calculated taking into consideration the effect of the response. PFM provides margin limit updates (TM wise) to the RMS. At the end of the trading session, RMS updates (flushes) the blocked margin at trade level to the PFM for the contracts associated with the ME id.

6.2.2. Stock Limit

All new and modification sell order requests for stock check contracts will be validated against stock limit. Stock limit is defined and validated at trading account level, for each contract. If stock limit is likely to be exceeded due to the incoming order, then the same will be rejected. For order request confirmation/rejection/cancellation and trade responses from the ORS, the blocked stock will be re-calculated taking into consideration the effect of the response. CMS provides stock limit updates (TAC wise) to the RMS. At the end of the trading session, RMS updates (flushes) the blocked stock at trade level to the CMS for the contracts associated with the ME id.

6.2.3. Exposure limit

All new and modification order requests except for the trading accounts and dealers with max exposure limit as zero, will be validated against the exposure limit. Exposure is defined at the TAC and dealer level. Exposure is calculated Trading account wise and contract wise and then summed up across all the contracts for validation at the trading account level and at dealer level. If exposure limit is likely to be exceeded due to the incoming order, then the order will be rejected. For order request confirmation/rejection/cancellation and trade responses from the ORS, used exposure will be re-calculated taking into consideration the effect of the response. End of session process will recalculate exposure used.
Exposure limit is read from TAC master and user master and may need to be updated in real time using the administrative interface.

6.3. Functions

6.3.1. Maintaining Limits

RMS maintains margin limits and used margins at TM level, stock limit and blocked stock at Trading Account plus Contract level and Exposure limit and used exposure at Trading Account as well as Dealer level.
The limit data is loaded from persistent store at start-up for TM level data and as required for Trading Account Level data into in-memory artifacts for quick risk computations and validations.

6.3.2. Setting up Listeners

Sets up and starts risk validation request listener, update listener and PFM /CMS interaction listeners. Risk validation

6.3.3. Risk Validation

For all new orders and order modifications, margin exposure and stock checks are carried out by RMS depending on the contract and settlement type.
RMS ensures that a new order or order modification is rejected if it is likely to result in breaching of stock limit, exposure limit at trading account level or dealer level or margin limit at TM level.
RMS updates blocked stock, used exposure and blocked margin when an order is valdiated.
The computation formulae for risk parameter updates are defined in sections 6.3.4 to 6.3.6.
Detailed algorithms for risk validation are defined in sections 7.1 and 7.2

6.3.4. Risk Computations - Margin

Margin availability is checked at TM level. TM level Blocked Margin is sum of trading account level Blocked Margin for Trading Accounts for a TM. Trading Account Level Blocked Margin is sum of Blocked Margin for order and trade position for each trading account for each contract.

Formula to compute blocked margin for each order and trade position is

For DNS type settlement contracts –

Blocked Margin = A+B+E

Where

A= MAX (C, D)

C= [Buy Order Value – (if NetQty < 0, ABS(NetValue),0)] *BuyMargin %

D= [Sell Order Value – (if NetQty >0, ABS(NetVal),0)] * SellMargin %

B= (If netQty<>0, ABS [NetVal* (If NetVal < 0, SellMargin %, BuyMargin %)],0)

E=(If NetQty =0 and NetVal >0, NetVal, 0)

Where

NetQty = Buy Trade Quantity – Sell Trade Quantity and

NetVal = Buy Trade Value – Sell Trade Value

For T2T type settlement contracts

P+Q

Where

P=(Buy Order Valu +Buy Trade Value) *BuyMargin %

Q= (Sell Order Value+Sell Trade Value] * SellMargin %

6.3.5. Risk Computations - Exposure

Exposure is checked at Dealer and Trading Account levels. Dealer level Used Exposure is sum of Used Exposure at Trading Account level. Trading Account Level Used Exposure is sum of Used Exposure for order and trade position for each trading account for each contract.

Used exposure for trading account position for each contract is calculated as follows.

For DNS type settlement contracts

Used Exposure = ABS (Buy Order Value +Buy Trade Value – Sell Order Value –Sell Trade Value)

For T2T type settlement contracts

Used Exposure = ABS (Buy Order Value +Buy Trade Value + Sell Order Value + Sell Trade Value)

6.3.6. Risk Computations - Stock

Stock check is carried out for only sell orders of stock check type contracts. Stock is checked for order and trade position for each trading account for each contract.

Blocked stock for trading account position for each contract is calculated as follows.

For DNS type settlement contracts

Blocked Stock = Sell Order Quantity – (Buy Trade Quanity – Sell Trade Quantity)

For T2T type settlement contracts

Blocked Stock = Sell Order Quantity +Sell Trade Quantity

6.3.7. Risk Parameter Updates

RMS updates risk parameters i.e. blocked stock, used exposure and blocked margin based on acceptance of order modifications, acceptance of order cancellations and trades. The computation formulae for risk parameter updates are the same as defined in section 6.3.4 to 6.3.6. Detailed algorithms for risk validation are defined in sections 7.3 to 7.7

6.3.8. Real Time Limit Updates

RMS supports real time updates to margin limits and exposure limits by providing an API to PFM for this purpose. It also supports real time updates to stock limits by providing an API to CMS for this purpose. Detailed real time update algorithms are handled in section 7.9.

6.3.9. Other Functions

Provides API forquery and modification of exposure limit at Trading Account and Dealer level.

Provides API to query margin information TM wise

Provides API for querying position information trading account wise

6.4. Configuration

RMS.conf

[RMS]

RMS_INSTANCE_ID=01

RMS_IP_ADDRESS=172.12.25.4 RMS_PORT=2024

DB_CONN_STRING = jdbc:oracle://172.12.25.10:3333/NSPOT_DB

IN_MEM_DB_CONN_STRING = <embedded database info>
DB_USERNAME = NSPOTUSR
DB_PASSWORD = NSPOTPW
IN_MEM_DB_USERNAME = NSPOTUSR
IN_MEM_DB_PASSWORD = NSPOTPW

6.5. In-memory artifacts

6.5.1. RMS Map

RMS Map is loaded from RMS.conf on startup and consists of the following information:

- RMS Instance ID
- IP Address and Port

6.5.2. Contract Map

Contract Map is loaded from the persistent store on startup and consists of the following information:

- Contract Code
- Sale Margin/Stock Check
- DNS/T2T
- Buy Margin %
- Sell Margin %
- Buy Quote % (For MM)
- Sell Quote % (For MM)
- Spread Limit (For MM)

6.5.3. Trading Member – Margin Map (TMMARGIN)

This structure is loaded from the persistent store on startup and populated on demand from the persistent store when an order request for that TM comes in. It consists of the following information:

- Trading Member ID
- Margin Limit
- Blocked Margin

6.5.4. Trading Account – Exposure Map (TACEXP)

This structure is loaded from the persistent store on startup and populated on demand from the persistent store when an order request for that Trading Account comes in. It consists of the following information:

- Trading Account ID
- TM ID
- Exposure Limit
- Exposure Used

6.5.5. Trading Account Positions Map (TACPOS)

This structure is loaded on startup and populated on-demand from the persistent store when an order request for the Trading Account comes in. It consists of the following information:

- Trading Member ID
- Trading Account ID
- Contract Code
- DNS/T2T
- Trade Buy Quantity
- Trade Buy Value
- Trade Sell Quantity
- Trade Sell Value
- Trade Net Quantity (will not be used for T2T contracts)
- Trade Net Value (will not be used for T2T contracts)
- Order Buy Quantity
- Order Buy Value
- Order Sell Quantity
- Order Sell Value
- Exposure Used
- Blocked Margin
- Blocked Stock
- Stock Limit (needs to be obtained in real time from CMS when the record is loaded)

6.5.6. Dealer Exposure Map: (DEXP)

This structure is loaded as needed and maintains dealer wise exposure limit and exposure used.

- Trading Member ID
- Dealer ID
- Exposure Limit
- Used Exposure

6.6. Startup Process

Sets up an executor service to handle concurrent execution of tasks

RMS loads the in-memory artifacts as detailed in the section **In-memory artifacts** above.

Sets up a listener for handling risk validation requests and risk re-computation requests, from ORS, margin updates from PFM, stock updates from CMS, trading session start/end notifications from ME.

Sets up a query processor to handle exposure, margin, stock, position related queries from the administrative interface

6.7. End of Trading Session process

RMS is alerted on trading session ending for a given ME ID and Session ID by ORS.

RMS writes to persistent tables TACPOS records for all contracts with the given ME-ID on as is basis. It also writes TACEXP, DEALEREXP and TMMargin records to persistent tables.

It then calls the PFM and initiates margin end of session process in PFM.

It then calls end of session process in CMS for each TAC and contract and sends the blocked stock update.

After completing the blocked stock update, it deletes the TACPOS records for contracts for which the session has ended.

It then recalculates margins and exposures for remaining TACPOS records and updates TAXEXP, DEALEREXP and TMMargin, blocked margin records as per section 7.8.

It then receives session blocked margin amount from PFM and reduces the margin limit by session blocked margin amount.

6.8. Shutdown Process

RMS process will shut down only if there blocked stocks in TACPosition records are zero and Blocked Margin in TM records is zero.

7. Algorithms for Risk Validation and Updates

7.1. New Orders including Auction Orders.

7.1.1. New DNS Buy Orders

Read TACPOS; UsedExposure and BlockedMargin

Set as OldUsedExposure and OldBlockedMargin

Update BOV with BOV+OV and BOQ with BOQ+OQ

Calculate NewUsedExposure as

ABS (BOV+BTV – SOV–STV)

Calculate NewBlockedMargin as

A+B+E

Where

A= MAX (C, D)

C= [BOV – (if NetQty < 0, ABS(NetValue),0)] *BuyMargin %

D= [SOV – (if NetQty >0, ABS(NetVal),0)] * SellMargin %

B= ABS [NetVal* (If NetVal < 0, SellMargin %, BuyMargin %)]

E=(If NetQty =0 and NetVal >0, NetVal, 0)

Update TACPos set UsedExp=NewUsedExp and BlockedMargin=NewBlockedMargin

Calculate

DeltaExp = NewUsedExp – OldUsedExp

DeltaMargin=NewBlockedMargin – OldBlockedMargin

Go to TACEXP Record and DEXP (DealerExp) record

If (TACExp ExpLimit<>0 and ExpLimit – UsedExp < DeltaExp) OR (DEXP ExpLimit<>0 and ExpLimit – UsedExp < DeltaExp)

Don't update TACEXP, DEXP

Update TACPOS (*revert changes to TACPOS*), set

UsedExp=UsedExp – DeltaExp

BlockedMargin=Blocked Margin – DeltaMargin

BOV=BOV–OV

BOQ=BOQ–OQ

Reject Order

Exit

Else

Update TACEXP set

UsedExp=UsedExp + DeltaExp

BlockedMargin = BlockedMargin + DeltaMargin

Update DEXP set

UsedExp=UsedExp + DeltaExp

End if

Go to TM

If MarginLimit – MarginBlocked < Delta Margin

Don't update TM

Update TACEXP (*revert changes to TACEXP*)

Set UsedExp = UsedExp – DeltaExp

BlockedMargin = BlockedMargin - DeltaMargin

Update DEXP (*revert changes to DEXP*)

Set UsedExp = UsedExp – DeltaExp

Update TACPOS (*revert changes to TACPOS*), set

UsedExp=UsedExp – DeltaExp

BlockedMargin=Blocked Margin – DeltaMargin

BOV=BOV–OV

BOQ=BOQ–OQ

Reject Order

Exit

Else

Update TM, set

Blocked Margin = Blocked Margin + DeltaMargin

Validate Order

Exit

End if

7.1.2. New DNS Sell Order Margin Contracts

Read TACPOS; UsedExposure and BlockedMargin

Set as OldUsedExposure and OldBlockedMargin

Update SOV with SOV+OV and SOQ with SOQ+OQ

Calculate NewUsedExposure as

ABS (BOV+BTV – SOV–STV)

Calculate NewBlockedMargin as

A+B+E

Where

A= MAX (C, D)

C= [BOV – (if NetQty < 0, ABS(NetValue),0)] *BuyMargin %

D= [SOV – (if NetQty >0, ABS(NetVal),0)] * SellMargin %

B= ABS [NetVal* (If NetVal < 0, SellMargin %, BuyMargin %)]

E=(If NetQty =0 and NetVal >0, NetVal, 0)

Update TACPos set UsedExp=NewUsedExp and BlockedMargin=NewBlockedMargin

Calculate

DeltaExp = NewUsedExp – OldUsedExp

DeltaMargin=NewBlockedMargin – OldBlockedMargin

Go to TACEXP Record

If (TACExp ExpLimit<>0 and ExpLimit – UsedExp < DeltaExp) OR (DExp ExpLimit<>0 and ExpLimit – UsedExp < DeltaExp)

Don't update TACEXP, DEXP

Update TACPOS (*revert changes to TACPOS*), set

UsedExp=UsedExp – DeltaExp

BlockedMargin=Blocked Margin – DeltaMargin

SOV=SOV–OV

SOQ=SOQ–OQ

Reject Order

Exit

Else

Update TACEXP set

UsedExp=UsedExp + DeltaExp

BlockedMargin = BlockedMargin + DeltaMargin

Update DEXP set

UsedExp=UsedExp + DeltaExp

End if

Go to TM

If MarginLimit – MarginBlocked < Delta Margin

Don't update TM

Update TACEXP (*revert changes to TACEXP*)

Set UsedExp = UsedExp – DeltaExp

BlockedMargin = BlockedMargin - DeltaMargin

Update DEXP (*revert changes to DEXP*)

Set UsedExp = UsedExp – DeltaExp

Update TACPOS (*revert changes to TACPOS*), set

UsedExp=UsedExp – DeltaExp

BlockedMargin=Blocked Margin – DeltaMargin

SOV=SOV–OV

SOQ=SOQ–OQ

Reject Order

Exit

Else

Update TM, set

Blocked Margin = Blocked Margin + DeltaMargin

Validate Order

Exit

End if

7.1.3. New DNS Sell Orders for Stock Contracts

Read TACPos, UsedExposure and BlockedStock

Set as OldUsedExposure, OldBlockedStock

If OQ > StockLimit – BlockedStock

Reject order

Exit

Else

Update SOV with SOV+OV and SOQ with SOQ+OQ

Calculate NewUsedExposure as

ABS (BOV+BTV – SOV–STV)

Calculate NewBlockedStock as

SOQ–NETQTY

Update TACPos set UsedExp=NewUsedExp and BlockedStock=NewBlockedStock

Calculate

DeltaExp = NewUsedExp – OldUsedExp

DeltaStock = NewBlockedStock – OldBlockedStock

Go to TACEXP Record

If (TACExp ExpLimit<>0 and ExpLimit – UsedExp < DeltaExp) OR (TACExp ExpLimit<>0 and ExpLimit – UsedExp < DeltaExp)

Don't update TACEXP, DEXP

Update TACPOS (*revert changes to TACPOS*), set

UsedExp=UsedExp – DeltaExp

BlockedStock=BlockedStock – DeltaStock

SOV=SOV–OV

SOQ=SOQ–OQ

Reject Order

Exit

Else

Update TACEXP set

UsedExp=UsedExp + DeltaExp

Update DEXP set

UsedExp=UsedExp + DeltaExp

Validate Order

Exit

End if

7.1.4. New T2T Buy Orders

Read TACPos, UsedExposure and BlockedMargin

Set as OldUsedExposure and OldBlockedMargin

Update BOV with BOV+OV and BOQ with BOQ+OQ

Calculate NewUsedExposure as

ABS (BOV+BTV + SOV+STV)

Calculate NewBlockedMargin as

P+Q

Where

P=(BOV+BTV) *BuyMargin %

Q= (SOV+STV] * SellMargin %

Update TACPos set UsedExp=NewUsedExp and BlockedMargin=NewBlockedMargin

Calculate

DeltaExp = NewUsedExp – OldUsedExp

DeltaMargin=NewBlockedMargin – OldBlockedMargin

Go to TACEXP Record

If (TACExp ExpLimit<>0 and ExpLimit – UsedExp < DeltaExp) OR (DExp ExpLimit<>0 and ExpLimit – UsedExp < DeltaExp)

