

Yamex

Technical & Functional specifications

Table of content

1. Welcome to Yamex!	1
2. Terminology	2
2.1. Design Considerations	2
2.1.1. Price/Time Algorithm	2
3. Overview	3
3.1. Consolidated tag views	3
4. Limit Order	4
4.1. Place a Buy Limit Order	4
4.2. Place a Sell Limit Order	4
4.3. Default Sell Limit Order	4
4.4. Time in Force - Accepted parameter	4
4.4.1. Examples	5
5. Market Order	6
5.1. Place a Sell Market order	6
5.2. Place a Buy Market order	6
6. Stop Order	7
6.1. Place a Buy stop order	7
6.2. Place a Sell stop order	7
7. Default order placed	8
7.1. Default Order	8
8. Order Book Best Prices	9
8.1. Best bid price - limit orders only	9
8.2. Best bid price - limit orders and market orders	9
8.3. Best ask price - limit orders only	9
8.4. Best ask price - limit orders and market orders	9
8.5. Best ask price - limit orders and stop orders	10
9. Order Book Accumulative Quantities	11
9.1. Cumulative Totals of Bids and Offers	11
10. Matching Principles for limit orders	12
10.1. Matching a Buy order partially - exact same price	12
10.2. Matching a Buy order partially - higher buy price	12
10.3. Matching a Sell order partially - exact same price	13
10.4. Matching a Sell order against multiple Buy orders - higher buy price fulfilled first	13
10.5. Matching a Sell order against multiple Buy orders - buys fulfilled in fifo	14
11. Matching Principles for market orders	15
11.1. Matching a new Market Order to Buy against existing multiple sell limit orders	15
11.2. Matching a new Limit Order to Buy against existing multiple sell orders market and limit	15
12. Control invariants	16
12.1. High volume of orders	16
Sample Steps	17

1. Welcome to Yamex!



Yamex is the next-gen market exchange engine that will probably drives all EURO, AMER and ASIA trading platform.

A central part of a market exchange engine is the matching engine. A matching engine is a program that accepts orders from buyers and sellers. The other matching module matches buy and sell orders, creates transactions to record the process, and updates the customers account balances.

The matching (or trade allocation) algorithm is an important part of an exchange trading mechanism, since it is responsible for resolving the buy/sell association in a fast and efficient way.

Exchanges set the institutional rules that govern trading and information flows about that trading. They are closely linked to the clearing facilities through which post-trade activities could be completed. An exchange centralizes the communication of bid and offer prices to all direct market participants, who can respond by selling or buying at one of the quotes or by replying with a different quote.

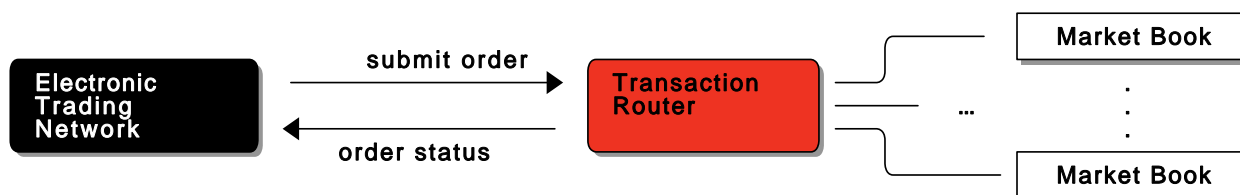


This specification describes the matching engine expected behaviors.

2. Terminology

- **Order** - An order is an instruction to buy or sell a stock at a specific price or better
- **Bid** - the price in a buy order
- **Ask** - the price in a sell order
- **Spread** - the difference between the bid and the ask
- **Time In Force** - indicates how long an order will remain active (see Appendix B)

2.1. Design Considerations



2.1.1. Price/Time Algorithm

Under the Price/Time algorithm, an incoming trade is matched with the first orders (at the best price), which is the so called FIFO method. More precisely: suppose n limit orders $Q_1 \dots Q_n$ of a cumulative volume Q lots, and an incoming trade of $N \leq Q$ lots.

