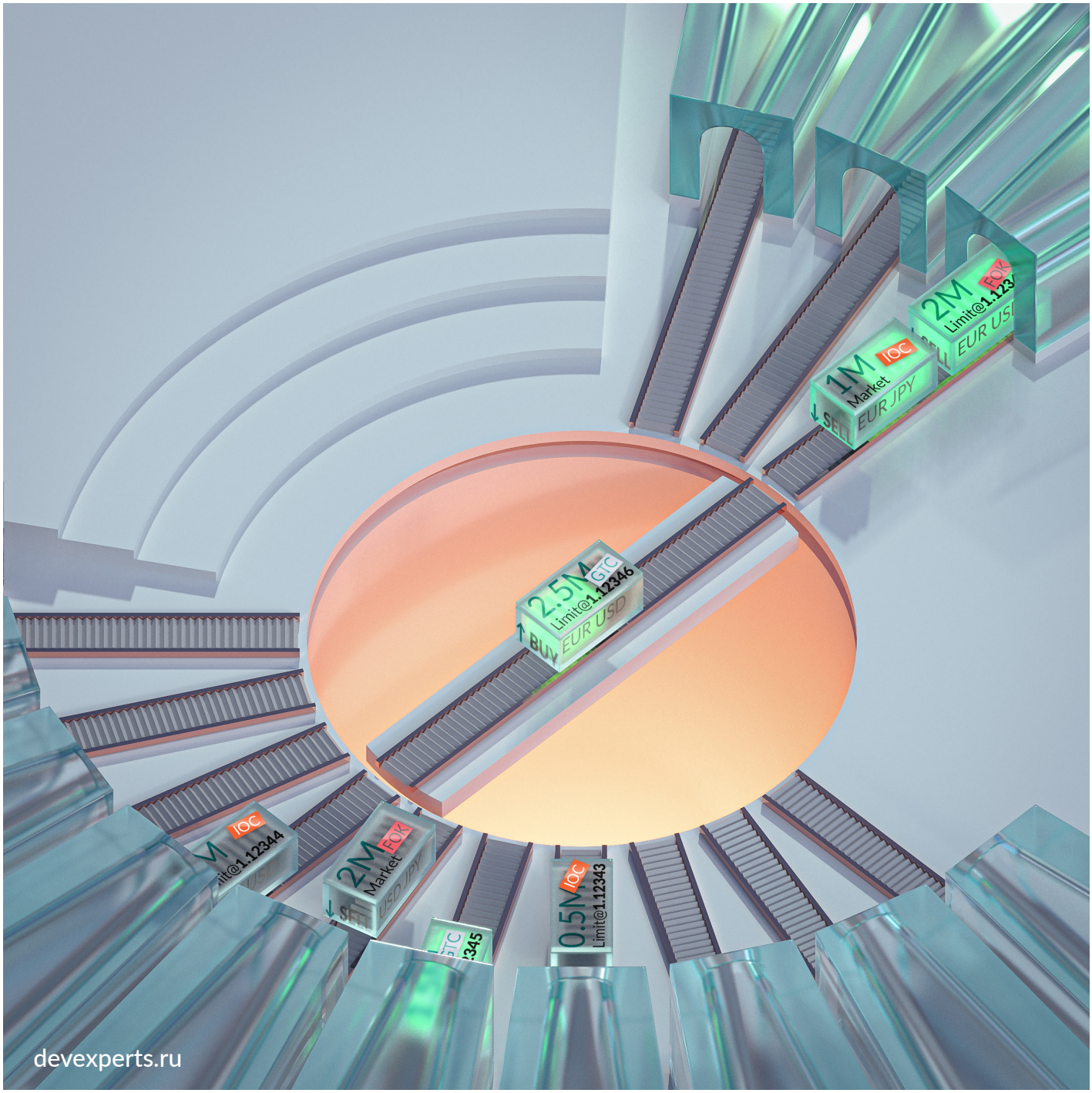
Devexperts

ORDER-SWEEPING ALGORITHMS IN FOREX TRADING



devexperts.ru

Disclaimer & Copyright Notice

This book provides helpful information about liquidity aggregation on the forex market. Statements, figures, calculations, images, and representations are used for illustrative purposes only and do not constitute an offer, inducement, or warranty.