Don't update TACEXP, DEXP

Update TACPOS (*revert changes to TACPOS*), set

UsedExp=UsedExp – DeltaExp

BlockedMargin=Blocked Margin – DeltaMargin

BOV=BOV–OV

BOQ=BOQ–OQ

Reject Order

Exit

Else

Update TACEXP set

UsedExp=UsedExp + DeltaExp

BlockedMargin = BlockedMargin + DeltaMargin

Update DEXP set

UsedExp=UsedExp + DeltaExp

End if

Go to TM

If MarginLimit – MarginBlocked < Delta Margin

Don't update TM

Update TACEXP (*revert changes to TACEXP*)

Set UsedExp = UsedExp – DeltaExp

Set BlockedMargin=BlockedMargin -DeltaMargin

Update DEXP (*revert changes to DEXP*)

Set UsedExp = UsedExp – DeltaExp

Update TACPOS (*revert changes to TACPOS*), set

UsedExp=UsedExp – DeltaExp

BlockedMargin=Blocked Margin – DeltaMargin

BOV=BOV–OV

BOQ=BOQ–OQ

Reject Order

Exit

Else

Update TM, set

Blocked Margin = Blocked Margin + DeltaMargin

Validate Order

Exit

End if

7.1.5. New T2T Sell Order Margin Contracts

Read TACPos, UsedExposure and BlockedMargin

Set as OldUsedExposure and OldBlockedMargin

Update SOV with SOV+OV and SOQ with SOQ+OQ

Calculate NewUsedExposure as

ABS (BOV+BTV + SOV+STV)

Calculate NewBlockedMargin as

P+Q

Where

P=(BOV+BTV) *BuyMargin %

Q= (SOV+STV] * SellMargin %

Update TACPos set UsedExp=NewUsedExp and BlockedMargin=NewBlockedMargin

Calculate

DeltaExp = NewUsedExp – OldUsedExp

DeltaMargin=NewBlockedMargin – OldBlockedMargin

Go to TACEXP Record

If (TACExp ExpLimit<>0 and ExpLimit – UsedExp < DeltaExp) OR (DExp ExpLimit<>0 and ExpLimit – UsedExp < DeltaExp)

Don't update TACEXP, DEXP

Update TACPOS (*revert changes to TACPOS*), set

UsedExp=UsedExp – DeltaExp

BlockedMargin=Blocked Margin – DeltaMargin

SOV=SOV–OV

SOQ=SOQ–OQ

Reject Order

Exit

Else

Update TACEXP set

UsedExp=UsedExp + DeltaExp

BlockedMargin = BlockedMargin + DeltaMargin

Update DEXP set

UsedExp=UsedExp + DeltaExp

End if

Go to TM

If MarginLimit – MarginBlocked < Delta Margin

Don't update TM

Update TACEXP (*revert changes to TACEXP*)

Set UsedExp = UsedExp – DeltaExp

BlockedMargin = BlockedMargin - DeltaMargin

Update DEXP (*revert changes to DEXP*)

Set UsedExp = UsedExp – DeltaExp

Update TACPOS (*revert changes to TACPOS*), set

UsedExp=UsedExp – DeltaExp

BlockedMargin=Blocked Margin – DeltaMargin

SOV=SOV–OV

SOQ=SOQ–OQ

Reject Order

Exit

Else

Update TM, set

Blocked Margin = Blocked Margin + DeltaMargin

Validate Order

Exit

End if

7.1.6. New T2T Sell Orders for Stock Contracts

Read TACPos, UsedExposure and BlockedStock

Set as OldUsedExposure, OldBlockedStock

If OQ > StockLimit – BlockedStock

Reject order

Exit

Else

Update SOV with SOV+OV and SOQ with SOQ+OQ

Calculate NewUsedExposure as

ABS (BOV+BTV – SOV–STV)

Calculate NewBlockedStock as

SOQ+STQ

Update TACPos set UsedExp=NewUsedExp and BlockedStock=NewBlockedStock

Calculate

DeltaExp = NewUsedExp – OldUsedExp

DeltaStock = NewBlockedStock – OldBlockedStock

Go to TACEXP Record

If (TACExp ExpLimit<>0 and ExpLimit – UsedExp < DeltaExp) OR (DExp ExpLimit<>0 and ExpLimit – UsedExp < DeltaExp)

Don't update TACEXP, DEXP

Update TACPOS (*revert changes to TACPOS*), set

UsedExp=UsedExp – DeltaExp

BlockedStock=BlockedStock – DeltaStock

SOV=SOV–OV

SOQ=SOQ–OQ

Reject Order

Exit

Else

Update TACEXP set

UsedExp=UsedExp + DeltaExp

Update DEXP set

UsedExp=Used Exp + DeltaExp

Validate Order

Exit

End if

7.2. Modified Orders

7.2.1. Modification DNS Buy Orders

Read TACPos, UsedExposure and BlockedMargin

Set as OldUsedExposure and OldBlockedMargin

If OLDOV >= OV

Validate Order

Exit

Else

Update BOV with BOV+(OV–OLDOV) and BOQ with BOQ+(OQ–OLDOQ)

Calculate NewUsedExposure as

ABS (BOV+BTV – SOV–STV)

Calculate NewBlockedMargin as

A+B+E

Where

A= MAX (C, D)

C= [BOV – (if NetQty < 0, ABS(NetValue),0)] *BuyMargin %

D= [SOV – (if NetQty >0, ABS(NetVal),0)] * SellMargin %

B= ABS [NetVal* (If NetVal < 0, SellMargin %, BuyMargin %)]

E=(If NetQty =0 and NetVal >0, NetVal, 0)

Update TACPos set UsedExp=NewUsedExp and BlockedMargin=NewBlockedMargin

Calculate

DeltaExp = NewUsedExp – OldUsedExp

DeltaMargin=NewBlockedMargin – OldBlockedMargin

Go to TACEXP Record

If (TACExp ExpLimit<>0 and ExpLimit – UsedExp < DeltaExp) OR (DExp ExpLimit<>0 and ExpLimit – UsedExp < DeltaExp)

Don't update TACEXP, DEXP

Update TACPOS (*revert changes to TACPOS*), set

UsedExp=UsedExp – DeltaExp

BlockedMargin=Blocked Margin – DeltaMargin

BOV=BOV–(OV–OLDOV)

BOQ=BOQ–(OQ–OLDOQ)

Reject Order

Exit

Else

Update TACEXP set

UsedExp=UsedExp + DeltaExp

BlockedMargin = BlockedMargin + DeltaMargin

Update DEXP set

UsedExp=UsedExp + DeltaExp

End if

Go to TM

If MarginLimit – MarginBlocked < Delta Margin

Don't update TM

Update TACEXP (*revert changes to TACEXP*)

Set UsedExp = UsedExp – DeltaExp

BlockedMargin = BlockedMargin - DeltaMargin

Update DEXP (*revert changes to DEXP*)

Set UsedExp = UsedExp – DeltaExp

Update TACPOS (*revert changes to TACPOS*), set

UsedExp=UsedExp – DeltaExp

BlockedMargin=Blocked Margin – DeltaMargin

BOV=BOV–(OV–OLDOV)

BOQ=BOQ–(OQ–OLDOQ)

Reject Order

Exit

Else

Update TM, set

Blocked Margin = Blocked Margin + DeltaMargin

Validate Order

Exit

End if

End if

7.2.2. Modification DNS Sell Order Margin Contracts

Read TACPos, UsedExposure and BlockedMargin

Set as OldUsedExposure and OldBlockedMargin

If OLDOV >= OV

Validate Order

Exit

Else

Update SOV with SOV+(OV–OLDOV) and SOQ with SOQ+(OQ–OLDOQ)

Calculate NewUsedExposure as

ABS (BOV+BTV – SOV–STV)

Calculate NewBlockedMargin as

A+B+E

Where

A= MAX (C, D)

C= [BOV – (if NetQty < 0, ABS(NetValue),0)] *BuyMargin %

D= [SOV – (if NetQty >0, ABS(NetVal),0)] * SellMargin %

B= ABS [NetVal* (If NetVal < 0, SellMargin %, BuyMargin %)]

E=(If NetQty =0 and NetVal >0, NetVal, 0)

Update TACPos set UsedExp=NewUsedExp and BlockedMargin=NewBlockedMargin

Calculate

DeltaExp = NewUsedExp – OldUsedExp

DeltaMargin=NewBlockedMargin – OldBlockedMargin

Go to TACEXP Record

If (TACExp ExpLimit<>0 and ExpLimit – UsedExp < DeltaExp) OR (DExp ExpLimit<>0 and ExpLimit – UsedExp < DeltaExp)

Don't update TACEXP, DEXP

Update TACPOS (*revert changes to TACPOS*), set

UsedExp=UsedExp – DeltaExp

BlockedMargin=Blocked Margin – DeltaMargin

SOV=SOV–(OV–OLDOV)

SOQ=SOQ–(OQ–OLDOQ)

Reject Order

Exit

Else

Update TACEXP set

UsedExp=UsedExp + DeltaExp

BlockedMargin = BlockedMargin + DeltaMargin

Update DEXP set

UsedExp=UsedExp + DeltaExp

End if

Go to TM

If MarginLimit – MarginBlocked < Delta Margin

Don't update TM

Update TACEXP (*revert changes to TACEXP*)

Set UsedExp = UsedExp – DeltaExp

BlockedMargin = BlockedMargin - DeltaMargin

Update DEXP (*revert changes to TACEXP*)

Set UsedExp = UsedExp – DeltaExp

Update TACPOS (*revert changes to TACPOS*), set

UsedExp=UsedExp – DeltaExp

BlockedMargin=Blocked Margin – DeltaMargin

SOV=SOV–(OV–OLDOV)

SOQ=SOQ–(OQ–OLDOQ)

Reject Order

Exit

Else

Update TM, set

Blocked Margin = Blocked Margin + DeltaMargin

Validate Order

Exit

End if

End if

7.2.3. Modification DNS Sell Orders for Stock Contracts

If OldOQ >= OQ

Validate Order

(This can lead to overexposure being allowed in cases where OV > OLDOV but it has been ignored as this can happen only with one modification and usually price ticks are much smaller than quantity ticks)

Exit

Else

Read TACPos, UsedExposure and BlockedStock

Set as OldUsedExposure, OldBlockedStock

If OQ – OLDOQ > StockLimit – BlockedStock

Reject order

Exit

Else

Update SOV with SOV+(OV–OLDOV) and SOQ with SOQ+(OQ–OLDOQ)

Calculate NewUsedExposure as

ABS (BOV+BTV – SOV–STV)

Calculate NewBlockedStock as

SOQ–NETQTY

Update TACPos set UsedExp=NewUsedExp and BlockedStock=NewBlockedStock

Calculate

DeltaExp = NewUsedExp – OldUsedExp

DeltaStock = NewBlockedStock – OldBlockedStock

Go to TACEXP Record

If (TACExp ExpLimit<>0 and ExpLimit – UsedExp < DeltaExp) OR (TACExp ExpLimit<>0 and ExpLimit – UsedExp < DeltaExp)

Don't update TACEXP, DEXP

Update TACPOS (*revert changes to TACPOS*), set

UsedExp=UsedExp – DeltaExp

BlockedStock=BlockedStock – DeltaStock

SOV=SOV–(OV–OLDOV)

SOQ=SOQ–(OQ–OLDOQ)

Reject Order

Exit

Else

Update TACEXP set

UsedExp=UsedExp + DeltaExp

Update DEXP set

UsedExp=UsedExp + DeltaExp

Validate Order

Exit

End if

7.2.4. Modification T2T Buy Orders

Read TACPos, UsedExposure and BlockedMargin

Set as OldUsedExposure and OldBlockedMargin

If OLDOV >= OV

Validate Order

Exit

Else

Update BOV with BOV+(OV–OLDOV) and BOQ with BOQ+(OQ–OLDOQ)

Calculate NewUsedExposure as

ABS (BOV+BTV + SOV+STV)

Calculate NewBlockedMargin as

P+Q

Where

P=(BOV+BTV) *BuyMargin %

Q= (SOV+STV] * SellMargin %

Update TACPos set UsedExp=NewUsedExp and BlockedMargin=NewBlockedMargin

Calculate

DeltaExp = NewUsedExp – OldUsedExp

DeltaMargin=NewBlockedMargin – OldBlockedMargin

Go to TACEXP Record

If (TACExp ExpLimit<>0 and ExpLimit – UsedExp < DeltaExp) OR (DExp ExpLimit<>0 and ExpLimit – UsedExp < DeltaExp)

Don't update TACEXP, DEXP

Update TACPOS (*revert changes to TACPOS*), set

UsedExp=UsedExp – DeltaExp

BlockedMargin=Blocked Margin – DeltaMargin

BOV=BOV–(OV–OLDOV)

BOQ=BOQ–(OQ–OLDOQ)

Reject Order

Exit

Else

Update TACEXP set

UsedExp=UsedExp + DeltaExp

BlockedMargin = BlockedMargin + DeltaMargin

Update TACEXP set

UsedExp=UsedExp + DeltaExp

End if

Go to TM

If MarginLimit – MarginBlocked < Delta Margin

Don't update TM

Update TACEXP (*revert changes to TACEXP*)

Set UsedExp = UsedExp – DeltaExp

BlockedMargin = BlockedMargin - DeltaMargin

Update DEXP (*revert changes to DEXP*)

Set UsedExp = UsedExp – DeltaExp

Update TACPOS (*revert changes to TACPOS*), set

UsedExp=UsedExp – DeltaExp

BlockedMargin=Blocked Margin – DeltaMargin

BOV=BOV–(OV–OLDOV)

BOQ=BOQ–(OQ–OLDOQ)

Reject Order

Exit

Else

Update TM, set

Blocked Margin = Blocked Margin + DeltaMargin

Validate Order

Exit

End if

End if

7.2.5. Modification T2T Sell Order Margin Contracts

Read TACPos, UsedExposure and BlockedMargin

Set as OldUsedExposure and OldBlockedMargin

If OLDOV >= OV

Validate Order

Exit

Else

Update SOV with SOV+(OV–OLDOV) and SOQ with SOQ+(OQ–OLDOQ)

Calculate NewUsedExposure as

ABS (BOV+BTV + SOV+STV)

Calculate NewBlockedMargin as

P+Q

Where

P=(BOV+BTV) *BuyMargin %

Q= (SOV+STV] * SellMargin %

Update TACPos set UsedExp=NewUsedExp and BlockedMargin=NewBlockedMargin

Calculate

DeltaExp = NewUsedExp – OldUsedExp

DeltaMargin=NewBlockedMargin – OldBlockedMargin

Go to TACEXP Record

If (TACExp ExpLimit<>0 and ExpLimit – UsedExp < DeltaExp) OR (DExp ExpLimit<>0 and ExpLimit – UsedExp < DeltaExp)

Don't update TACEXP, DEXP

Update TACPOS (*revert changes to TACPOS*), set

UsedExp=UsedExp – DeltaExp

BlockedMargin=Blocked Margin – DeltaMargin

SOV=SOV–(OV–OLDOV)

SOQ=SOQ–(OQ–OLDOQ)

Reject Order

Exit

Else

Update TACEXP set

UsedExp=UsedExp + DeltaExp

BlockedMargin = BlockedMargin + DeltaMargin

Update DEXP set

Update DEXP set

UsedExp=UsedExp + DeltaExp

End if

Go to TM

If MarginLimit – MarginBlocked < Delta Margin

Don't update TM

Update TACEXP (*revert changes to TACEXP*)

Set UsedExp = UsedExp – DeltaExp

BlockedMargin = BlockedMargin - DeltaMargin

Update DEXP (*revert changes to DEXP*)

Set UsedExp = UsedExp – DeltaExp

Update TACPOS (*revert changes to TACPOS*), set

UsedExp=UsedExp – DeltaExp

BlockedMargin=Blocked Margin – DeltaMargin

SOV=SOV–(OV–OLDOV)

SOQ=SOQ–(OQ–OLDOQ)

Reject Order

Exit

Else

Update TM, set

Blocked Margin = Blocked Margin + DeltaMargin

Validate Order

Exit

End if

End if

7.2.6. Modification T2T Sell Orders for Stock Contracts

If OldOQ >= OQ

Validate Order

(This can lead to overexposure being allowed in cases where OV > OLDOV but it has been ignored as this can happen only with one mdofication and usually price ticks are much smaller than quantity ticks)