Then there exists an index $1 \leq j \leq n$ such that:

$$\sum_{i=1}^j Q_i \leq N$$

and

$$\sum_{i=1}^{\min(j+1, n)} Q_i \geq N$$

Under the Price/Time algorithm, the first j limit orders are filled in full, and the remaining lots are assigned to the $(j+1)$ -th limit order (if $j < n$)

3. Overview

3.1. Consolidated tag views

Tag/Description	Passed	Failed	Other
@bestPrices - Order book query to retrieve actual best prices (best bid or ask)	5/5	0/5	0/5
@cumulativeView - Order book cumulative view	1/1	0/1	0/1
@default - Define standard and predefined objects with basic characteristics (e.g. limit order...)	3/3	0/3	0/3
@limitOrder - Limit Order	12/16	0/16	4/16
@marketOrder - Market Order	4/4	0/4	0/4
@matchingPrinciple - Order book's Matching principles	7/7	0/7	0/7
@notImplemented - Behavior not implemented	0/0	0/0	0/0
@orderBook - Order Book	14/14	0/14	0/14
@placeOrder - Order placed in order book	6/6	0/6	0/6
@stopOrder - Stop 'Loss' Order	3/3	0/3	0/3
@timeInForce - Time In Force	0/4	0/4	4/4

Tag/Description	Passed	Failed	Other
Orders (non wip)	18/22	0/22	4/22
Matching Principles	7/7	0/7	0/7

4. Limit Order

Uri: [yamex/feature/order/01-orders/01-limit-order.feature](#)

A limit order is an order to buy or sell a contract at a specific price or better.

A **buy** limit order can only be executed at the **limit price or lower**, and a **sell** limit order can only be executed at the **limit price or higher**.

Use of a Limit order helps ensure that the customer will not receive an execution at a price less favorable than the limit price. Use of a Limit order, however, does not guarantee an execution.

- A buy limit order for FFLY at \$125 will buy shares of FFLY at \$125 or less.
- A sell limit order for FFLY at \$125 will sell shares of FFLY for \$125 or more.
- A limit order must have a Time in Force (TIF) value



In our system we support Limit Orders with various time in force parameters. Fill-or-Kill (FoK), Immediate-or-Cancel (IoC) works the same way as for Market Orders. Good-Till Date or Good-Till-Cancel and Day Orders are valid until specified time requested by investor and cancelled after that.

4.1. Place a Buy Limit Order

Tags: [@limitOrder](#), [@placeOrder](#)

- ✓ Given an empty order book
- ✓ When a limit order is placed to buy 150 FFLY at 10.4€
- ✓ Then the order book should be updated with this new order

4.2. Place a Sell Limit Order

Tags: [@limitOrder](#), [@placeOrder](#)

- ✓ Given an empty order book
- ✓ When a limit order is placed to sell 150 FFLY at 10.4€
- ✓ Then the order book should be updated with this new order

4.3. Default Sell Limit Order

Tags: [@default](#), [@limitOrder](#)

- ✓ Given a default limit order to sell
- ✓ Then the order should have the following properties:

<i>order type</i>	<i>way</i>	<i>instrument</i>	<i>qty</i>	<i>price</i>	<i>time in force</i>
Limit Order	Sell	FFLY	100	10.1	none

4.4. Time in Force - Accepted parameter

Tags: @limitOrder, @timeInForce

When one try to place a default limit order with the following specifics:

<i>way</i>	<i>time in force</i>
sell	<time-in-force>

Then the order book should be updated with this new order

4.4.1. Examples

time-in-force

? fok

When one try to place a default limit order with the following specifics:

? iok

When one try to place a default limit order with the following specifics:

? gtd

When one try to place a default limit order with the following specifics:

? gtc

When one try to place a default limit order with the following specifics:

5. Market Order

Uri: [yamex/feature/order/01-orders/02-market-order.feature](#)

A market order is an order to buy or sell a contract at the best available price. Generally, this type of order will be executed immediately. However, the price at which a market order will be executed is not guaranteed. It is important for traders to remember that the last-traded price is not necessarily the price at which a market order will be executed. In fast-moving markets, the price at which a market order will execute often deviates from the last-traded price or “real time” quote.