Information in this book is presented based on research and circumstances as of April 2022. Evgeny Sorokin, the author of this work, and Devexperts accept no liability for any damages, costs and/or expenses resulting from the application of any information contained in this book by any party.

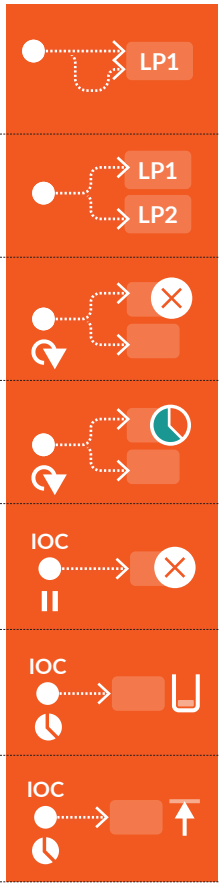
Acknowledgements

This book would not have been made possible without editing and valuable remarks from Evros Stylianou, George Papamatthaiakis, Elena Gorbacheva, Jon Light, Vitaly Kudinov, Andrew Artamonov, Samantha Schorzman, and OWL Agency.

About the author

Evgeny Sorokin is SVP of Software Engineering at Devexperts, a financial software provider company, where he is heading the R&D Department and managing around 600 IT engineers. He has worked at a number of high tech companies including Quotix (an FxPro Company) and Intel Corporation. His strength lies in his considerable knowledge of technology trends and innovations, and how they can be implemented into the development process. Along with developing software, Evgeny is focused on looking for talent and building the right teams suited for a particular project.

CONTENTS



CONTENTS

DEVEXPERTS

31 Conclusion

INTRODUCTION

Forex brokers work with liquidity providers (LPs), who can be banks, large brokers, ECNs, etc. In turn, LPs compete with each other for brokers' order flow and are interested to receive as much trading volume as possible from every client. To that end, LPs set their pricing feed (refresh rate, spread mark-ups) individually for every forex/CFD broker. So what you need to know about the pricing methodology is that the better a broker negotiates the price, the better prices they'll get.

The important note here is that LPs are in the risk management business, rather than in the investment business. So every time they need to figure out the best deal, they undergo empirical estimation, balancing risk and reward, to offer the broker their price. At some point, LPs might even figure out the broker's order flow pattern, like if they always send orders over 5M in smaller chunks. In this case, this LP will adapt the tier structure accordingly. They can even change the tier sizes or spreads for one specific broker to maximize profits.

LPs often try to get brokers to channel incoming and outgoing flows through them by offering a tighter spread, so they can have a bigger picture of the market. That said, that doesn't mean brokers are getting the best prices.

How can brokers protect themselves from such a dependency on their LP?

First of all, they need to add redundancy by sending orders to more than one LP and let them compete for the broker's order flow. That way, brokers get to choose where to send the orders. They also secure additional liquidity streams, in case an LP sends off-market quotes or no quotes at all.

Here's where a liquidity aggregator comes in. When we say 'aggregation', we imply that there are several LPs for a given instrument. A liquidity aggregator is a piece of software that processes price streams (liquidity) from several trading venues (LPs), those who signed with this broker, and converges these streams into one to ensure the best execution. It automatically spots a malfunction and reacts quickly, protecting brokers from outdated liquidity.

We'll examine real scenarios later in this e-book and how the aggregator would improve execution price or fill ratio. If you're a broker, then knowing these scenarios will help you see the weak points and make sure your aggregator works properly. If you're a trader, a software engineer, financial analyst, compliance officer, or c-level executive, we hope all scenarios in this e-book will help you understand complex order management systems. For now, let's get to know how the LP sets the prices for orders.

PRICE AGGREGATION

Let's assume the broker is currently working with three LPs and this is how they see the incoming price tiers for the same instrument (e.g. EURUSD):

Example 1 In this example, the tiers are independent, and the broker or their dealers see the bigger picture of the market, or in other words the whole depth of market (DoM)

DoM — depth of market

LP1	
Volume	Price
1M	1.21124
2M	1.21125
3M	1.21126

LP2	
Volume	Price
1M	1.21125
3M	1.21127
5M	1.21128

LP3	
Volume	Price
0.5M	1.21124
1M	1.21125
2M	1.21127

Total DoM:	
1.5M	1.21124
4M	1.21125
3M	1.21126
5M	1.21127
5M	1.21128

In most cases, the table above is reorganized into the table below, and this is just another representation of the DoM. The algorithm has to sort the tiers with the same price level (or a price tie) somehow. In our implementation, the algorithm will favour the tier with the bigger size.