Exit

Else

Read TACPos, UsedExposure and BlockedStock

Set as OldUsedExposure, OldBlockedStock

If (OQ–OldOQ) > StockLimit – BlockedStock

Reject order

Exit

Else

Update SOV with SOV+(OV–OLDOV) and SOQ with SOQ+(OQ–OLDOQ)

Calculate NewUsedExposure as

ABS (BOV+BTV – SOV–STV)

Calculate NewBlockedStock as

SOQ+STQ

Update TACPos set UsedExp=NewUsedExp and BlockedStock=NewBlockedStock

Calculate

DeltaExp = NewUsedExp – OldUsedExp

DeltaStock = NewBlockedStock – OldBlockedStock

Go to TACEXP Record

If (TACExp ExpLimit<>0 and ExpLimit – UsedExp < DeltaExp) OR (DExp ExpLimit<>0 and ExpLimit – UsedExp < DeltaExp)

Don't update TACEXP, DEXP

Update TACPOS (*revert changes to TACPOS*), set

UsedExp=UsedExp – DeltaExp

BlockedStock=BlockedStock – DeltaStock

SOV=SOV–(OV–OLDOV)

SOQ=SOQ–(OQ–OLDOQ)

Reject Order

Exit

Else

Update TACEXP set

UsedExp=UsedExp + DeltaExp

Update DEXP set

UsedExp=UsedExp + DeltaExp

Validate Order

Exit

End if

7.3. Modified Orders Confirmation Update.

7.3.1. Modification Confirmation DNS Buy Orders

Read TACPos, UsedExposure and BlockedMargin

Set as OldUsedExposure and OldBlockedMargin

If OLDOV <= OV

Exit

Else

Update BOV with BOV+(OV–OLDOV) and BOQ with BOQ+(OQ–OLDOQ)

Calculate NewUsedExposure as

ABS (BOV+BTV – SOV–STV)

Calculate NewBlockedMargin as

A+B+E

Where

A= MAX (C, D)

C= [BOV – (if NetQty < 0, ABS(NetValue),0)] *BuyMargin %

D= [SOV – (if NetQty >0, ABS(NetVal),0)] * SellMargin %

B= ABS [NetVal* (If NetVal < 0, SellMargin %, BuyMargin %)]

E=(If NetQty =0 and NetVal >0, NetVal, 0)

Update TACPos set UsedExp=NewUsedExp and BlockedMargin=NewBlockedMargin

Calculate

DeltaExp = NewUsedExp – OldUsedExp

DeltaMargin=NewBlockedMargin – OldBlockedMargin

Go to TACEXP, DEXP Record

Update TACEXP set

UsedExp=UsedExp + DeltaExp

BlockedMargin = BlockedMargin + DeltaMargin

Update DEXP set

UsedExp=UsedExp + DeltaExp

Go to TM

Update TM, set

Blocked Margin = Blocked Margin + DeltaMargin

Exit

End if

7.3.2. Modification Confirmation DNS Sell Order Margin Contracts

Read TACPos, UsedExposure and BlockedMargin

Set as OldUsedExposure and OldBlockedMargin

If OLDOV <= OV

Exit

Else

Update SOV with SOV+(OV–OLDOV) and SOQ with SOQ+(OQ–OLDOQ)

Calculate NewUsedExposure as

ABS (BOV+BTV – SOV–STV)

Calculate NewBlockedMargin as

A+B+E

Where

A= MAX (C, D)

C= [BOV – (if NetQty < 0, ABS(NetValue),0)] *BuyMargin %

D= [SOV – (if NetQty >0, ABS(NetVal),0)] * SellMargin %

B= ABS [NetVal* (If NetVal < 0, SellMargin %, BuyMargin %)]

E=(If NetQty =0 and NetVal >0, NetVal, 0)

Update TACPos set UsedExp=NewUsedExp and BlockedMargin=NewBlockedMargin

Calculate

DeltaExp = NewUsedExp – OldUsedExp

DeltaMargin=NewBlockedMargin – OldBlockedMargin

Go to TACEXP, DEXP Record

Update TACEXP set

UsedExp=UsedExp + DeltaExp

BlockedMargin = BlockedMargin + DeltaMargin

Update DEXP set

UsedExp=UsedExp + DeltaExp

Go to TM

Update TM, set

Blocked Margin = Blocked Margin + DeltaMargin

Exit

End if

7.3.3. Modification Confirmation DNS Sell Orders for Stock Contracts

If OldOQ <= OQ

Exit

Else

Read TACPos, UsedExposure and BlockedStock

Set as OldUsedExposure, OldBlockedStock

Update SOV with SOV+(OV–OLDOV) and SOQ with SOQ+(OQ–OLDOQ)

Calculate NewUsedExposure as

ABS (BOV+BTV – SOV–STV)

Calculate NewBlockedStock as

SOQ–NETQTY

Update TACPos set UsedExp=NewUsedExp and BlockedStock=NewBlockedStock

Calculate

DeltaExp = NewUsedExp – OldUsedExp

Go to TACEXP, DEXP Record

Update TACEXP set

UsedExp=UsedExp + DeltaExp

Update DEXP set

UsedExp=UsedExp + DeltaExp

End if

7.3.4. Modification Confirmation T2T Buy Orders

Read TACPos, UsedExposure and BlockedMargin

Set as OldUsedExposure and OldBlockedMargin

If OLDOV <= OV

Exit

Else

Update BOV with BOV+(OV–OLDOV) and BOQ with BOQ+(OQ–OLDOQ)

Calculate NewUsedExposure as

ABS (BOV+BTV + SOV+STV)

Calculate NewBlockedMargin as

P+Q

Where

P=(BOV+BTV) *BuyMargin %

Q= (SOV+STV] * SellMargin %

Update TACPos set UsedExp=NewUsedExp and BlockedMargin=NewBlockedMargin

Calculate

DeltaExp = NewUsedExp – OldUsedExp

DeltaMargin=NewBlockedMargin – OldBlockedMargin

Go to TACEXP Record

Update TACEXP set

UsedExp=UsedExp + DeltaExp

BlockedMargin = BlockedMargin + DeltaMargin

Go to TM

Update TM, set

Blocked Margin = Blocked Margin + DeltaMargin

Exit

End if

7.3.5. Modification Confirmation T2T Sell Order Margin Contracts

Read TACPos, UsedExposure and BlockedMargin

Set as OldUsedExposure and OldBlockedMargin

If OLDOV <= OV

Exit

Else

Update SOV with SOV+(OV–OLDOV) and SOQ with SOQ+(OQ–OLDOQ)

Calculate NewUsedExposure as

ABS (BOV+BTV + SOV+STV)

Calculate NewBlockedMargin as

P+Q

Where

P=(BOV+BTV) *BuyMargin %

Q= (SOV+STV] * SellMargin %

Update TACPos set UsedExp=NewUsedExp and BlockedMargin=NewBlockedMargin

Calculate

DeltaExp = NewUsedExp – OldUsedExp

DeltaMargin=NewBlockedMargin – OldBlockedMargin

Go to TACEXP Record

Update TACEXP set

UsedExp=UsedExp + DeltaExp

BlockedMargin = BlockedMargin + DeltaMargin

Go to TM

Update TM, set

Blocked Margin = Blocked Margin + DeltaMargin

Exit

End if

7.3.6. Modification Confirmation T2t Sell Orders for Stock Contracts

If OldOQ <= OQ

Exit

Else

Read TACPos, UsedExposure and BlockedStock

Set as OldUsedExposure, OldBlockedStock

Update SOV with SOV+(OV–OLDOV) and SOQ with SOQ+(OQ–OLDOQ)

Calculate NewUsedExposure as

ABS (BOV+BTV – SOV–STV)

Calculate NewBlockedStock as

SOQ+STQ

Update TACPos set UsedExp=NewUsedExp and BlockedStock=NewBlockedStock

Calculate

DeltaExp = NewUsedExp – OldUsedExp

Go to TACEXP Record

Update TACEXP set

UsedExp=UsedExp + DeltaExp

End if

7.4. Modification Rejection Update.

7.4.1. Modification Rejection DNS Buy Orders

Read TACPos, UsedExposure and BlockedMargin

Set as OldUsedExposure and OldBlockedMargin

If OLDOV >= OV

Exit

Else

Update BOV with BOV+(OV–OLDOV) and BOQ with BOQ+(OQ–OLDOQ)

Calculate NewUsedExposure as

ABS (BOV+BTV – SOV–STV)

Calculate NewBlockedMargin as

A+B+E

Where

A= MAX (C, D)

C= [BOV – (if NetQty < 0, ABS(NetValue),0)] *BuyMargin %

D= [SOV – (if NetQty >0, ABS(NetVal),0)] * SellMargin %

B= ABS [NetVal* (If NetVal < 0, SellMargin %, BuyMargin %)]

E=(If NetQty =0 and NetVal >0, NetVal, 0)

Update TACPos set UsedExp=NewUsedExp and BlockedMargin=NewBlockedMargin

Calculate

DeltaExp = NewUsedExp – OldUsedExp

DeltaMargin=NewBlockedMargin – OldBlockedMargin

Go to TACEXP Record

Update TACEXP set

UsedExp=UsedExp + DeltaExp

BlockedMargin = BlockedMargin + DeltaMargin

Go to TM

Update TM, set

Blocked Margin = Blocked Margin + DeltaMargin

Exit

End if

7.4.2. Modification Rejection DNS Sell Order Margin Contracts

Read TACPos, UsedExposure and BlockedMargin

Set as OldUsedExposure and OldBlockedMargin

If OLDOV >= OV

Exit

Else

Update SOV with SOV+(OV–OLDOV) and SOQ with SOQ+(OQ–OLDOQ)

Calculate NewUsedExposure as

ABS (BOV+BTV – SOV–STV)

Calculate NewBlockedMargin as

A+B+E

Where

A= MAX (C, D)

C= [BOV – (if NetQty < 0, ABS(NetValue),0)] *BuyMargin %

D= [SOV – (if NetQty >0, ABS(NetVal),0)] * SellMargin %

B= ABS [NetVal* (If NetVal < 0, SellMargin %, BuyMargin %)]

E=(If NetQty =0 and NetVal >0, NetVal, 0)

Update TACPos set UsedExp=NewUsedExp and BlockedMargin=NewBlockedMargin

Calculate

DeltaExp = NewUsedExp – OldUsedExp

DeltaMargin=NewBlockedMargin – OldBlockedMargin

Go to TACEXP Record

Update TACEXP set

UsedExp=UsedExp + DeltaExp

BlockedMargin = BlockedMargin + DeltaMargin

Go to TM

Update TM, set

Blocked Margin = Blocked Margin + DeltaMargin

Exit
End if

7.4.3. Modification Rejection DNS Sell Orders for Stock Contracts

If OldOQ >= OQ
    Exit
Else
Read TACPos, UsedExposure and BlockedStock
Set as OldUsedExposure, OldBlockedStock
Update SOV with SOV+(OV–OLDOV) and SOQ with SOQ+(OQ–OLDOQ)
Calculate NewUsedExposure as
    ABS (BOV+BTV – SOV–STV)
Calculate NewBlockedStock as
    SOQ–NETQTY
Update TACPos set UsedExp=NewUsedExp and BlockedStock=NewBlockedStock
Calculate
    DeltaExp = NewUsedExp – OldUsedExp
Go to TACEXP Record
    Update TACEXP set
    UsedExp=UsedExp + DeltaExp
End if

7.4.4. Modification Rejection T2T Buy Orders

Read TACPos, UsedExposure and BlockedMargin
Set as OldUsedExposure and OldBlockedMargin
If OLDOV >= OV
    Exit
Else
Update BOV with BOV+(OV–OLDOV) and BOQ with BOQ+(OQ–OLDOQ)
Calculate NewUsedExposure as
    ABS (BOV+BTV + SOV+STV)
Calculate NewBlockedMargin as
    P+Q

Where

P=(BOV+BTV) *BuyMargin %

Q= (SOV+STV] * SellMargin %

Update TACPos set UsedExp=NewUsedExp and BlockedMargin=NewBlockedMargin

Calculate

DeltaExp = NewUsedExp – OldUsedExp

DeltaMargin=NewBlockedMargin – OldBlockedMargin

Go to TACEXP Record

Update TACEXP set

UsedExp=UsedExp + DeltaExp

BlockedMargin = BlockedMargin + DeltaMargin

Go to TM

Update TM, set

Blocked Margin = Blocked Margin + DeltaMargin

Exit

End if

7.4.5. Modification Rejection T2T Sell Order Margin Contracts

Read TACPos, UsedExposure and BlockedMargin

Set as OldUsedExposure and OldBlockedMargin

If OLDOV >= OV

Exit

Else

Update SOV with SOV+(OV–OLDOV) and SOQ with SOQ+(OQ–OLDOQ)

Calculate NewUsedExposure as

ABS (BOV+BTV + SOV+STV)

Calculate NewBlockedMargin as

P+Q

Where

P=(BOV+BTV) *BuyMargin %

Q= (SOV+STV] * SellMargin %

Update TACPos set UsedExp=NewUsedExp and BlockedMargin=NewBlockedMargin

Calculate

DeltaExp = NewUsedExp – OldUsedExp

DeltaMargin=NewBlockedMargin – OldBlockedMargin

Go to TACEXP Record

Update TACEXP set

UsedExp=UsedExp + DeltaExp

BlockedMargin = BlockedMargin + DeltaMargin

Go to TM

Update TM, set

Blocked Margin = Blocked Margin + DeltaMargin

Exit

End if

7.4.6. Modification Rejection T2t Sell Orders for Stock Contracts

If OldOQ >= OQ

Exit

Else

Read TACPos, UsedExposure and BlockedStock

Set as OldUsedExposure, OldBlockedStock

Update SOV with SOV+(OV–OLDOV) and SOQ with SOQ+(OQ–OLDOQ)

Calculate NewUsedExposure as

ABS (BOV+BTV – SOV–STV)

Calculate NewBlockedStock as

SOQ+STQ

Update TACPos set UsedExp=NewUsedExp and BlockedStock=NewBlockedStock

Calculate

DeltaExp = NewUsedExp – OldUsedExp

DeltaStock = NewBlockedStock – OldBlockedStock

Go to TACEXP Record

Update TACEXP set

UsedExp=UsedExp + DeltaExp

End if

7.5. Order Cancellation Update.

7.5.1. Cancellation Update DNS Buy Orders

Read TACPos, UsedExposure and BlockedMargin

Set as OldUsedExposure and OldBlockedMargin

Update BOV with BOV–OV and BOQ with BOQ–OQ

Calculate NewUsedExposure as

ABS (BOV+BTV – SOV–STV)

Calculate NewBlockedMargin as

A+B+E

Where

A= MAX (C, D)

C= [BOV – (if NetQty < 0, ABS(NetValue),0)] *BuyMargin %

D= [SOV – (if NetQty >0, ABS(NetVal),0)] * SellMargin %

B= ABS [NetVal* (If NetVal < 0, SellMargin %, BuyMargin %)]

E=(If NetQty =0 and NetVal >0, NetVal, 0)

Update TACPos set UsedExp=NewUsedExp and BlockedMargin=NewBlockedMargin

Calculate

DeltaExp = NewUsedExp – OldUsedExp

DeltaMargin=NewBlockedMargin – OldBlockedMargin

Go to TACEXP Record

Update TACEXP set

UsedExp=UsedExp + DeltaExp

BlockedMargin = BlockedMargin + DeltaMargin

Go to TM

Update TM, set

Blocked Margin = Blocked Margin + DeltaMargin

Exit

7.5.2. Cancellation Update DNS Sell Order Margin Contracts

Read TACPos, UsedExposure and BlockedMargin

Set as OldUsedExposure and OldBlockedMargin

Update SOV with SOV–OV and SOQ with SOQ–OQ

Calculate NewUsedExposure as

ABS (BOV+BTV – SOV–STV)

Calculate NewBlockedMargin as

A+B+E

Where

A= MAX (C, D)

C= [BOV – (if NetQty < 0, ABS(NetValue),0)] *BuyMargin %

D= [SOV – (if NetQty >0, ABS(NetVal),0)] * SellMargin %

B= ABS [NetVal* (If NetVal < 0, SellMargin %, BuyMargin %)]

E=(If NetQty =0 and NetVal >0, NetVal, 0)