Market Orders can have Fill-or-Kill (FoK) and Immediate or Cancel (IoC) parameters. The first one means that orders either will be filled completely or cancelled in case of insufficient liquidity. IoC order by contrast will be filled up to the existing level of liquidity and the resting part will be cancelled.

5.1. Place a Sell Market order

Tags: @marketOrder, @placeOrder

- ✓ Given an empty order book
- ✓ When a market order is placed to sell 150 FFLY
- ✓ Then the order book should be updated with this new order

5.2. Place a Buy Market order

Tags: @marketOrder, @placeOrder

- ✓ Given an empty order book
- ✓ When a market order is placed to buy 150 FFLY
- ✓ Then the order book should be updated with this new order

6. Stop Order

Uri: [yamex/feature/order/01-orders/03-stop-order.feature](#)

A stop order, also referred to as a stop-loss order, is an order to buy or sell a stock once the price of the stock reaches a specified price, known as the stop price. When the stop price is reached, a stop order becomes a market order. A Stop order is not guaranteed a specific execution price and may execute significantly away from its stop price.

A buy stop order is entered at a stop price above the current market price. Investors **generally use a buy stop order to limit a loss or to protect a profit** on a stock that they have sold short.

A sell stop order is entered at a stop price below the current market price. Investors generally use a sell stop order to limit a loss or to protect a profit on a stock that they own.

6.1. Place a Buy stop order

Tags: [@stopOrder](#), [@placeOrder](#)

- ✓ Given an empty order book
- ✓ When a stop order is placed to buy 150 FFLY at 10.4€
- ✓ Then the order book should be updated with this new order

6.2. Place a Sell stop order

Tags: [@stopOrder](#), [@placeOrder](#)

- ✓ Given an empty order book
- ✓ When a stop order is placed to sell 150 FFLY at 10.4€
- ✓ Then the order book should be updated with this new order

7. Default order placed

Uri: [yamex/feature/order/02-orderbook-queries/00-orderbook-defaults.feature](#)

7.1. Default Order

Tags: @default, @orderBook

- Given an empty order book
- And the following orders have been placed:

<i>way</i>	<i>qty</i>	<i>price</i>
Buy	150	8.4
Buy	15	9.9
Sell		12.4
Sell		

- Then the order book should be composed of the following orders:

<i>order type</i>	<i>way</i>	<i>qty</i>	<i>price</i>
Limit Order	Buy	150	8.4
Limit Order	Buy	15	9.9
Limit Order	Sell	100	12.4
Limit Order	Sell	100	10.1

8. Order Book Best Prices

Uri: [yamex/feature/order/02-orderbook-queries/01-orderbook-best-prices.feature](#)

In the case of limit orders, orders with the best possible prices:

- highest price limit for buy orders
- lowest price limit for sell orders
- **Bid** - the price in a buy order
- **Ask** - the price in a sell order

8.1. Best bid price - limit orders only

Tags: @default, @orderBook, @bestPrices

- ✓ Given an empty order book
- ✓ And the following orders have been placed:

<i>instrument</i>	<i>order type</i>	<i>way</i>	<i>qty</i>	<i>price</i>
FFLY	Limit Order	Buy	150	10.4
FFLY	Limit Order	Buy	15	11.9
FFLY	Limit Order	Sell	15	12.9

- ✓ Then the order book's best bid price should be 11.9€

8.2. Best bid price - limit orders and market orders

Tags: @orderBook, @bestPrices

- ✓ Given an empty order book
- ✓ And the following orders have been placed:

<i>instrument</i>	<i>order type</i>	<i>way</i>	<i>qty</i>	<i>price</i>
FFLY	Market Order	Buy	150	
FFLY	Limit Order	Buy	150	10.4
FFLY	Limit Order	Buy	15	11.9
FFLY	Limit Order	Sell	15	12.9

- ✓ Then the order book's best bid price should be 11.9€

8.3. Best ask price - limit orders only

Tags: @orderBook, @bestPrices, @limitOrder

- ✓ Given an empty order book
- ✓ And the following orders have been placed:

<i>instrument</i>	<i>order type</i>	<i>way</i>	<i>qty</i>	<i>price</i>
FFLY	Limit Order	Sell	150	11.9
FFLY	Limit Order	Sell	15	10.4
FFLY	Limit Order	Buy	15	10.1

- ✓ Then the order book's best ask price should be 10.4€

8.4. Best ask price - limit orders and market orders

Tags: @orderBook, @bestPrices, @limitOrder

- ✓ Given an empty order book

- ✓ And the following orders have been placed:

<i>instrument</i>	<i>order type</i>	<i>way</i>	<i>qty</i>	<i>price</i>
FFLY	Market Order	Sell	150	
FFLY	Limit Order	Sell	150	11.9
FFLY	Limit Order	Sell	15	10.4
FFLY	Limit Order	Buy	15	12.9

- ✓ Then the order book's best ask price should be 10.4€

8.5. Best ask price - limit orders and stop orders

Tags: @orderBook, @bestPrices, @limitOrder, @stopOrder



Stop order should not be taken into account for best price.

- ✓ Given an empty order book

- ✓ And the following orders have been placed:

<i>instrument</i>	<i>order type</i>	<i>way</i>	<i>qty</i>	<i>price</i>
FFLY	Stop Order	Sell	150	10.1
FFLY	Limit Order	Sell	150	11.9
FFLY	Limit Order	Sell	15	10.4
FFLY	Limit Order	Buy	15	10.0

- ✓ Then the order book's best ask price should be 10.4€

9. Order Book Accumulative Quantities

Uri: [yamex/feature/order/02-orderbook-queries/02-orderbook-accumulative-qty.feature](#)

Spreading information : Share market collects every type of information in respect of the listed companies.

Generally, this information is published or in case of need anybody can get it from the stock exchange free of any cost. In this way, the stock exchange guides the investors by providing various types of information.

Consequently, the number of shareholders in companies is increasing continuously. Thus, the stock exchanges are playing a vital role in ensuring wider share ownership.

9.1. Cumulative Totals of Bids and Offers

Tags: @orderBook, @cumulativeView, @limitOrder

- ✓ Given an empty order book
- ✓ And the following orders have been placed:

<i>way</i>	<i>qty</i>	<i>price</i>
Buy	150	47
Buy	70	46
Buy	100	45
Buy	1	44
Buy	30	43
Sell	226	48
Sell	1	49
Sell	100	50
Sell	30	51

- ✓ Then the order book's view should look like:

<i>Sum of Bids</i>	<i>Bid Qty</i>	<i>Bid Price</i>	<i>Ask Price</i>	<i>Ask Qty</i>	<i>Sum of Asks</i>
150	150	47	48	226	226
220	70	46	49	1	227
320	100	45	50	100	327
321	1	44	51	30	357
351	30	43			

- ✓ And the order book's available quantities for sell under 50.5€ should be 327
- ✓ And the order book's available quantities for buy over 44.5€ should be 320

10. Matching Principles for limit orders

Uri: [yamex/feature/order/03-matching-principles/01-matching-principles-limit-orders.feature](#)

When a new order (or quote) is entered, the Yamex system first checks the limits of all orders contained in the central order book. If the incoming order is immediately executable, meaning it is capable of being matched against an existing order or orders, one or more transactions are generated.

To be immediately executable, the order must be:

- A market order, where opposite already exist in the central order book;
- an order to buy at a price at or above the lowest offer in the central order book;
- an order to sell at a price at or below the highest bid in the book.

The orders already present in the order book are always executed at their specified limit price. Price improvements for orders in the order book are only possible during an auction process - opening or closing auction. Orders going into the order book are always matched at the appropriate prices available in the order book, up to the specified limit price.

10.1. Matching a Buy order partially - exact same price

Tags: [@orderBook](#), [@matchingPrinciple](#), [@limitOrder](#)

Order book contains already two sell orders.

The buy order triggered is not fully fulfilled, thus remains in the order book but with only the missing quantity.