Example 2 Disclosed DoM—tiers displayed separately

BROKER'S BOOK		
Volume	Price	
1M	1.21124	LP1
0.5M	1.21124	LP3
2M	1.21125	LP1
1M	1.21125	LP2
1M	1.21125	LP3
3M	1.21126	LP1
3M	1.21127	LP2
2M	1.21127	LP3
5M	1.21126	LP2

You might've already seen this example in the previous e-book: **"The Ins and Outs of Forex Liquidity Aggregation".**

Undisclosed vs Disclosed Market Depth

Aside from organizing liquidity and serving pricing advantages, the liquidity aggregator helps the broker in cases when they want to show the depth of market in an anonymized way.

They might prefer to maintain LP anonymity and not disclose their DoM for the following reasons:

1. **Client retention:** Since brokers act as a counterparty for the client's trade, disclosing the LPs might prompt clients to go and work with the specific LP directly, thus leaving the broker with less business. This can be the case when the LP is the entity of the same type (e.g. a retail broker sources liquidity from another retail broker, or an institutional broker sources liquidity from another institutional broker).

Vitaly Kudinov,
SVP of Business Development

> ***Or the broker has both retail and institutional divisions and even though spreads are different for B2B and B2C price feeds, it will be still an advertisement of the competitor.***

2. **Non-bank market makers:** Another case for undisclosed liquidity is when a non-bank market maker may want to hide that they participate in a specific liquidity pool.

3. **Reputational loss:** Sometimes, LPs don't want a broker to disclose to their traders that they're behind the trade rejection to avoid a reputational loss.

Alternatively, the broker might prefer to disclose their DoM, for these reasons:

PoP —
prime of prime

1. **Quality advertisement:** A prime of prime (PoP) may want to advertise the quality of its liquidity to clients. Since most of the PoP's clients can't afford proper aggregation and a prime brokerage setup, a PoP suffers little client loss risk that they would start working with the LPs directly.

2. **Market monitoring:** When dealers may be monitoring the market. They'd want to visualize who is the top performing LP, what is the top of the book size, etc.

Now that we've gone through how price aggregation looks and the reasons for disclosed and undisclosed view, let's examine how we get to the aggregates.

EXECUTION

The aggregator employs various order management and routing algorithms to facilitate and route trading with LPs. This enables better execution prices, especially for large orders.

How does this work? The aggregator splits the order into several chunks and sends them to several LPs. That way the LPs compete for the order flow, which makes trading cheaper as the LPs shrink their spreads to land the deal. That translates into a better volume-weighted average price (VWAP). More liquidity enables to execute larger orders, which otherwise would not have been filled.

VWAP — volume-weighted average price

LPs have different preferences for how one can receive their liquidity. Some market makers provide VWAP pricing, while others provide liquidity in independent tiers, as visualized in examples 1 and 2 above. It depends on their specifications and the aggregator brings them all into a single view. In this book, we show you different ways of representation to illustrate how the aggregator accumulates different liquidity flows in a consolidated view to be used together.

Some LPs could be PoPs (Prime of Primes) who also aggregate liquidity and therefore, can be treated similarly to an exchange's limit order book.

In addition, tiers might be exclusive and inclusive. An exclusive one, which resembles the limit order book construction at an exchange, offers streams of independent tiers. An inclusive one has already calculated VWAP and shows a tiered price structure.

BUY Market 1,1M EURJPY

Inclusive Volume	Price	Exclusive Volume	Price
1M	130.7**21**	1M	130.7**21**
2M	130.7**22**	1M	130.7**23**
3M	130.7**23**	1M	130.7**25**

For example, a broker sends a Buy Market Order 1.1M. To start execution, the aggregator will analyze the LP price tiers. When receiving inclusive tiers, the aggregator will execute the order at a price of 130.722, and the transaction cost is 1.1 × 130.722 = 143.794.

For exclusive tiers, the aggregator will execute a Buy 1M order at a price of 130.721 and a Buy 100K order at a price of 130.723. That results in a VWAP of 130.7212, which is 0.0008 JPY better (130.722-130.7212 = 0.0008 JPY) when comparing to the execution against inclusive tiers. The client paid 1.1M*130.7212 = 143,793,320 JPY instead of 143,794,200 JPY. In this example, executing the order with exclusive price tiers saved the client 880 JPY.