Update TACPos set UsedExp=NewUsedExp and BlockedMargin=NewBlockedMargin

Calculate

DeltaExp = NewUsedExp – OldUsedExp

DeltaMargin=NewBlockedMargin – OldBlockedMargin

Go to TACEXP Record

Update TACEXP set

UsedExp=UsedExp + DeltaExp

BlockedMargin = BlockedMargin + DeltaMargin

Go to TM

Update TM, set

Blocked Margin = Blocked Margin + DeltaMargin

Exit

7.5.3. Cancellation Update DNS Sell Orders for Stock Contracts

Read TACPos, UsedExposure and BlockedStock

Set as OldUsedExposure, OldBlockedStock

Update SOV with SOV–OV and SOQ with SOQ–OQ

Calculate NewUsedExposure as

ABS (BOV+BTV – SOV–STV)

Calculate NewBlockedStock as

SOQ–NETQTY

Update TACPos set UsedExp=NewUsedExp and BlockedStock=NewBlockedStock

Calculate

DeltaExp = NewUsedExp – OldUsedExp

Go to TACEXP Record

Update TACEXP set

UsedExp=UsedExp + DeltaExp

Exit

7.5.4. Cancellation Update T2T Buy Orders

Read TACPos, UsedExposure and BlockedMargin

Set as OldUsedExposure and OldBlockedMargin

Update BOV with BOV–OV and BOQ with BOQ–OQ

Calculate NewUsedExposure as

ABS (BOV+BTV + SOV+STV)

Calculate NewBlockedMargin as

P+Q

Where

P=(BOV+BTV) *BuyMargin %

Q= (SOV+STV] * SellMargin %

Update TACPos set UsedExp=NewUsedExp and BlockedMargin=NewBlockedMargin

Calculate

DeltaExp = NewUsedExp – OldUsedExp

DeltaMargin=NewBlockedMargin – OldBlockedMargin

Go to TACEXP Record

Update TACEXP set

UsedExp=UsedExp + DeltaExp

BlockedMargin = BlockedMargin + DeltaMargin

Go to TM

Update TM, set

Blocked Margin = Blocked Margin + DeltaMargin

Exit

End if

7.5.5. Cancellation Update T2T Sell Order Margin Contracts

Read TACPos, UsedExposure and BlockedMargin

Set as OldUsedExposure and OldBlockedMargin

Update SOV with SOV– OV and SOQ with SOQ–OQ

Calculate NewUsedExposure as

ABS (BOV+BTV + SOV+STV)

Calculate NewBlockedMargin as

P+Q

Where

P=(BOV+BTV) *BuyMargin %

Q= (SOV+STV] * SellMargin %

Update TACPos set UsedExp=NewUsedExp and BlockedMargin=NewBlockedMargin

Calculate

DeltaExp = NewUsedExp – OldUsedExp

DeltaMargin=NewBlockedMargin – OldBlockedMargin

Go to TACEXP Record

Update TACEXP set

UsedExp=UsedExp + DeltaExp

BlockedMargin = BlockedMargin + DeltaMargin

Go to TM

Update TM, set

Blocked Margin = Blocked Margin + DeltaMargin

Exit

7.5.6. Cancellation Update T2T Sell Orders for Stock Contracts

Read TACPos, UsedExposure and BlockedStock

Set as OldUsedExposure, OldBlockedStock

Update SOV with SOV–OV and SOQ with SOQ–OQ

Calculate NewUsedExposure as

ABS (BOV+BTV – SOV–STV)

Calculate NewBlockedStock as

SOQ+STQ

Update TACPos set UsedExp=NewUsedExp and BlockedStock=NewBlockedStock

Calculate

DeltaExp = NewUsedExp – OldUsedExp

DeltaStock = NewBlockedStock – OldBlockedStock

Go to TACEXP Record

Update TACEXP set

UsedExp=UsedExp + DeltaExp

Exit

7.6. Trade Confirmation Update All Orders Except Auction Bids.

7.6.1. Trade Update DNS Buy Orders

Read TACPos, UsedExposure and BlockedMargin

Set as OldUsedExposure and OldBlockedMargin

Update BOV with BOV–(TQ*OP) and BOQ with BOQ–TQ and BTV with BTV+TV and BTQ with BTQ+TQ

Calculate NewUsedExposure as

ABS (BOV+BTV – SOV–STV)

Calculate NewBlockedMargin as

A+B+E

Where

A= MAX (C, D)

C= [BOV – (if NetQty < 0, ABS(NetValue),0)] *BuyMargin %

D= [SOV – (if NetQty >0, ABS(NetVal),0)] * SellMargin %

B= ABS [NetVal* (If NetVal < 0, SellMargin %, BuyMargin %)]

E=(If NetQty =0 and NetVal >0, NetVal, 0)

Update TACPos set UsedExp=NewUsedExp and BlockedMargin=NewBlockedMargin

Calculate

DeltaExp = NewUsedExp – OldUsedExp

DeltaMargin=NewBlockedMargin – OldBlockedMargin

Go to TACEXP Record

Update TACEXP set

UsedExp=UsedExp + DeltaExp

BlockedMargin = BlockedMargin + DeltaMargin

Go to TM

Update TM, set

Blocked Margin = Blocked Margin + DeltaMargin

Exit

7.6.2. Trade Update DNS Sell Order Margin Contracts

Read TACPos, UsedExposure and BlockedMargin

Set as OldUsedExposure and OldBlockedMargin

Update SOV with SOV–(TQ*OP) and SOQ with SOQ–TQ and STV with STV+TV and STQ with STQ+TQ

Calculate NewUsedExposure as

ABS (BOV+BTV – SOV–STV)

Calculate NewBlockedMargin as

A+B+E

Where

A= MAX (C, D)

C= [BOV – (if NetQty < 0, ABS(NetValue),0)] *BuyMargin %

D= [SOV – (if NetQty >0, ABS(NetVal),0)] * SellMargin %

B= ABS [NetVal* (If NetVal < 0, SellMargin %, BuyMargin %)]

E=(If NetQty =0 and NetVal >0, NetVal, 0)

Update TACPos set UsedExp=NewUsedExp and BlockedMargin=NewBlockedMargin

Calculate

DeltaExp = NewUsedExp – OldUsedExp

DeltaMargin=NewBlockedMargin – OldBlockedMargin

Go to TACEXP Record

Update TACEXP set

UsedExp=UsedExp + DeltaExp

BlockedMargin = BlockedMargin + DeltaMargin

Go to TM

Update TM, set

Blocked Margin = Blocked Margin + DeltaMargin

Exit

7.6.3. Trade Update DNS Sell Orders for Stock Contracts

Read TACPos, UsedExposure and BlockedStock

Set as OldUsedExposure, OldBlockedStock

Update SOV with SOV–(TQ*OP) and SOQ with SOQ–OQ and STV with STV+TV and STQ with STQ+TQ

Calculate NewUsedExposure as

ABS (BOV+BTV – SOV–STV)

Calculate NewBlockedStock as

SOQ–NETQTY

Update TACPos set UsedExp=NewUsedExp and BlockedStock=NewBlockedStock

Calculate

DeltaExp = NewUsedExp – OldUsedExp

DeltaStock = NewBlockedStock – OldBlockedStock

Go to TACEXP Record

Update TACEXP set

UsedExp=UsedExp + DeltaExp

Exit

7.6.4. Trade Update T2T Buy Orders

Read TACPos, UsedExposure and BlockedMargin

Set as OldUsedExposure and OldBlockedMargin

Update BOV with BOV–(TQ*OP) and BOQ with BOQ–TQ and STV with BTV+TV and BTQ with BTQ+TQ

Calculate NewUsedExposure as

ABS (BOV+BTV + SOV+STV)

Calculate NewBlockedMargin as

P+Q

Where

P=(BOV+BTV) *BuyMargin %

Q= (SOV+STV] * SellMargin %

Update TACPos set UsedExp=NewUsedExp and BlockedMargin=NewBlockedMargin

Calculate

DeltaExp = NewUsedExp – OldUsedExp

DeltaMargin=NewBlockedMargin – OldBlockedMargin

Go to TACEXP Record

Update TACEXP set

UsedExp=UsedExp + DeltaExp

BlockedMargin = BlockedMargin + DeltaMargin

Go to TM

Update TM, set

Blocked Margin = Blocked Margin + DeltaMargin

Exit

End if

7.6.5. Trade Update T2T Sell Order Margin Contracts

Read TACPos, UsedExposure and BlockedMargin

Set as OldUsedExposure and OldBlockedMargin

Update SOV with SOV– (TQ*OP) and SOQ with SOQ–TQ and STV with STV+TV and STQ with STQ+TQ

Calculate NewUsedExposure as

ABS (BOV+BTV + SOV+STV)

Calculate NewBlockedMargin as

P+Q

Where

P=(BOV+BTV) *BuyMargin %

Q= (SOV+STV] * SellMargin %

Update TACPos set UsedExp=NewUsedExp and BlockedMargin=NewBlockedMargin

Calculate

DeltaExp = NewUsedExp – OldUsedExp

DeltaMargin=NewBlockedMargin – OldBlockedMargin

Go to TACEXP Record

Update TACEXP set

UsedExp=UsedExp + DeltaExp

BlockedMargin = BlockedMargin + DeltaMargin

Go to TM

Update TM, set

Blocked Margin = Blocked Margin + DeltaMargin

Exit

7.6.6. Trade Update T2T Sell Orders for Stock Contracts

Read TACPos, UsedExposure and BlockedStock
Set as OldUsedExposure, OldBlockedStock
Update SOV with SOV–(TQ*OP) and SOQ with SOQ–TQ and STV with STV+TV and STQ with STQ+TQ
Calculate NewUsedExposure as
ABS (BOV+BTV – SOV–STV)
Calculate NewBlockedStock as
SOQ+STQ
Update TACPos set UsedExp=NewUsedExp and BlockedStock=NewBlockedStock
Calculate
DeltaExp = NewUsedExp – OldUsedExp
Go to TACEXP Record
Update TACEXP set
UsedExp=UsedExp + DeltaExp
Exit

7.7. BID Validation and Updates

7.7.1. Buy Bids Validation

Read BIDPOS
If BV < BBV
Reject Bid
Exit
Else
Read BIDNO, BBQ, BBV
Update and set PrevBIDNO=BIDNO, PrevBBQ=BBQ, PrevBBV=BBV
Update and set BIDNO=BNO, BBQ=BQ, BBV=BV

Read TACPOS : BlockedMargin
Set as OldBlockedMargin
Update BOV with BOV+BBV-PrevBBV and BOQ with BOQ+BBQ-PrevBBQ
Calculate NewBlockedMargin as
P+Q
Where

P=(BOV+BTV) *BuyMargin %

Q= (SOV+STV] * SellMargin %

Update TACPOS set BlockedMargin=NewBlockedMargin

Calculate

DeltaMargin=NewBlockedMargin – OldBlockedMargin

Go to TM

If MarginLimit – MarginBlocked < Delta Margin

Don't update TM

Update TACPOS (*revert changes to TACPOS*), set

BlockedMargin=Blocked Margin – DeltaMargin

BOV=BOV+PrevBBV-BBV

BOQ=BOQ+PrevBBQ-BBQ

Update BIDPOS (*revert changes to BIDPOS*)

Set BBQ=PrevBBQ, BIDNO=PrevBIDNO, BBV=PrevBBV

Set PrevBBQ, PrevBBV, PrevBIDNO to zero

Reject Order

Exit

Else

Update TM, set

Blocked Margin = Blocked Margin + DeltaMargin

Validate BID

Exit

End if

7.7.2. Sell Bids Validation Margin Contracts

Read BIDPOS

If BV > SBV

Reject Bid

Exit

Else

Read BIDNO, SBQ, SBV

Update and set PrevBIDNO=BIDNO, PrevSBQ=SBQ, PrevSBV=SBV

Update and set BIDNO=BNO, SBQ=BQ, SBV=BV

Read TACPOS : BlockedMargin
Set as OldBlockedMargin
Update SOV with SOV+SBV-PrevSBV and SOQ with SOQ+SBQ-PrevSBQ
Calculate NewBlockedMargin as
P+Q
Where
P=(BOV+BTV) *BuyMargin %
Q= (SOV+STV] * SellMargin %

Update TACPOS set BlockedMargin=NewBlockedMargin
Calculate
DeltaMargin=NewBlockedMargin – OldBlockedMargin
Go to TM
If MarginLimit – MarginBlocked < Delta Margin
Don't update TM
Update TACPOS (*revert changes to TACPOS*), set
BlockedMargin=Blocked Margin – DeltaMargin
SOV=SOV+PrevSBV-SBV
SOQ=SOQ+PrevSBQ-SBQ
Update BIDPOS (*revert changes to BIDPOS*)
Set SBQ=PrevSBQ, BIDNO=PrevBIDNO, SBV=PrevSBV
Set PrevSBQ, PrevSBV, PrevBIDNO to zero
Reject Order
Exit
Else
Update TM, set
Blocked Margin = Blocked Margin + DeltaMargin
Validate BID
Exit
End if

7.7.3. Sell Bids Validation Stock Contracts

Read TACPOS

If BQ > StockLimit - BlockedStock

Reject Bid

Exit

Else if BQ = SBQ (BIDPOS)

Accept Bid

Exit

Else

Update TACPOS

Update SOV with SOV+BV and SOQ with SOQ+BQ

Set BlockedStock = STQ+SOQ

Update TACPOS with BlockedStock

Read BIDPOS

Read BIDNO, SBQ, SBV

Update and set PrevBIDNO=BIDNO, PrevSBQ=SBQ, PrevSBV=SBV

Update and set BIDNO=BNO, SBQ=BQ, SBV=BV

Accept Bid

Exit

End if

7.7.4. Rejection or Knock-off Buy Bids

Read BIDPOS

Read BIDNO, BBQ, BBV

If BIDNO=PREVBIDNO

Set PrevBIDNO, PrevBBQ, PrevBBV to zero

Exit

Else if BIDNO=BIDNO

Update and set BIDNO=PrevBIDNO, BBQ=PrevBBQ, BBV=PrevBBV

Update and set PrevBIDNO, PrevBBQ, PrevBBV to zero

Else if BIDNO<>PrevBIDNO and BIDNO<>BIDNO

Log a fatal error

Exit

End if

Read TACPOS : BlockedMargin

Set as OldBlockedMargin

Update BOV with SUM(BBV) and BOQ with SUM(BBQ)

Calculate NewBlockedMargin as

P+Q

Where

P=(BOV+BTV) *BuyMargin %

Q= (SOV+STV) * SellMargin %

Update TACPOS set BlockedMargin=NewBlockedMargin

Calculate

DeltaMargin=NewBlockedMargin – OldBlockedMargin

Go to TM

Update TM, set

Blocked Margin = Blocked Margin + DeltaMargin

Validate BID

Exit

7.7.5. Rejection or Knock-off Sell Bids Margin Contracts

Read BIDPOS

Read BIDNO, SBQ, SBV

If BIDNO=PREVBIDNO

Set PrevBIDNO, PrevSBQ, PrevSBV to zero

Exit

Else if BIDNO=BIDNO

Update and set BIDNO=PrevBIDNO, SBQ=PrevSBQ, SBV=PrevSBV

Update and set PrevBIDNO, PrevSBQ, PrevSBV to zero

Else if BIDNO<>PrevBIDNO and BIDNO<>BIDNO

Log a fatal error

Exit

End if

Read TACPOS : BlockedMargin

Set as OldBlockedMargin

Update SOV with SUM(SBV) and SOQ with SUM(SBQ)

Calculate NewBlockedMargin as

P+Q

Where

P=(BOV+BTV) *BuyMargin %

Q= (SOV+STV) * SellMargin %

Update TACPOS set BlockedMargin=NewBlockedMargin

Calculate

DeltaMargin=NewBlockedMargin – OldBlockedMargin

Go to TM

Update TM, set

Blocked Margin = Blocked Margin + DeltaMargin

Validate BID

Exit

7.7.6. Rejection or Knock-off Sell Bids Stock Contracts

Read BIDPOS

Read BIDNO, SBQ, SBV

If BIDNO=PREVBIDNO

Set PrevBIDNO, PrevSBQ, PrevSBV to zero

Exit

Else if BIDNO=BIDNO

Update and set BIDNO=PrevBIDNO, SBQ=PrevSBQ, SBV=PrevSBV

Update and set PrevBIDNO, PrevSBQ, PrevSBV to zero

Else if BIDNO<>PrevBIDNO and BIDNO<>BIDNO

Log a fatal error

Exit

End if

Update TACPOS set

SOV with SUM(SBV) and SOQ with SUM(SBQ)

BlockedStock = STQ+SOQ, update blocked_margin

7.7.7. Trade Updates for Buy Bids

Read BIDPOS

Delete BIDPOS record

Read TACPOS : BlockedMargin

Set as OldBlockedMargin

Update BOV with SUM(BBV), BOQ with SUM(BBQ) and BTV with TV, BTQ with TQ.