But an execution is triggered for the partial fulfillment

Alternate scenario?

Given an order book containing two sell orders: 15@10.4 by B1 and 150@11.9 by B2
When a buy order is placed 20@10.4 by BBuyer
Then the buy order should remain in the order book with the remaining quantity of 5@10.4
But an execution should have been triggered: [B1->BBuyer 15@10.4]

✓ Given an empty order book

✓ And the following orders have been placed:

Broker	order type	way	qty	price
B1	Limit Order	Sell	15	10.4
B2	Limit Order	Sell	150	11.9

✓ When a limit order is placed to buy 20 FFLY at 10.4€ by "Broker-A"

✓ Then the order book should be composed of the following orders:

Broker	order type	way	qty	price
Broker-A	Limit Order	Buy	5	10.4
B2	Limit Order	Sell	150	11.9

✓ And the following execution should have been triggered:

seller broker	buyer broker	qty	price
B1	Broker-A	15	10.4

10.2. Matching a Buy order partially - higher buy price

Tags: [@orderBook](#), [@matchingPrinciple](#), [@limitOrder](#)

In the case of limit orders, orders with the best possible prices:

- highest price limit for buy orders
- lowest price limit for sell orders

always take precedence in the matching process over other orders with worse prices. Again, if the limit orders have the same price limit, the criterion used for establishing matching priority is the order timestamp.

If not executed upon entry, an order is held in the central order book.

✓ Given an empty order book

✓ And the following orders have been placed:

<i>Broker</i>	<i>order type</i>	<i>way</i>	<i>qty</i>	<i>price</i>
B1	Limit Order	Sell	15	10.4

✓ When a limit order is placed to buy 20 FFLY at 11.4€ by "Broker-A"

✓ Then the order book should be composed of the following orders:

<i>Broker</i>	<i>order type</i>	<i>way</i>	<i>qty</i>	<i>price</i>
Broker-A	Limit Order	Buy	5	11.4

✓ And the following execution should have been triggered:

<i>seller broker</i>	<i>buyer broker</i>	<i>qty</i>	<i>price</i>
B1	Broker-A	15	11.4

10.3. Matching a Sell order partially - exact same price

Tags: @orderBook, @matchingPrinciple, @limitOrder

✓ Given an empty order book

✓ And the following orders have been placed:

<i>Broker</i>	<i>order type</i>	<i>way</i>	<i>qty</i>	<i>price</i>
B1	Limit Order	Buy	150	10.4
B2	Limit Order	Buy	15	11.9

✓ When a limit order is placed to sell 20 FFLY at 11.9€ by "Broker-A"

✓ Then the order book should be composed of the following orders:

<i>Broker</i>	<i>order type</i>	<i>way</i>	<i>qty</i>	<i>price</i>
Broker-A	Limit Order	Sell	5	11.9
B1	Limit Order	Buy	150	10.4

✓ And the following execution should have been triggered:

<i>seller broker</i>	<i>buyer broker</i>	<i>qty</i>	<i>price</i>
Broker-A	B2	15	11.9

10.4. Matching a Sell order against multiple Buy orders - higher buy price fulfilled first

Tags: @orderBook, @matchingPrinciple, @limitOrder

Orders may not necessarily be executed at a single price, but may generate several partial transactions at different prices. When a large order executes against the total available quantity at a given price level, the next best price level becomes best. This process continues as long as the incoming order remains executable.

Also, it is possible for a single order to generate multiple executions at different points in time. For example, an order may generate a partial execution upon entry, while the remaining open order remains in the order book. The open portion may get executed a minute later, an hour later, or even a day later, if its validity extends beyond the current trading day.