In this e-book, we will take the simplest approach and treat liquidity as if it was segmented into **exclusive tiers**, the same as in an exchange's limit order book. We also assume the market snapshot is always up-to-date.

Order Sweeping Algorithm Workflow

As introduced above, brokers employ liquidity aggregators to work fairly with several liquidity providers. From the moment the broker issues the order, the system should process it following specific rules set in the liquidity aggregator. It then executes it on heterogeneous liquidity streams. The aggregator usually works on a price priority basis and tries to protect the broker if an LP wants to take advantage of this broker. In case the LP starts to send unusually high prices compared with other LPs, the aggregator simply won't forward the order flow to this LP.

Taking into account the available liquidity and according to the tiers, the execution algorithm splits each order into parts iterating from the top of the book down, until the entire order volume is accounted for. Once the aggregator finalizes the order split, it sends these chunk parts to the LPs. The simplest implementation of order sweeping ignores the information about the relationships among different tiers in the market depth (e.g. the tiers may belong to the same LP) and may hit the same LP multiple times. However, we need to keep in mind that LPs may have some preferences regarding how to handle multiple orders against the same tiers. We'll investigate how to ensure execution in this case further down.

Sweeping Triggers

An incoming order triggers the algorithm to process it. How it happens, though, depends on the order specifications. The most common orders are Market, Limit, and Stop orders. All of them are subject to sweeping immediately upon receipt from the client.

In case of an incoming Market order: once the order is swept and all outgoing orders for the incoming order are sent to the LPs, the aggregator shall wait for the results from the LPs. After receiving any execution report from the LPs, the aggregator shall analyze if all the order's volume is filled and another iteration is required. If the entire market order is filled, then the algorithm stops. If the entire order isn't filled, and its specification allows further processing, the order shall be held resting to wait for the next tick event for the given instrument. On a new tick for this instrument, the order-sweeping process is re-triggered, but only takes the remaining volume into account.

In case of an incoming Limit order: if no price in the market depth is suitable for the Limit order, then the order remains rested. If the available price in the DoM is suitable for the limit order, then the aggregator sends the outgoing orders towards the LPs behind the suitable tiers. Once the limit order is swept and all outgoing orders for the incoming order are sent to the LPs, the algorithm shall wait for responses from the LPs. After receiving a response from the LP, the aggregator checks if the order is filled fully or not. If the entire order volume is filled then the algorithm stops. If the full order volume wasn't filled, the remaining portion of the limit order will be put on hold to wait for the next tick event. The next tick for this instrument re-triggers the sweeping process for the remaining portion of the order.

Finally, in the case of an incoming Stop order: if the market doesn't meet the stop price, the algorithm keeps waiting for the stop order condition to be met. Once this happens, the algorithm processes the order exactly as it does with market orders.

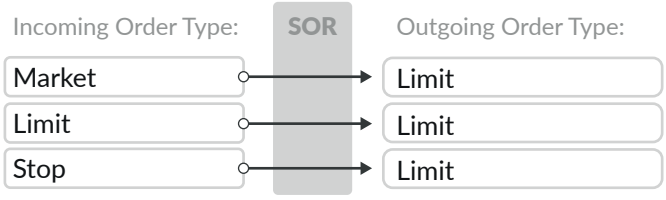
Incoming—Outgoing Order Type Mapping

For any incoming order, the aggregator issues one or more outgoing order(s) and sends them to LPs.

Since one of the key reasons for a broker to use an aggregator is to get a price improvement, the algorithm shall control the execution prices at all times. Generally, this means that even if an incoming order is a market order, the order-sweeping algorithm shall generate limit orders. If the algorithm sends market orders to the LPs, it could have an uncontrollable slippage, undermining the value of the liquidity aggregation.

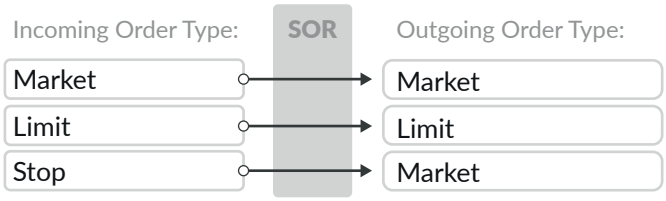
That said, because of this 'limit orders' approach, the broker may not receive a timely fill in a volatile market. If the limit orders hit LPs when the market already moved from the limit price, the algorithm would receive a cancellation and re-iterate. That would typically result in a negative slippage.