Calculate NewBlockedMargin as

P+Q

Where

P=(BOV+BTV) *BuyMargin %

Q= (SOV+STV) * SellMargin %

Update TACPOS set BlockedMargin=NewBlockedMargin

Calculate

DeltaMargin=NewBlockedMargin – OldBlockedMargin

Go to TM

Update TM, set

Blocked Margin = Blocked Margin + DeltaMargin

Exit

7.7.8. Trade Updates for Sell Bids Margin Contract

Read BIDPOS

Delete BIDPOS record

Read TACPOS : BlockedMargin

Set as OldBlockedMargin

Update SOV with SUM(SBV), SOQ with SUM(SBQ) and BTV with TV, BTQ with TQ.

Calculate NewBlockedMargin as

P+Q

Where

P=(BOV+BTV) *BuyMargin %

Q= (SOV+STV) * SellMargin %

Update TACPOS set BlockedMargin=NewBlockedMargin
Calculate
DeltaMargin=NewBlockedMargin – OldBlockedMargin
Go to TM
Update TM, set
Blocked Margin = Blocked Margin + DeltaMargin
Exit

7.7.9. Trade Updates for Sell Bids Stock Check Contract

Read BIDPOS
Delete BIDPOS record
Read TACPOS
Update SOV with SUM(SBV), SOQ with SUM(SBQ) and BTV with TV, BTQ with TQ.
Update Blocked Stock with STQ+SOQ, update blocked_margin

7.8. Recalculate

For each TACPOS record
ReCalculate and update Exposure Used and Blocked Margin based on appropriate formulae.
Read TACPOS SUM (Used Exposure), SUM (Blocked Margin) group by Trading Account ID and update TACExp Used Exposure and Blocked Margin.
Read TACExp SUM(Used Exposure) group by Dealer ID and Update DealerExp Used Exposure.
Read TACExp SUM(Blocked Margin) and Update TMMARGIN Blocked Margin)

7.9. Updates from External Systems

RMS receives margin updates from the PFM (TM wise) and Stock updates from the CMS (TAC wise). The updates are processed as indicated in the sections below:

7.9.1. Gross margin credit update

RMS validates if the new margin limit is sufficient to meet the blocked margins. If yes, the same will be updated against the TM's margin limit else the update will be rejected.

7.9.2. Margin credit increment update

RMS adds the incremental amount to the TM's margin limit.

7.9.3. Margin credit decrement update

RMS validates if the margin limit after deducting the decrement amount is sufficient to meet the blocked margins. If yes, the same will be deducted from the TM's margin limit, else the update will be rejected.

7.9.4. Incremental locked stock addition update

RMS adds the incremental locked stock to the TAC's stock limit for the contract.

7.9.5. Incremental locked stock deduction update

RMS validates if the stock limit after deducting the incremental locked stock is sufficient to meet the blocked margins. If yes, the same will be deducted from the TAC's stock limit, else the update will be rejected.

8. Price Time Priority Matching Engine

8.1. Overview

PTP ME is responsible for matching orders based on the price time priority algorithm and generating the corresponding trades.

8.2. Functions

8.2.1. Receiving Order Requests

PTP matching eingine receives following types of order requests from ORS

- New order
- Modify order
- Cancel order

8.2.2. Order Types and Conditions

PTP matching engine supports following order types and conditions.

- Market (Immediate or Cancel) order
- Limit order
- Partial Disclosure Order
- OTC market order
- OTC specified rate order
- All or none i.e. entire order quantity match in a single trade or no match.

8.2.3. Processing Order Requests

PTP matching engine maintains 4 logical books (queues) Buy Book, Sell Book, OTC Buy Book, OTC Sell Book per contract. Uses the price (highest for buy order, lowest for sell order) priority and then within price, time priority in matching an incoming order with an order in the opposite book.

On matching, generates trades corresponding to both orders and sends trade messages to ORS and in case of no match, places the order in the order queue for future processing, except in case of market order in which case a cancelation response is sent. Complete PTP matching algorithm is specified in section 8.10.

8.2.4. Execution Reports to ORS

PTP matching engine sends following types of execution reports to ORS

- Order confirmation
- Order modification confirmation
- Order modification rejection
- Order cancelation rejection
- Trade
- Order cancelation
- Order expired due to session expiry.

8.2.5. Market Data Updates

PTP Matching Engine sends market data updates to Broadcast Engine. On every trade it sends LTP to BE. On every book (queue) update, it sends top n rates and quantities from the updated book.

8.3. Configuration

ME.conf

ME_INSTANCE_ID=01

ME_IP_ADDRESS=172.12.25.4

ME_PORT=2024

MATCHING_TYPE = PTP

TRADING_CALENDAR_ID = S101

DB_CONN_STRING = jdbc:oracle://172.12.25.10:3333/NSPOT_DB

IN_MEM_DB_CONN_STRING = <embedded database info>

DB_USERNAME = NSPOTUSR

DB_PASSWORD = NSPOTPW

IN_MEM_DB_USERNAME = NSPOTUSR

IN_MEM_DB_PASSWORD = NSPOTPW

TRADING_SESSION_MONITOR_INTERVAL = 30

8.4. In-memory artifacts

8.4.1. ME-Map

ME Map is loaded from ME.conf on startup and consists of the following information:

- ME Instance ID

- IP Address
- Port
- Matching Type
- Trading Session ID
- Session Start Date and Time
- Session End Date and Time
- Trading Session Monitor Interval

8.4.2. Books or Queues

Buy Queue, **Sell Queue**, **OTC Buy Queue** & **OTC Sell Queue** structures are loaded on startup for each contract. Each queue consists of the following information:

- Order No
- Client Order No
- Original Client Order No
- Order Type
- Order Side
- Contract Code
- TM Code
- TAC
- AON/NON-AON
- Expiry Date
- Counterparty TM Code
- Price (will hold base price/LTP for OTC market orders)
- Quantity
- Original Quantity
- Disclosed Quantity
- Timestamp

8.5. Startup Process

Startup process sets up an executor service to handle concurrent execution of tasks. Sets up a scheduled executor service to handle scheduled tasks. It loads the in-memory artifacts as required and sets up a listener for receiving order requests from ORS

Sets up a trading session start and end timer using the scheduled executor service to trigger the trading session startup and shutdown when the session start time and end time is reached for the associated ME instance id.

8.6. Trading Session Startup Process

Sends trading session start notification consisting of ME instance id and trading session id to all ORS instances.

8.7. Trading Session Shutdown Process

Sends trading session end notification consisting of ME instance id and trading session id to all ORS instances

Sends done for session updates to ORS for all the orders pending on the queue and removes them from the queue. Sends trading session end instruction to BE.

Initiates shutdown of the matching engine instance

8.8. Shutdown Process

The shutdown process checks for any active trading session and allows a shut down only if the active trading session has ended.

8.9. Trade Number Generation Logic

The trade number is a 15 character string structured as shown below and is guaranteed to be unique for a period of one year. The assumption here is that the trade generation rate per ME instance will not exceed 9999 per second. The initial 2 digits will be the ME Instance ID of the ME which receives the order and will range from 01-99.

The next 3 digits will be the day of the year which will range from 001-365.

The next 6 digits will be the time stamp when the order is received by the ORS. The format for the same would be hh:mm:ss and can range from 00:00:00 to 23:59:59.

The last 4 digits will be the sequence no which will range from 0001-9999 assuming not more than 9999 orders per second. When the counter reaches 9999, it is reset to 1.

9	9	3	6	5	2	3	5	9	5	9	9	9	9	9
ME Instance ID		Day of year			Time in hh:mm:ss format						Sequence No			
(01-99)		(001-365)			(00:00:00 – 23:59:59)						(0001 – 9999)			

8.10. PTP Matching Engine Algorithm

8.10.1. Check Request Type, Order Type and Side

Check the Request Type, OTC/NONOTC and side of the incoming Order.

If order type is new and NON OTC, go to step

If Request Type is Modification or cancellation go to step 8.10.2

8.10.2. Request Type Modify or Cancel

Check if order no exists in the queue the same side queue and if not reject the modification/cancellation request.

If yes and request type is cancellation, delete order from queue and send cancellation confirmation.

8.10.3. Check Queue

Check if there are any more pending orders in the same contract on the opposite side in the Order Queue; If pending order exists, sort the queue on price time (descending price - ascending time for buy queue, ascending price- descending time for sell queue) priority and take orders from top of the queue.

If no pending orders exist go to 8.10.7.

If Request type is OTC go to step 8.10.6.

8.10.4. Price Match Check

For Limit Order if Buy Order Price >= Sell Order Price, Price Match is true

For Market Order, Price Match is true at queue order price.

If match is true go to 8.10.5.

If match is false go to 8.10.7.

8.10.5. Quantity Condition

If (Incoming order qty = Queue Order Qty), then Quantity Condition is True.

If (Incoming order is AON and Queue Order is NON-AON and Incoming order qty < Queue Order Qty), then Quantity Condition is True.

If (Incoming order is non-AON and Queue Order is AON and Incoming order qty > Queue Order Qty), then Quantity Condition is True.

If (Incoming order is non-AON and Queue Order is non-AON), then Quantity Condition is True.

If quantity condition is true, put match quantity as minimum of (incoming order qty, queue order qty) and match price as queue order price and generate a trade.
Calculate and update balance qty of incoming order and queue order. If order qty less than disclosed qty, update disclosed qty with order qty.
If queue order qty is zero, delete queue order.
If incoming order balance quantity is zero go to 8.10.8.
If incoming order quantity is not zero proceed to 8.10.3.
If Quantity Condition is false, go to 8.10.7

8.10.6. Counterparty Condition

See if incoming order trading account and TM ids match with queue counterparty fields.
Check if Incoming order qty = Queue Order Qty
Check if incoming order price = queue order price for specific rate order. This check is always true for market order.
If all checks are true, put match quantity as incoming order qty, match price as incoming order LTP and generate a trade. Trigger execution report and delete queue order.
If any of the checks is false, go to 8.10.7

8.10.7. Action on Order

In case of new order request, if it's a non-otc market order, send order rejection.
If non-otc limit order, add to the same side queue.
In case of modification order request, delete original order from same side queue if there is a complete match, else replace original order with the modified order (unmatched quantity) on the same side queue.
Trigger quote update after adding to queue.
If it's an OTC order put in the queue.

8.10.8. Exit matching process

9. Auction Matching Engine

9.1. Overview

Auction ME is responsible for matching orders and bids based on the English, Yankee and Dutch matching algorithms and generating the corresponding trades.

9.2. Functions

9.2.1. Receiving Order Requests

Aucton engine receives auction order requests, auction order cancellation requests and bid requests. No modifications to auction orders nor to bids are allowed.

9.2.2. Sending Execution Reports

Auction Engine Sends following type of execution reports.

- Auction order acceptance
- Auction order rejection
- Auction order cancellation rejection
- Auction order cancellation confirmation
- Bid Acceptance
- Bid Rejection
- Bid knock-off
- Trades
- Auction Order Expiry
- Bid cancellations (due to auction order cancellation)

9.2.3. Auction Engine Processing

Auction engine maintains 2 logical books (queues) for each buy auction order i.e. Auction Buy Book and Bid Sell Book and it maintains 2 logical books (queues) for each sell auction order i.e. Auction Sell Book and Bid Buy Book.
Bid Buy Book is matched to Auction Sell Book; and Bid Sell Book is matched to Auction Buy Book.

9.2.4. Matching Priorities and Queue Depth

English auction priorities – Price (descending for buy bids and ascending for sell bids) and then ascending time.
Yankee and Dutch auction priorities – Price (descending for buy bids and ascending for sell bids) and then descending quantity and then ascending time.
For English auction, maintains a bid queue depth per auction order of not more than one; whereas for Yankee and Dutch auction, maintains a bid queue depth per auction order such that some non-zero quantity can be allocated to the bid.
The bid processing logic is discussed in detail in section 9.9.

9.3. Configuration

ME.conf
ME_INSTANCE_ID=01
ME_IP_ADDRESS=172.12.25.4
ME_PORT=2024
MATCHING_TYPE = AUCTION
TRADING_CALENDAR_ID = A101
DB_CONN_STRING = jdbc:oracle://172.12.25.10:3333/NSPOT_DB
IN_MEM_DB_CONN_STRING = <embedded database info>
DB_USERNAME = NSPOTUSR
DB_PASSWORD = NSPOTPW
IN_MEM_DB_USERNAME = NSPOTUSR
IN_MEM_DB_PASSWORD = NSPOTPW
TRADING_SESSION_MONITOR_INTERVAL = 30
AUCTION_END_MONITOR_INTERVAL = 30
NUM_EXTENSION_WINDOWS = 3
EXTENSION_WINDOW_DURATION = 45

9.4. In-memory artifacts

9.4.1. ME Map

ME Map is loaded from ME.conf on startup and consists of the following information. Trading session ID is loaded from persistent store.

- ME Instance ID
- IP Address
- Port
- Matching Type
- Trading Session ID
- Trading Session Monitor Interval
- Auction End Monitor Interval
- Number of Extension Windows
- Extension Window Duration
- Session Start Date and time
- Session End Date and time

9.4.2. Books

Auction Sell Book, Auction Buy Book, Bid Sell Book & **Bid Buy Book** structures are loaded on startup for each auction. Each queue consists of the following information:

- Order No
- Client Order No
- Original Client Order No
- Auction Type (for auction queues only)
- Side (Buy/Sell)
- Contract Code
- Price
- Quantity
- Allocated Qty
- Timestamp
- Auction Start Date Time (for auction queues only)
- Auction End Date Time (for auction queues only)
- Extensions (for auction queues only)
- Last Bid Timestamp (for auction queues only)
- Sort No (for bid queues only)

9.5. Startup Process

Sets up an executor service to handle concurrent execution of tasks

Sets up a scheduled executor service to handle scheduled tasks

Loads the in-memory artifacts.

Sets up a listener for receiving order requests from ORS

Sets up a the scheduled executor service to handle trading session start and end and auction start and end.

9.6. Scheduled Execution Service

Runs at the configured intervals and checks for trading session start and end as well auction start and end and starts the appropriate processes as below.

9.6.1. Trading Session Startup Process

If trading session start is due within the +/- timer interval, sends notification consisting of ME instance id and trading session id to all ORS instances.

9.6.2. Auction Start Process.

If any auction is due to start within +/- timer interval, sends auction side market quote to BE.

9.6.3. Auction Extention Process

Checks if any auction bidding period is due to end within + timer interval. Sleeps till bidding period is due to close. Checks if any bid has been accepted within extention window and extends bidding period end time by extention period. While extending auction period, also extends trading session end by 5 minutes more than last auction end if trading session end is prior to auction end. If no bid has been accepted within the extention window, initiates auction close process.

9.6.4. Auction Close Process

Checks for auction order quantity condition and allocated quantity of the auction order. If all quantity is allocated or quanity condition is not all or none generates trades for bids in the queue, cancels balance auction order quantity if any and deletes auction order and queue and sends trade updates to ORS and LTP updates to BE. Trade generation logic is as per section 9.9.

Otherwise cancels the auction order and all corresponding bids and sends cancellation messages to ORS.

9.6.5. Trading Session Shutdown Process

Sends trading session end notification consisting of ME instance id and trading session id to all ORS instances. Initiates shutdown of the matching engine instance provided no auction is active. If any auctions are active, those are cancelled first. The process also shuts down the engine.

9.7. Auction Order and Bid Validations

Following are the list of auction order validations:

9.7.1. Date Time Validations

For Auction orders, bidding start time must not be greater than (x+y+z) minutes before the end of the trading session where

x = max number of extension cycles * cycle duration

y = minimum bidding period

z = minimum interval between end of bidding period and end of session

For Auction orders, bidding end time must not be greater than (x+z) minutes before the end of the trading session where

x = max number of extension cycles * cycle duration

z = minimum interval between end of bidding period and end of session

9.7.2. Bid Validations

Following are the list of auction bid validations:

Auction order number referred to on the bid must exist on the auction book.