✓ Given an empty order book

✓ And the following orders have been placed:

<i>Broker</i>	<i>order type</i>	<i>way</i>	<i>qty</i>	<i>price</i>
B1	Limit Order	Buy	150	10.4
B2	Limit Order	Buy	15	11.9
B3	Limit Order	Buy	15	12.3

✓ When a limit order is placed to sell 20 FFLY at 11.9€ by "Broker-A"

✓ Then the order book should be composed of the following orders:

<i>Broker</i>	<i>order type</i>	<i>way</i>	<i>qty</i>	<i>price</i>
B1	Limit Order	Buy	150	10.4
B2	Limit Order	Buy	10	11.9

✓ And the following executions should have been triggered:

<i>seller broker</i>	<i>buyer broker</i>	<i>qty</i>	<i>price</i>
Broker-A	B3	15	12.3
Broker-A	B2	5	11.9

10.5. Matching a Sell order against multiple Buy orders - buys fulfilled in fifo

Tags: @orderBook, @matchingPrinciple, @limitOrder

When an order (or quote) is entered into the order book, it is assigned a timestamp. This timestamp is used to prioritize orders in the book with the same price - the order entered earliest at a given price limit gets executed first (FIFO).

✓ Given an empty order book

✓ And the following orders have been placed in order:

<i>Broker</i>	<i>order type</i>	<i>way</i>	<i>qty</i>	<i>price</i>
B1	Limit Order	Buy	150	10.4
B2	Limit Order	Buy	15	12.3
B3	Limit Order	Buy	15	11.9
B4	Limit Order	Buy	15	12.3

✓ When a limit order is placed to sell 20 FFLY at 11.9€ by "Broker-A"

✓ Then the order book should be composed of the following orders:

<i>Broker</i>	<i>order type</i>	<i>way</i>	<i>qty</i>	<i>price</i>
B1	Limit Order	Buy	150	10.4
B3	Limit Order	Buy	15	11.9
B4	Limit Order	Buy	10	12.3

✓ And the following executions should have been triggered:

<i>seller broker</i>	<i>buyer broker</i>	<i>qty</i>	<i>price</i>
Broker-A	B2	15	12.3
Broker-A	B4	5	12.3

11. Matching Principles for market orders

Uri: [yamex/feature/order/03-matching-principles/02-matching-principles-market-orders.feature](#)

11.1. Matching a new Market Order to Buy against existing multiple sell limit orders

Tags: @orderBook, @matchingPrinciple, @marketOrder

- ✓ Given an empty order book
- ✓ And the following orders have been placed:

Broker	order type	way	qty	price
B1	Limit Order	Sell	15	10.4
B2	Limit Order	Sell	150	11.9

- ✓ When a market order is placed to buy 20 FFLY by "Broker-A"

- ✓ Then the order book should be composed of the following orders:

Broker	order type	way	qty	price
B2	Limit Order	Sell	145	11.9

- ✓ And the following execution should have been triggered:

seller broker	buyer broker	qty	price
B1	Broker-A	15	10.4
B2	Broker-A	5	11.9

11.2. Matching a new Limit Order to Buy against existing multiple sell orders market and limit

Tags: @orderBook, @matchingPrinciple, @marketOrder

- ✓ Given an empty order book
- ✓ And the following orders have been placed:

Broker	order type	way	qty	price
B1	Limit Order	Sell	150	11.9
B2	Market Order	Sell	15	

- ✓ When a market order is placed to buy 20 FFLY by "Broker-A"

- ✓ Then the order book should be composed of the following orders:

Broker	order type	way	qty	price
B1	Limit Order	Sell	145	11.9

- ✓ And the following execution should have been triggered:

seller broker	buyer broker	qty	price
B2	Broker-A	15	11.9
B1	Broker-A	5	11.9






12. Control invariants

Uri: [yamex/feature/order/99-invariants/01-invariants.feature](#)

12.1. High volume of orders

- ✓ Given an empty order book
- ✓ When 1500 of (limit/market) orders are placed to (sell/buy) <any> qty at <any>€
- ✓ Then no sell market order should remain if there are any buy limit orders remaining
- ✓ And no buy market order should remain if there are any sell limit orders remaining
- ✓ And $(\sum \text{quantities in order book}) + (\sum \text{quantities in executions})$ should be identical to $(\sum \text{quantities in placed orders})$

Sample Steps

-  Given a passed step
-  And a failed step
-  When a pending step
-  But an undefined step
-  Then a skipped step