SOR —
smart order routing



To minimize rejects on highly volatile instruments, the broker can utilize the built-in slippage protection on the limit orders that get sent to the LPs. That means the algorithm sends the order for a slightly worse price to make sure it gets filled if prices have moved. Although this doesn't often make the fill price worse, as the LP will always fill at the best available price, even if the broker sends an off-market limit order.

Note that this feature of the algorithm (outgoing order types) has to be made configurable. Some clients may prefer a higher probability of a timely fill to best execution. In this case, incoming market/stop orders shall be swept into market orders.



Please note, that many market makers do not support market/limit orders. Instead, they would stream quotes with the unique identifies and accept an incoming order that specifies the respective quote ID and the size. The difference between quote-based and limit order-based liquidity, is that the former offers dependent (inclusive) tiers, whereas in the latter case the tiers are independent (exclusive). The execution against quote-based liquidity features no possibility to obtain a positive slippage.

Incoming—Outgoing TIF Mapping

TIF —
time in force

GTC —
good 'till cancelled

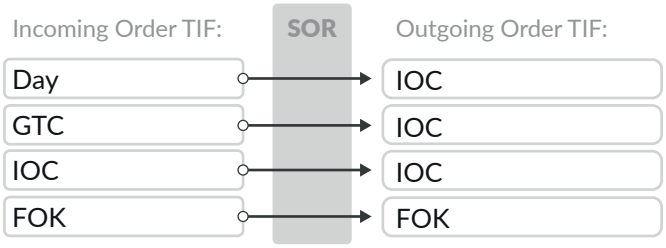
IOC —
immediate or cancel

FOK —
fill or kill

An order has two attributes: type and time in force (TIF). In the previous section, we saw the 3 types an order can take. Now, we'll get to know more about TIF mapping.

A common mapping of TIF is when the aggregator converts all orders into immediate-or-cancel (IOC) and sends them towards LPs as IOC. The only exception is an incoming fill-or-kill (FOK), which has to be sent as a single order with FOK TIF.

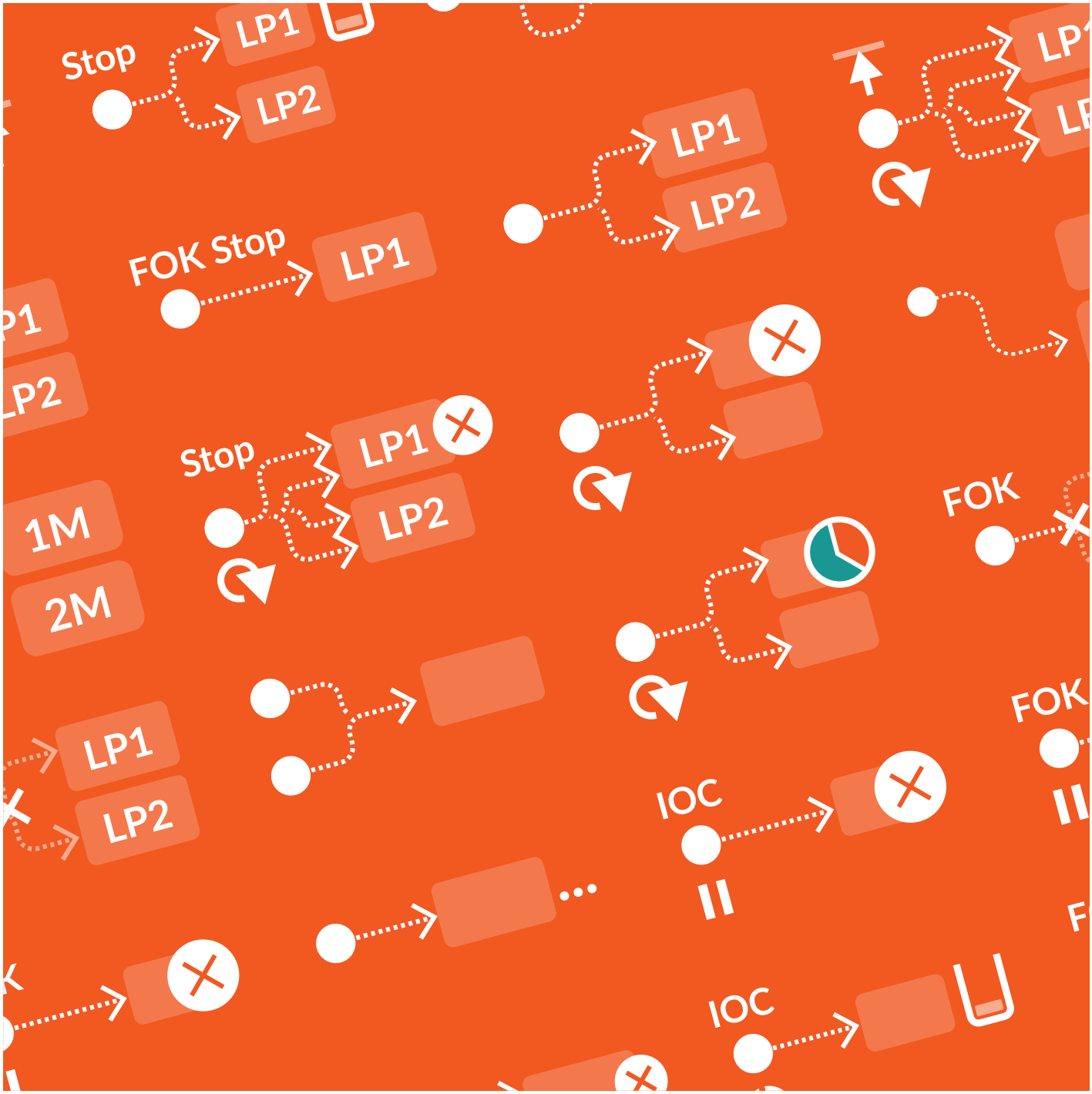
One example is when a good-till-cancelled (GTC) order can be split and sent in chunks as multiple IOC orders. The order-sweeping algorithm should iterate until it achieves a full fill, depending on how many iterations the TIF of an incoming order permits.