Side specified in the bid must be appropriate for the auction order.

Bid must have arrived between the auction start time and auction end time.

For forward auction bids, the bid price must be greater than the base price specified in the auction order.

For reverse auction bids, the bid price must be lesser than the ceiling price specified in the auction order.

9.7.3. Auction Cancellation Validation

Auction order must exist, bidding period should not be over and there should not have been any time extentions.

9.8. Trade Number Generation Logic

The trade number is a 15 character string structured as shown below and is guaranteed to be unique for a period of one year. The assumption here is that the trade generation rate per ME instance will not exceed 9999 per second.

The initial 2 digits will be the ME Instance ID of the ME which receives the order and will range from 01-99.

The next 3 digit will be the day of the year which will range from 001-365.

The next 6 digits will be the time stamp when the order is received by the ORS. The format for the same would be hh:mm:ss and can range from 00:00:00 to 23:59:59.

The last 4 digits will be the sequence no which will range from 0001-9999 assuming not more than 9999 orders per second. When the counter reaches 9999, it is reset to 1.

9	9	3	6	5	2	3	5	9	5	9	9	9	9	9
ME Instance ID		Day of year			Time in hh:mm:ss format						Sequence No			
(01-99)		(001-365)			(00:00:00 – 23:59:59)						(0001 – 9999)			

9.9. Auction Matching Engine Algorithm

9.9.1. Auction Order and Cancellation

Forward Auction order when received is put in AuctionSellBook. Reverse Auction order when received is put in AuctionBuyBook and quote update is sent to BE. On receipt and validation of auction order cancellation request, auction order and all corresponding bids are deleted from the books and cancellation message is sent to ORS and BE.

9.9.2. Check bid against Auction

If buy bid check select auctionsellbook else select auctionbuybook

Find bid auction no in auction order no, if not found go to ??

Check currentdatetime between auctionorder start and end if wrong go to ??

Check auction type and if English go to ??. Else go to ??.

9.9.3. English Matching

Match bid quantity with auction order quantity. If not equal go to ??
Pick bidbuybook (if bid side B) or bidsellbook (if bid side S). If no bid in queue add incoming bid in queue, update auction lastbidtimestamp and trigger quoteupdate. If bid in queue compare incoming bid price with queue bid price. If incoming price better (>bidbuybook, < bidsellbook) than queue price put incoming bid in queue, update auction lastbidtimestamp and knock-off old bid in queue. Trigger quoteupdate for incoming, trigger cancellation for knock-off bid. Go to ??

9.9.4. Checking for better bid

Pick bidbuybook (if bid side B) or bidsellbook (if bid side S). Check if a bid exists for the same bidder for the same auction number, if yes check if the incoming bid is better than the existing bid. If not reject the bid. If yes or no bid exists in queue insert bid in queue.
Better buy bid is defined as price*quantity higher than earlier bid and better sell bid is defined as (price less but same or more quantity) or (price same but more quantity).

9.9.5. Bid Sorting and Quantity Allocation

Sort the queue on price, quantity, time (descending price, descending quantity, ascending time for bidbuybook; ascending price, descending quanity, ascending time for bidsellbook)
Reorder queue as per sorting rule. Set bidallocqty to zero for all bids in queue. Select auction order from auction queue, read auction quantity, reset auction allocqty to zero. Loop once in bid queue from top to bottom, with following actions for each bid in queue,

- when auction allocqty=auctionqty exit loop
- bidqty > (auction qty – auction allocqty) update bidallocqty with (auction qty – auctionallocqty) and auction allocaty with auction allocqty+bidallocqty(updated) .
- if bidqty < (auctionqty – auctionallocqty) update bidallocqty with bidqty and auction allocaty with auction allocqty+bidallocqty(updated)
- if no more bids in queue, exit loop
- if more bids in queue, loop for the next bid

After looping, delete bids in queue where bidallocqty=0 and trigger bid knockoffs for deleted bids. If incoming bid has bidallocqty =0 send rejection message instead of knock-off. If bidallocqty >0, send bid acceptance message and send quote update to BE.

9.10. Auction Extensions

Auction ME uses three parameters to control the auction extension and closure process:

- Time Window (w)
- Extension Period (p)
- Number of extensions (n)

The timer process checks every *w* seconds for auctions supposed to end within the next w seconds. It checks, if difference between lastbidtimestamp and auctionenddatetime is less than *w;* if yes, it checks if extensions are < N, if yes it updates the auctionenddatetime with lastbidtimestamp+P.

9.11. Auction Closure Process

If no bid is received in the time window or if extentions have been exhausted, auction closure process is initiated. If no bid exists, an auction done for session notification is sent.

For English Auction, it creates a trade at bid price and quantity, triggers trade notification and clears the queues.

For Yankee Auction, it creates trades at each of the bid price and bid allocquantity remaining in the queue and triggers trade notifications. Sends expiry notification for (auctionqty – auctionallocqty) if not zero and clears the queues.

For Dutch Auction, it creates trades for each of the bids at the worst (lowest for buybids, highestfor sell bids) price in the queue and bidallocquantity remaining in the queue and triggers trade notifications. Sends expiry notification for auctionqty - auctionallocqty if not zero and clears the queues.

9.12. Implementation

AuctionMatchingEngine is the main class that controls the overall life-cycle of the PTP matching engine and is responsible for setting up, operating and shutting down of the various components of the engine.

AuctionMatchingEngine sets up an executor service for concurrent execution of tasks and a scheduled executor service for handling scheduled tasks.

TradingStartTimer is set up using the scheduled executor service to trigger the trading session startup.

TradingStopTimer is set up using the scheduled executor service to trigger the trading session shutdown process.

AuctionTimer is set up using the scheduled executor service to control the auction extension and closure process.

AuctionMEListener sets itself up as a CAJO server to listen to order requests from ORS

AuctionMEListener submits an **OrdReqProc** task to the executor service for handling each order request from the ORS

OrdReqProc processes the new/cancel order request as per the section **Auction Matching Engine Algorithm**

OrdReqProc sends entire market depth to BE whenever the queue is reordered

InMemObjLoader loads all the in-memory artifacts from configuration files and persistent store at ORS startup

10. Non-Functional Requirements

10.1. Supported Platforms

Linux or any Unix variant OS

Commodity or fault tolerant hardware

10.2. Network and Protocols

FIX 4.4 messaging

TCP/IP based messaging

MPLS and Internet connectivity support

10.3. Front End

Direct Client Support

Multiple CTCL Support

10.4. Performance Requirements

Base

Active contracts = 5000

Active users = 100000

Active TMs = 5000

Segments = 100

Matching engines – up to n, where $n > 5$

Peak

Concurrent users / session = 300

ORS per instance = 60 orders per second

ME per instance = 60 orders and 10 trades per second

BE per instance = 300 messages per second

Duration of peak = 30 seconds

Sustained

Concurrent users / sessions = 100

ORS per instance = 20 orders per second

ME per instance = 20 orders and 4 trades per second

BE per instance = 100 messages per second

Latency for average load

Edge component to edge component latency of less than 300 ms on > 99 % basis.

Latency for peak load

Edge component to edge component latency of less than 500 ms on > 95 % basis.

11. Annexure A – PTP Matching Scenarios

11.1. Limit Orders

11.1.1. Case 1: No orders on the other side. (first order)

Order Book for Contract ABC:

Buy Side			Sell Side		
Price	Qty	Terms	Price	Qty	Terms

Active Order:

Type	Qty	Price	Terms
Buy	1000	101	-

Result: The active order will not be matched as there are no orders in the order book. But the order will be placed in the order book as the first order on the buy side

11.1.2. Case 2: No match (Best Buy price < Best Sell Price)

Order Book for Contract ABC:

Buy Side			Sell Side		
Price	Qty	Terms	Price	Qty	Terms
100	1000	-	102	1000	-

Active Order:

Type	Qty	Price	Terms
Buy	1000	101	-

Result: The active order will not be matched as the Best Sell Price > Active buy Price. but will be queued up in the order book on the Buy side as the best buy order.

11.1.3. Case 3: Complete match against a single order. (Same Price , Different Qty) PQ>AQ

Order Book for Contract ABC:

Buy Side			Sell Side		
Price	Qty	Terms	Price	Qty	Terms
101	1000	-	102	1000	-
100	1000				

Active Order:

Type	Qty	Price	Terms
Sell	500	101	-

Result: Active order matches against best buy order of 1000 at 101 (against order #1). Trade takes place at 101 for 500 contracts.

11.1.4. Case 4: Complete match against a single order. (Different price, Diff Qty)

Order Book for Contract ABC:

Buy Side			Sell Side		
Price	Qty	Terms	Price	Qty	Terms
101	1000	-	102	1000	-
100	1000				

Active Order:

Type	Qty	Price	Terms
Sell	1000	98	-

Result: Active order matches against best buy order of 1000 at 101. Trade takes place at 101 for 1000 contracts.

11.1.5. Case 5: Complete match against multiple orders. (Same Price)

Order Book for Contract ABC:

Buy Side			Sell Side		
Price	Qty	Terms	Price	Qty	Terms
98	1000	-	102	1000	-
98	1000				

Active Order:

Type	Qty	Price	Terms
Sell	2000	98	-

Result: Active order matches against two buy orders. The first trade takes place at Qty =1000 Price =98, and the second trade takes place at Qty =1000, price =98

11.1.6. Case 6: Complete match against multiple orders. At diff price

Order Book for Contract ABC:

Buy Side			Sell Side		
Price	Qty	Terms	Price	Qty	Terms
101	1000	-	102	1000	-

100	1000				

Active Order:

Type	Qty	Price	Terms
Sell	2000	98	-

Result: Active order matches against two buy orders. The first trade takes place at Qty =1000 Price =101, and the second trade takes place at Qty =1000, price =100

11.1.7. Case 7: Complete match against Single order for an active AON orders. At diff price

Order Book for Contract ABC:

Buy Side			Sell Side		
Price	Qty	Terms	Price	Qty	Terms
101	2000	-	102	1000	-
100	1000				

Active Order:

Type	Qty	Price	Terms
Sell	2000	98	AON

Result: Active order matches against the best buy order. The trade takes place at Qty =1000 Price =101.

11.1.8. Case 8: No Match against the orders for an active AON Order

Order Book for Contract ABC:

Buy Side			Sell Side		
Price	Qty	Terms	Price	Qty	Terms
101	1000	-	102	1000	-
100	500				

Active Order:

Type	Qty	Price	Terms
Sell	2000	98	AON

Result: The active order will not be matched as the order qty available on the buy side is less than the active order qty. Hence, the order will be queued up in the order book as the best sell order.

11.1.9. Case 9: Complete match against single orders, Both the Orders Active as well as passive orders are AON terms orders

Order Book for Contract ABC:

Buy Side			Sell Side		
Price	Qty	Terms	Price	Qty	Terms
101	1000	AON	102	1000	-
100	1000				

Active Order:

Type	Qty	Price	Terms
Sell	1000	98	AON

Result: Active order matches against the best buy order. The trade takes place at Qty =1000 Price =101.

11.1.10. Case 10: Complete match against single orders, Both the Orders Active as well as passive orders are AON terms orders

Order Book for Contract ABC:

Buy Side			Sell Side		
Price	Qty	Terms	Price	Qty	Terms
101	1000	AON	102	1000	-
100	500	AON			

Active Order:

Type	Qty	Price	Terms
Sell	500	98	AON

Result: Active order matches against the Second best buy order. The trade takes place at Qty =500 Price =100.

11.1.11. Case 11: Partial match for an active Normal Order

Order Book for Contract ABC:

Buy Side			Sell Side		
Price	Qty	Terms	Price	Qty	Terms
101	1000	-	102	1000	-
100	1500	AON			

Active Order:

Type	Qty	Price	Terms
Sell	1500	98	-

Result: Active order partially matches against the best buy orders. The trade takes place at Qty =1000 Price =101, and the active order and Qty =500 @ Rs. 98 will be placed in the order book

11.1.12.Case 12: Partial match of a Passive order for an active AON Order

Order Book for Contract ABC:

Buy Side			Sell Side		
Price	Qty	Terms	Price	Qty	Terms
101	1500	-	102	1000	-
100	1500	AON			

Active Order:

Type	Qty	Price	Terms
Sell	1000	98	AON

Result: Active order will be fully matched against the best buy orders. The trade takes place at Qty =1000 Price =101, and the best buy order Qty will be updated to 500 @ Rs. 101 in the order book.

11.1.13.Case 13: Partial match against multiple orders. (at same price)

Order Book for Contract ABC:

Buy Side			Sell Side		
Price	Qty	Terms	Price	Qty	Terms
98	1000	-	102	1000	-

98	2000				

Active Order:

Type	Qty	Price	Terms
Sell	4000	98	-

Result: Active order will be partially matches against two buy orders. The first trade takes place at Qty =1000 Price =98, and the second trade will be for Qty =2000 @ Rs. 98. The active order qty will be updated to 1000 and will be placed in the buy side in the order book at Rs. 98.

11.1.14.Case 14: Partial match against multiple orders. (at diff price)

Order Book for Contract ABC:

Buy Side			Sell Side		
Price	Qty	Terms	Price	Qty	Terms
101	1000	-	102	1000	-
100	2000				

Active Order:

Type	Qty	Price	Terms
Sell	4000	98	-

Result: Active order will be partially matches against two buy orders. The first trade takes place at Qty =1000 Price =101, and the second trade will be for Qty =2000 @ Rs. 100. The active order qty will be updated to 1000 and will be placed in the buy side in the order book at Rs. 98.

11.2. Market Orders

In case of a market price order no price is specified. Instead, the system is instructed to find the best price. Following are some examples of how the market price is determined.

11.2.1. Case 1: Complete match against single orders when there are orders on the other side of the market.

Order Book for Contract ABC:

			Sell Side		
Price	Qty	Terms	Price	Qty	Terms
101	1000	-			
100	2000				

Active Order:

Type	Qty	Price	Terms
Sell	1000	MKT	IOC

Result: Active order matches against the best buy order. Trade takes place at 101 for 10000 contracts (against order #1).

11.2.2. Case 2: Complete match against multiple orders when there are orders on the other side of the market.

Order Book for Contract ABC:

Buy Side			Sell Side		
Price	Qty	Terms	Price	Qty	Terms
101	1000	-			
100	2000				

Active Order:

Type	Qty	Price	Terms
Sell	3000	MKT	IOC

Result: Active order matches against two buy orders. The first Trade takes place at 101 for 10000 contracts (against order #1). and the second trade takes place at 100 Qty =2000.

11.2.3. Case 3: Partial match against single orders when there are orders on the other side of the market.

Order Book for Contract ABC:

Buy Side			Sell Side		
Price	Qty	Terms	Price	Qty	Terms
101	1000	-			

Active Order:

Type	Qty	Price	Terms
Sell	1500	MKT	IOC

Result: Active order matches partially against the best buy order. Trade takes place at 101 for 10000 contracts (against order #1). The remaining 500 Qty will be cancelled.

11.2.4. Case 4: Complete match against multiple orders when there are orders on the other side of the market.

Order Book for Contract ABC:

Buy Side			Sell Side		
Price	Qty	Terms	Price	Qty	Terms

101	1000	-			
100	2000				

Active Order:

Type	Qty	Price	Terms
Sell	3500	MKT	IOC

Result: Active order matches partially against two buy orders. The first Trade takes place at 101 for 10000 contracts (against order #1). and the second trade takes place at 100 Qty =2000. The remaining Qty =500 will be cancelled.

11.2.5. Case 5: When there are no orders on the other side of the market.

Order Book for Contract ABC:

Buy Side			Sell Side		
Price	Qty	Terms	Price	Qty	Terms
101	1000	-			
100	2000				

Active Order:

Type	Qty	Price	Terms
Buy	3500	MKT	IOC

Result: Active order will not be matched as there are no orders on the opposite side. Therefore, the entire Qty of the active order will be cancelled.

11.3. Partially Disclosed Orders

Matching of metered qty will be similar to that of the limit orders, irrespective of the metered qty.

11.4. OTC Market Orders

11.4.1. Case 1: No match

Order Book for Contract ABC:

Buy Side				Sell Side			
TM Code	Counter party Code	Price	Qty	TM Code	Counter party Code	Price	Qty
A001	A002	MKT	15000				

Active Order:

Type	Qty	Price	TM code	Counter party code
Sell	3500	MKT	A003	A001

Result: The active order will not be traded as there is no counter order corresponding to the same. The active order will be instead be placed in the order book.

11.4.2. Case 2: Complete match (LTP exists)

Order Book for Contract ABC:

Buy Side	Sell Side

TM Code	Counter party Code	Price	Qty	TM Code	Counter party Code	Price	Qty
A001	A002	MKT	15000				

Active Order:

Type	Qty	Price	TM code	Counter party code
Sell	1500	MKT	A002	A001

LTP =100

Result: The active order will be traded against the pending order in the order book as the TM ID, and qty matches with the active order. Also since the LTP exists for the contract, the trade will be generated

11.4.3. Case 3: Complete match (LTP does not exists)

Order Book for Contract ABC:

Buy Side				Sell Side			
TM Code	Counter party Code	Price	Qty	TM Code	Counter party Code	Price	Qty
A001	A002	MKT	15000				

Active Order:

Type	Qty	Price	TM code	Counter party code

Sell	1500	MKT	A002	A001

Result: Though the active order fully matches with the counter party order, the trade will not be generated, as there is no LTP. However, the same will be queued up in the order book and will be matched at the end of the session at the closing price.

11.5. OTC specified rate orders

11.5.1. Case 1: No match

Order Book for Contract ABC:

Buy Side				Sell Side			
TM Code	Counter party Code	Price	Qty	TM Code	Counter party Code	Price	Qty
A001	A002	150	15000				

Active Order:

Type	Qty	Price	TM code	Counter party code
Sell	3500	150	A003	A001

Result: The active order will not be traded as there is no counter order corresponding to the same. The active order will be instead be placed in the order book.

11.5.2. Case 2: Complete match

Order Book for Contract ABC:

Buy Side				Sell Side			
TM Code	Counter party Code	Price	Qty	TM Code	Counter party Code	Price	Qty
A001	A002	150	15000				

Active Order:

Type	Qty	Price	TM code	Counter party code
Sell	1500	150	A002	A001

Result: The active order will be traded against the pending order in the order book as the TM ID, price and qty matches with the active order. Hence the trade will be generated for qty =1500 @rs.150.

12. Annexure B – Auction Matching Scenarios

12.1. English Forward Auction

12.1.1. Case 1: Full Match

Auction Order for Contract ABC:

Buy Side			Sell Side		
Price	Qty	Terms	Price	Qty	Terms
		-	100	1000	-

Auction Bid Book in Contract ABC:

Buy Side			Sell Side		
Price	Qty	Terms	Price	Qty	Terms
102	1000				-

Result: At the end of the session, the auction bid will be traded against the auction order at Rs. 102

12.1.2. Case 2: No Match

Auction Order for Contract ABC:

Buy Side			Sell Side		
Price	Qty	Terms	Price	Qty	Terms
100	1000	-			-

Auction Bid Book in Contract ABC:

Buy Side			Sell Side		
Price	Qty	Terms	Price	Qty	Terms
					-

Result: At the end of the session, there will not be any trades as no auction bids have been received.

12.1.3. Case 3: Better Bid Knocking off the earlier Bid

Auction Order for Contract ABC:

Buy Side			Sell Side		
Price	Qty	Terms	Price	Qty	Terms
		-	100	1000	-

Auction Bid Book in Contract ABC:

Buy Side			Sell Side		
Price	Qty	Terms	Price	Qty	Terms
102	1000				-

Incoming Bid:

Buy Side			Sell Side		
Price	Qty	Terms	Price	Qty	Terms
104	1000				-

Result: As the new bid is better than the existing bid, the new bid will be accepted and updated in the Auction Bid Book @ Rs. 104 and knocking off the earlier bid.

12.1.4. Case 4: Incoming bid which is not better than the previous one and hence knock off the same

Auction Order for Contract ABC:

Buy Side	Sell Side

Price	Qty	Terms	Price	Qty	Terms
		-	100	1000	-

Auction Bid Book in Contract ABC:

Buy Side			Sell Side		
Price	Qty	Terms	Price	Qty	Terms
102	1000				-

Incoming Bid:

Buy Side			Sell Side		
Price	Qty	Terms	Price	Qty	Terms
100	1000				-

Result: As the new bid is not better than the existing bid, the new bid will not be accepted and will be knocked off from the earlier bid.

12.2. English Reverse auction

12.2.1. Case 1: Full Match

Auction Order for Contract ABC:

Buy Side			Sell Side		
Price	Qty	Terms	Price	Qty	Terms
100	1000	-			-

Auction Bid Book in Contract ABC:

Buy Side			Sell Side		
Price	Qty	Terms	Price	Qty	Terms
			98	1000	-

Result: At the end of the session, the auction bid will be traded against the auction order at Rs. 98

12.2.2. Case2: No Match

Auction Order for Contract ABC:

Buy Side			Sell Side		
Price	Qty	Terms	Price	Qty	Terms
100	1000	-			-

Auction Bid Book in Contract ABC:

Buy Side			Sell Side		
Price	Qty	Terms	Price	Qty	Terms
					-

Result: At the end of the session, there will not be any trades as no auction bids have been received.

12.2.3. Case 3: Better Bid Knocking off the earlier Bid

Auction Order for Contract ABC:

Buy Side			Sell Side		
Price	Qty	Terms	Price	Qty	Terms
100	1000	-			-

Auction Bid Book in Contract ABC:

Buy Side			Sell Side		
Price	Qty	Terms	Price	Qty	Terms
			99	1000	-

Incoming Bid:

Buy Side			Sell Side		
Price	Qty	Terms	Price	Qty	Terms
			98	1000	-

Result: As the new bid is better than the existing bid, the new bid will be accepted and updated in the Auction Bid Book @ Rs. 98 and knocking off the earlier bid.

12.2.4. Case 4: Incoming bid which is not better than the previous one and hence knock off the same

Auction Order for Contract ABC:

Buy Side			Sell Side		
Price	Qty	Terms	Price	Qty	Terms
100	1000	-			-

Auction Bid Book in Contract ABC:

Buy Side			Sell Side		
Price	Qty	Terms	Price	Qty	Terms
			99	1000	-

Incoming Bid:

Buy Side			Sell Side		
Price	Qty	Terms	Price	Qty	Terms
			100	1000	-

Result: As the new bid is not better than the existing bid, the new bid will not be accepted and will be knocked off.

12.3. Yankee Forward Auction

12.3.1. Case 1: Full Match

Auction Order for Contract ABC:

Buy Side			Sell Side		
Price	Qty	Terms	Price	Qty	Terms
		-	98	1000	-

Auction Bid Book in Contract ABC:

Buy Side					
Price	Bid Qty	Allocated Qty	Traded Price	Cummulative Allocated Order Qty	Balance Order Qty
102	200	200	102	200	800-
101	200	200	101	400	600
99	400	400	100	800	200
99	200	200	100	1000	0

Result: At the end of the session, the following trades will be generated.

Trade Price	Trade Qty
102	200
101	200
99	200
99	400

12.3.2. Case 2: No Match

Auction Order for Contract ABC:

Buy Side			Sell Side		
Price	Qty	Terms	Price	Qty	Terms

		-	100	1000	-

Auction Bid Book in Contract ABC:

Buy Side					
Price	Bid Qty	Allocated Qty	Traded Price	Cummulative Allocated Order Qty	Balance Order Qty

Result: At the end of the session, no trades will be generated as there are no bids received.

12.3.3. Case 3: Partial match

Auction Order for Contract ABC:

Buy Side			Sell Side		
Price	Qty	Terms	Price	Qty	Terms
		-	100	1000	-

Auction Bid Book in Contract ABC:

Buy Side					
Price	Bid Qty	Allocated Qty	Traded Price	Cummulative Allocated Order Qty	Balance Order Qty
102	200	200	102	200	800-
101	200	200	101	400	600
99	400	400	100	800	200

Result: At the end of the session, the following trades will be generated. and 200 order Qty will be cancelled as there are no bids corresponding to the same.

Trade Price	Trade Qty
102	200
101	200

99	400

12.3.4. Case 4: Better Bid Knocking off the earlier Bid (Price)

Auction Order for Contract ABC:

Buy Side			Sell Side		
Price	Qty	Terms	Price	Qty	Terms
		-	98	1000	-

Auction Bid Book in Contract ABC:

Buy Side					
Price	Bid Qty	Allocated Qty	Traded Price	Cummulative Allocated Order Qty	Balance Order Qty
102	200	200	102	200	800-
101	200	200	101	400	600
99	400	400	100	800	200
99	200	200	100	1000	0

Incoming Bid:

Buy Side			Sell Side		
Price	Qty	Terms	Price	Qty	Terms
100	200				-

Result: Since the new Bid is better than the available bids, the Bid will be accepted and the Allocated Qty will be recomputed, thus knocking off the last bid.

New Bid Book:

Buy Side					
Price	Bid Qty	Allocated Qty	Traded Price	Cummulative Allocated Order Qty	Balance Order Qty

102	200	200	102	200	800
101	200	200	101	400	600
100	200	200	100	600	400
99	400	400	99	1000	0

12.3.5. Case 5: Better Bid Knocking off the earlier Bid (Qty)

Auction Order for Contract ABC:

Buy Side			Sell Side		
Price	Qty	Terms	Price	Qty	Terms
		-	98	1000	-

Auction Bid Book in Contract ABC:

Buy Side					
Price	Bid Qty	Allocated Qty	Traded Price	Cummulative Allocated Order Qty	Balance Order Qty
102	200	200	102	200	800-
101	200	200	101	400	600
99	400	400	100	800	200
99	200	200	100	1000	0

Incoming Bid:

Buy Side			Sell Side		
Price	Qty	Terms	Price	Qty	Terms
99	600				-

Result: Since the new Bid is better than the available bids, the Bid will be accepted and the Allocated Qty will be recomputed, thus knocking off the last two bid.

New Bid Book:

Buy Side					
Price	Bid Qty	Allocated Qty	Traded Price	Cummulative Allocated Order Qty	Balance Order Qty
102	200	200	102	200	800
101	200	200	101	400	600
99	600	600	100	1000	0

12.3.6. Case 6: Better Bid and change in the allocated Qty for each bid

Auction Order for Contract ABC:

Buy Side			Sell Side		
Price	Qty	Terms	Price	Qty	Terms
		-	98	1000	-

Buy Side					
Price	Bid Qty	Allocated Qty	Traded Price	Cummulative Allocated Order Qty	Balance Order Qty
102	200	200	102	200	800
101	200	200	101	400	600
100	200	200	100	600	400
99	400	400	99	1000	0

Incoming Bid:

Buy Side			Sell Side		
Price	Qty	Terms	Price	Qty	Terms
100	100				-

Result: Since the new Bid is better than the available last bid in the Queue, the Bid will be accepted and the Allocated Qty will be recomputed, thus resulting in reduction in allocated Qty for the last bid in the Queue.

New Bid Book:

Buy Side					
Price	Bid Qty	Allocated Qty	Traded Price	Cummulative Allocated Order Qty	Balance Order Qty
102	200	200	102	200	800
101	200	200	101	400	600
100	200	200	100	600	400
100	100	100	100	700	300
99	300	300	99	1000	0

12.3.7. Case 7: Incoming bid which is not better than the previous one and hence knock off

Auction Order for Contract ABC:

Buy Side			Sell Side		
Price	Qty	Terms	Price	Qty	Terms
		-	98	1000	-

Buy Side					
Price	Bid Qty	Allocated Qty	Traded Price	Cummulative Allocated Order Qty	Balance Order Qty
102	200	200	102	200	800
101	200	200	101	400	600
100	200	200	100	600	400

99	400	400	99	1000	0

Incoming Bid:

Buy Side			Sell Side		
Price	Qty	Terms	Price	Qty	Terms
99	200				-

Result: Since the new Bid is not better than the available last bid in the Queue, the Bid will be not be accepted and hence will be knocked off.

There will be no change in the Bid Book

12.3.8. Case 8: Incoming bid which is not better than the previous one, but there exists pending Qty to be allocated

Auction Order for Contract ABC:

Buy Side			Sell Side		
Price	Qty	Terms	Price	Qty	Terms
		-	98	1000	-

Buy Side					
Price	Bid Qty	Allocated Qty	Traded Price	Cummulative Allocated Order Qty	Balance Order Qty
102	200	200	102	200	800
101	200	200	101	400	600
100	200	200	100	600	400
99	100	100	99	700	300

Incoming Bid:

Buy Side			Sell Side		
Price	Qty	Terms	Price	Qty	Terms
99	100				-

Result: Since there is pending Qty to be allocated, the new Bid will be accepted and updated, in the bid Book.

New Bid Book:

Buy Side					
Price	Bid Qty	Allocated Qty	Traded Price	Cummulative Allocated Order Qty	Balance Order Qty
102	200	200	102	200	800
101	200	200	101	400	600
100	200	200	100	600	400
99	100	100	99	700	300
99	100	100	99	800	200

12.4. Yankee Reverse Auction

12.4.1. Case 1: Full Match

Auction Order for Contract ABC:

Buy Side			Sell Side		
Price	Qty	Terms	Price	Qty	Terms
100	1000	-			-

Bid Book:

Sell Side					
Price	Bid Qty	Allocated Qty	Traded Price	Cummulative Allocated Order Qty	Balance Order Qty
97	400	400	97	400	600
98	200	200	98	600	400
99	200	200	99	800	200
100	200	200	100	1000	0

Result: At the end of the session, the following trades will be generated.

Trade Price	Trade Qty
97	400
98	200
99	200
100	200

12.4.2. Case 2: No Match

Auction Order for Contract ABC:

Buy Side			Sell Side		
Price	Qty	Terms	Price	Qty	Terms
100	1000	-			-

Bid Book:

Sell Side					
Price	Bid Qty	Allocated Qty	Traded Price	Cummulative Allocated Order Qty	Balance Order Qty

Result: At the end of the session, no trades will be generated as there are no bids received.

12.4.3. Case 3: Partial match

Auction Order for Contract ABC:

Buy Side			Sell Side		
Price	Qty	Terms	Price	Qty	Terms
100	1000	-			-

Auction Bid Book in Contract ABC:

Sell Side					
Price	Bid Qty	Allocated Qty	Traded Price	Cummulative Allocated Order Qty	Balance Order Qty
97	400	400	97	400	600
98	200	200	98	600	400
99	200	200	99	800	200

Result: At the end of the session, the following trades will be generated. and 400 order Qty will be cancelled as there are no bids corresponding to the same.

Trade Price	Trade Qty
97	400
98	200
99	200

12.4.4. Case 4: Better Bid Knocking off the earlier Bid (Price)

Auction Order for Contract ABC:

Buy Side	Sell Side

Price	Qty	Terms	Price	Qty	Terms
100	1000	-			-

Auction Bid Book in Contract ABC:

Sell Side					
Price	Bid Qty	Allocated Qty	Traded Price	Cummulative Allocated Order Qty	Balance Order Qty
97	200	200	97	200	800-
98	200	200	98	400	600
99	400	400	99	800	200
99	200	200	99	1000	0

Incoming Bid:

Buy Side			Sell Side		
Price	Qty	Terms	Price	Qty	Terms
			98	200	-

Result: Since the new Bid is better than the available bids, the Bid will be accepted and the Allocated Qty will be recomputed, thus knocking off the last bid.

New Bid Book:

Sell Side					
Price	Bid Qty	Allocated Qty	Traded Price	Cummulative Allocated Order Qty	Balance Order Qty
97	200	200	97	200	800
98	200	200	98	400	600
98	200	200	98	600	400
99	400	400	99	1000	0

12.4.5. Case 5: Better Bid Knocking off the earlier Bid (Qty)

Auction Order for Contract ABC:

Buy Side			Sell Side		
Price	Qty	Terms	Price	Qty	Terms
100	1000	-			-

Auction Bid Book in Contract ABC:

Sell Side					
Price	Bid Qty	Allocated Qty	Traded Price	Cummulative Allocated Order Qty	Balance Order Qty
97	200	200	97	200	800
98	200	200	98	400	600
99	400	400	99	800	200
99	100	200	99	1000	100

Incoming Bid:

Buy Side			Sell Side		
Price	Qty	Terms	Price	Qty	Terms
			99	200	-

Result: Since the new Bid is better than the available bids, the Bid will be accepted and the Allocated Qty will be recomputed, thus knocking off the last two bid.

New Bid Book:

Buy Side					
Price	Bid Qty	Allocated Qty	Traded Price	Cummulative Allocated Order Qty	Balance Order Qty
97	200	200	97	200	800
98	200	200	98	400	600
99	400	600	99	800	200

99	200	200	99	1000	-

12.4.6. Case 6: Better Bid and change in the allocated Qty for each bid

Auction Order for Contract ABC:

Buy Side			Sell Side		
Price	Qty	Terms	Price	Qty	Terms
100	1000	-			-

Bid Book

Sell Side					
Price	Bid Qty	Allocated Qty	Traded Price	Cummulative Allocated Order Qty	Balance Order Qty
97	200	200	102	200	800
98	200	200	101	400	600
99	400	400	100	800	200

Incoming Bid:

Buy Side			Sell Side		
Price	Qty	Terms	Price	Qty	Terms
			98	300	-

Result: Since the new Bid is better than the available last bid in the Queue, the Bid will be accepted and the Allocated Qty will be recomputed, thus resulting in reduction in allocated Qty for the last bid in the Queue.

New Bid Book:

Buy Side					
Price	Bid Qty	Allocated Qty	Traded Price	Cummulative Allocated Order Qty	Balance Order Qty

97	200	200	97	200	800
98	200	200	98	400	600
98	300	300	98	700	300
99	400	300	99	1000	0

12.4.7. Case 7: Incoming bid which is not better than the previous one and hence knock off

Auction Order for Contract ABC:

Buy Side			Sell Side		
Price	Qty	Terms	Price	Qty	Terms
98	1000	-			-

Bid Book

Sell Side					
Price	Bid Qty	Allocated Qty	Traded Price	Cummulative Allocated Order Qty	Balance Order Qty
97	200	200	97	200	800
98	200	200	98	400	600
98	300	300	98	700	300
99	400	300	99	1000	0

Incoming Bid:

Buy Side			Sell Side		
Price	Qty	Terms	Price	Qty	Terms
			99	100	-

Result: Since the new Bid is not better than the available last bid in the Queue, the Bid will be not be accepted and hence will be knocked off.

There will be no change in the Bid Book

12.4.8. Case 8: Incoming bid which is not better than the previous one, but there exists pending Qty to be allocated

Auction Order for Contract ABC:

Buy Side			Sell Side		
Price	Qty	Terms	Price	Qty	Terms
100	1000	-			-

Bid Book

Sell Side					
Price	Bid Qty	Allocated Qty	Traded Price	Cummulative Allocated Order Qty	Balance Order Qty
97	200	200	102	200	800
98	200	200	101	400	600
99	400	400	100	800	200

Incoming Bid:

Buy Side			Sell Side		
Price	Qty	Terms	Price	Qty	Terms
99	200				-

Result: Since there is pending Qty to be allocated, the new Bid will be accepted and updated, in the bid Book.

New Bid Book:

Sell Side					
Price	Bid Qty	Allocated Qty	Traded Price	Cummulative Allocated Order Qty	Balance Order Qty
97	200	200	102	200	800
98	200	200	101	400	600
99	400	400	100	800	200
99	200	200	99	1000	0

12.5. Dutch Forward Auction

12.5.1. Case 1: Full Match

Auction Order for Contract ABC:

Buy Side			Sell Side		
Price	Qty	Terms	Price	Qty	Terms
		-	98	1000	-

Auction Bid Book in Contract ABC:

Buy Side					
Price	Bid Qty	Allocated Qty	Traded Price	Cummulative Allocated Order Qty	Balance Order Qty
102	200	200	99	200	800-
101	200	200	99	400	600
99	400	400	99	800	200
99	200	200	99	1000	0

Result: At the end of the session, the following trades will be generated.

Trade Price	Trade Qty
99	200
99	200
99	200
99	400

12.5.2. Case 2: No Match

Auction Order for Contract ABC:

Buy Side			Sell Side		
Price	Qty	Terms	Price	Qty	Terms
		-	100	1000	-

Auction Bid Book in Contract ABC:

Buy Side					
Price	Bid Qty	Allocated Qty	Traded Price	Cummulative Allocated Order Qty	Balance Order Qty

Result: At the end of the session, no trades will be generated as there are no bids received.

12.5.3. Case 3: Partial match

Auction Order for Contract ABC:

Buy Side			Sell Side		
Price	Qty	Terms	Price	Qty	Terms
		-	100	1000	-

Auction Bid Book in Contract ABC:

Buy Side					
Price	Bid Qty	Allocated	Traded Price	Cummulative	Balance

		Qty		Allocated Order Qty	Order Qty
102	200	200	99	200	800-
101	200	200	99	400	600
99	400	400	99	800	200

Result: At the end of the session, the following trades will be generated. and 200 order Qty will be cancelled as there are no bids corresponding to the same.

Trade Price	Trade Qty
99	200
99	200
99	400

12.5.4. Case 4: Better Bid Knocking off the earlier Bid (Price)

Auction Order for Contract ABC:

Buy Side			Sell Side		
Price	Qty	Terms	Price	Qty	Terms
		-	98	1000	-

Auction Bid Book in Contract ABC:

Buy Side					
Price	Bid Qty	Allocated Qty	Traded Price	Cummulative Allocated Order Qty	Balance Order Qty
102	200	200	99	200	800-
101	200	200	99	400	600
99	400	400	99	800	200
99	200	200	99	1000	0

Incoming Bid:

Buy Side			Sell Side		
Price	Qty	Terms	Price	Qty	Terms
100	200				-

Result: Since the new Bid is better than the available bids, the Bid will be accepted and the Allocated Qty will be recomputed, thus knocking off the last bid.

New Bid Book:

Buy Side					
Price	Bid Qty	Allocated Qty	Traded Price	Cummulative Allocated Order Qty	Balance Order Qty
102	200	200	99	200	800
101	200	200	99	400	600
100	200	200	99	600	400
99	400	400	99	1000	0

12.5.5. Case 5: Better Bid Knocking off the earlier Bid (Qty)

Auction Order for Contract ABC:

Buy Side			Sell Side		
Price	Qty	Terms	Price	Qty	Terms
		-	98	1000	-

Auction Bid Book in Contract ABC:

Buy Side					
Price	Bid Qty	Allocated Qty	Traded Price	Cummulative Allocated Order Qty	Balance Order Qty
102	200	200	99	200	800-
101	200	200	99	400	600

99	400	400	99	800	200
99	200	200	99	1000	0

Incoming Bid:

Buy Side			Sell Side		
Price	Qty	Terms	Price	Qty	Terms
99	600				-

Result: Since the new Bid is better than the available bids, the Bid will be accepted and the Allocated Qty will be recomputed, thus knocking off the last two bid.

New Bid Book:

Buy Side					
Price	Bid Qty	Allocated Qty	Traded Price	Cummulative Allocated Order Qty	Balance Order Qty
102	200	200	99	200	800
101	200	200	99	400	600
99	600	600	99	1000	0

12.5.6. Case 6: Better Bid and change in the allocated Qty for each bid

Auction Order for Contract ABC:

Buy Side			Sell Side		
Price	Qty	Terms	Price	Qty	Terms
		-	98	1000	-

Buy Side					
Price	Bid Qty	Allocated Qty	Traded Price	Cummulative Allocated	Balance Order Qty

				Order Qty	
102	200	200	99	200	800
101	200	200	99	400	600
100	200	200	99	600	400
99	400	400	99	1000	0

Incoming Bid:

Buy Side			Sell Side		
Price	Qty	Terms	Price	Qty	Terms
100	100				-

Result: Since the new Bid is better than the available last bid in the Queue, the Bid will be accepted and the Allocated Qty will be recomputed, thus resulting in reduction in allocated Qty for the last bid in the Queue.

New Bid Book:

Buy Side					
Price	Bid Qty	Allocated Qty	Traded Price	Cummulative Allocated Order Qty	Balance Order Qty
102	200	200	99	200	800
101	200	200	99	400	600
100	200	200	99	600	400
100	100	100	99	700	300
99	300	300	99	1000	0

12.5.7. Case 7: Incoming bid which is not better than the previous one and hence knock off

Auction Order for Contract ABC:

Buy Side			Sell Side		
Price	Qty	Terms	Price	Qty	Terms
		-	98	1000	-

Buy Side					
Price	Bid Qty	Allocated Qty	Traded Price	Cummulative Allocated Order Qty	Balance Order Qty
102	200	200	99	200	800
101	200	200	99	400	600
100	200	200	99	600	400
99	400	400	99	1000	0

Incoming Bid:

Buy Side			Sell Side		
Price	Qty	Terms	Price	Qty	Terms
99	200				-

Result: Since the new Bid is not better than the available last bid in the Queue, the Bid will be not be accepted and hence will be knocked off.

There will be no change in the Bid Book

12.5.8. Case 8: Incoming bid which is not better than the previous one, but there exists pending Qty to be allocated

Auction Order for Contract ABC:

Buy Side			Sell Side		
Price	Qty	Terms	Price	Qty	Terms
		-	98	1000	-

Buy Side					
Price	Bid Qty	Allocated Qty	Traded Price	Cummulative Allocated Order Qty	Balance Order Qty
102	200	200	99	200	800
101	200	200	99	400	600
100	200	200	99	600	400
99	100	100	99	700	300

Incoming Bid:

Buy Side			Sell Side		
Price	Qty	Terms	Price	Qty	Terms
99	100				-

Result: Since there is pending Qty to be allocated, the new Bid will be accepted and updated, in the bid Book.

New Bid Book:

Buy Side					
Price	Bid Qty	Allocated Qty	Traded Price	Cummulative Allocated Order Qty	Balance Order Qty
102	200	200	99	200	800
101	200	200	99	400	600
100	200	200	99	600	400
99	100	100	99	700	300
99	100	100	99	800	200

12.6. Dutch Reverse Auction

12.6.1. Case 1: Full Match

Auction Order for Contract ABC:

Buy Side			Sell Side		
Price	Qty	Terms	Price	Qty	Terms
100	1000	-			-

Bid Book:

Sell Side					
Price	Bid Qty	Allocated Qty	Traded Price	Cummulative Allocated Order Qty	Balance Order Qty
97	400	400	100	400	600
98	200	200	100	600	400
99	200	200	100	800	200
100	200	200	100	1000	0

Result: At the end of the session, the following trades will be generated.

Trade Price	Trade Qty
100	400
100	200
100	200
100	200

12.6.2. Case 2: No Match

Auction Order for Contract ABC:

Buy Side			Sell Side		
Price	Qty	Terms	Price	Qty	Terms
100	1000	-			-

Bid Book:

Sell Side					
Price	Bid Qty	Allocated Qty	Traded Price	Cummulative Allocated Order Qty	Balance Order Qty

Result: At the end of the session, no trades will be generated as there are no bids received.

12.6.3. Case 3: Partial match

Auction Order for Contract ABC:

Buy Side			Sell Side		
Price	Qty	Terms	Price	Qty	Terms
100	1000	-			-

Auction Bid Book in Contract ABC:

Sell Side					
Price	Bid Qty	Allocated Qty	Traded Price	Cummulative Allocated Order Qty	Balance Order Qty
97	400	400	99	400	600
98	200	200	99	600	400
99	200	200	99	800	200

Result: At the end of the session, the following trades will be generated. and 400 order Qty will be cancelled as there are no bids corresponding to the same.

Trade Price	Trade Qty
99	400
99	200
99	200

12.6.4. Case 4: Better Bid Knocking off the earlier Bid (Price)

Auction Order for Contract ABC:

Buy Side			Sell Side		
Price	Qty	Terms	Price	Qty	Terms
100	1000	-			-

Auction Bid Book in Contract ABC:

Sell Side					
Price	Bid Qty	Allocated Qty	Traded Price	Cummulative Allocated Order Qty	Balance Order Qty
97	200	200	99	200	800-
98	200	200	99	400	600
99	400	400	99	800	200
99	200	200	99	1000	0

Incoming Bid:

Buy Side			Sell Side		
Price	Qty	Terms	Price	Qty	Terms
			98	200	-

Result: Since the new Bid is better than the available bids, the Bid will be accepted and the Allocated Qty will be recomputed, thus knocking off the last bid.

New Bid Book:

Sell Side					
Price	Bid Qty	Allocated Qty	Traded Price	Cummulative Allocated Order Qty	Balance Order Qty
97	200	200	99	200	800
98	200	200	99	400	600
98	200	200	99	600	400
99	400	400	99	1000	0

12.6.5. Case 5: Better Bid Knocking off the earlier Bid (Qty)

Auction Order for Contract ABC:

Buy Side			Sell Side		
Price	Qty	Terms	Price	Qty	Terms
100	1000	-			-

Auction Bid Book in Contract ABC:

Sell Side					
Price	Bid Qty	Allocated Qty	Traded Price	Cummulative Allocated Order Qty	Balance Order Qty
97	200	200	99	200	800
98	200	200	99	400	600
99	400	400	99	800	200
99	100	200	99	1000	100

Incoming Bid:

Buy Side			Sell Side		
Price	Qty	Terms	Price	Qty	Terms
			99	200	-

Result: Since the new Bid is better than the available bids, the Bid will be accepted and the Allocated Qty will be recomputed, thus knocking off the last two bid.

New Bid Book:

Buy Side					
Price	Bid Qty	Allocated Qty	Traded Price	Cummulative Allocated Order Qty	Balance Order Qty
97	200	200	99	200	800
98	200	200	99	400	600
99	400	600	99	800	200
99	200	200	99	1000	-

12.6.6. Case 6: Better Bid and change in the allocated Qty for each bid

Auction Order for Contract ABC:

Buy Side			Sell Side		
Price	Qty	Terms	Price	Qty	Terms
100	1000	-			-

Bid Book

Sell Side					
Price	Bid Qty	Allocated Qty	Traded Price	Cummulative Allocated Order Qty	Balance Order Qty
97	200	200	99	200	800
98	200	200	99	400	600
99	400	400	99	800	200

Incoming Bid:

Buy Side			Sell Side		
Price	Qty	Terms	Price	Qty	Terms

			98	300	-

Result: Since the new Bid is better than the available last bid in the Queue, the Bid will be accepted and the Allocated Qty will be recomputed, thus resulting in reduction in allocated Qty for the last bid in the Queue.

New Bid Book:

Buy Side					
Price	Bid Qty	Allocated Qty	Traded Price	Cummulative Allocated Order Qty	Balance Order Qty
97	200	200	99	200	800
98	200	200	99	400	600
98	300	300	99	700	300
99	400	300	99	1000	0

12.6.7. Case 7: Incoming bid which is not better than the previous one and hence knock off

Auction Order for Contract ABC:

Buy Side			Sell Side		
Price	Qty	Terms	Price	Qty	Terms
98	1000	-			-

Bid Book

Sell Side					
Price	Bid Qty	Allocated Qty	Traded Price	Cummulative Allocated Order Qty	Balance Order Qty
97	200	200	99	200	800

98	200	200	99	400	600
98	300	300	99	700	300
99	400	300	99	1000	0

Incoming Bid:

Buy Side			Sell Side		
Price	Qty	Terms	Price	Qty	Terms
			99	100	-

Result: Since the new Bid is not better than the available last bid in the Queue, the Bid will be not be accepted and hence will be knocked off.

There will be no change in the Bid Book

12.6.8. Case 8: Incoming bid which is not better than the previous one, but there exists pending Qty to be allocated

Auction Order for Contract ABC:

Buy Side			Sell Side		
Price	Qty	Terms	Price	Qty	Terms
100	1000	-			-

Bid Book

Sell Side					
Price	Bid Qty	Allocated Qty	Traded Price	Cummulative Allocated Order Qty	Balance Order Qty
97	200	200	99	200	800
98	200	200	99	400	600
99	400	400	99	800	200

Incoming Bid:

Buy Side			Sell Side		
Price	Qty	Terms	Price	Qty	Terms
99	200				-

Result: Since there is pending Qty to be allocated, the new Bid will be accepted and updated, in the bid Book.

New Bid Book:

Sell Side					
Price	Bid Qty	Allocated Qty	Traded Price	Cummulative Allocated Order Qty	Balance Order Qty
97	200	200	99	200	800
98	200	200	99	400	600
99	400	400	99	800	200
99	200	200	99	1000	0