These are the basics you need to remember to better memorize the following execution scenarios.

SCENARIOS

Let's now go through different execution scenarios and how the liquidity aggregator can provide the best execution for them.

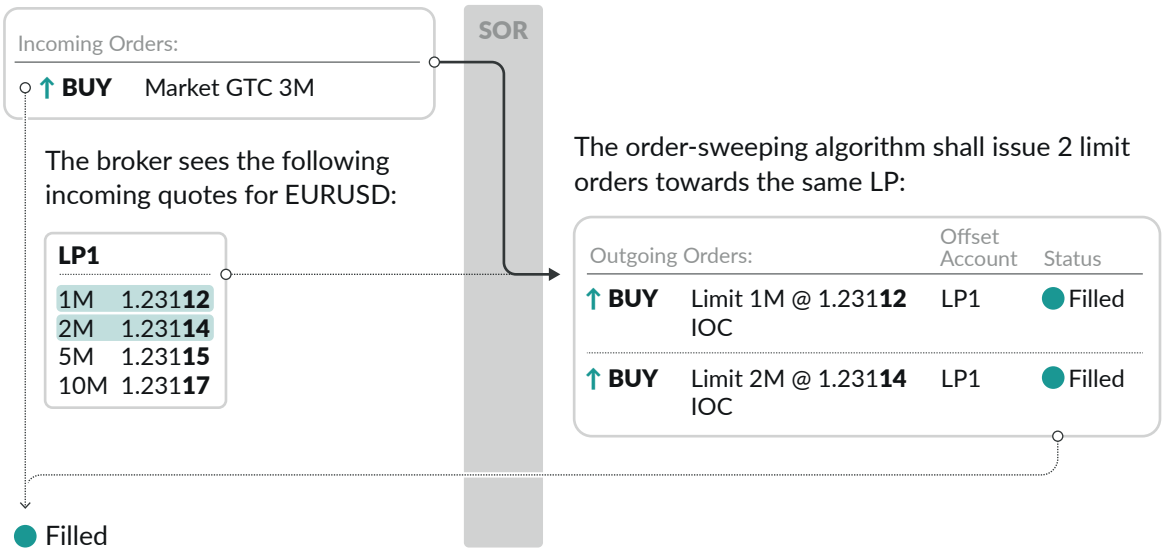


LP1

Scenario 1

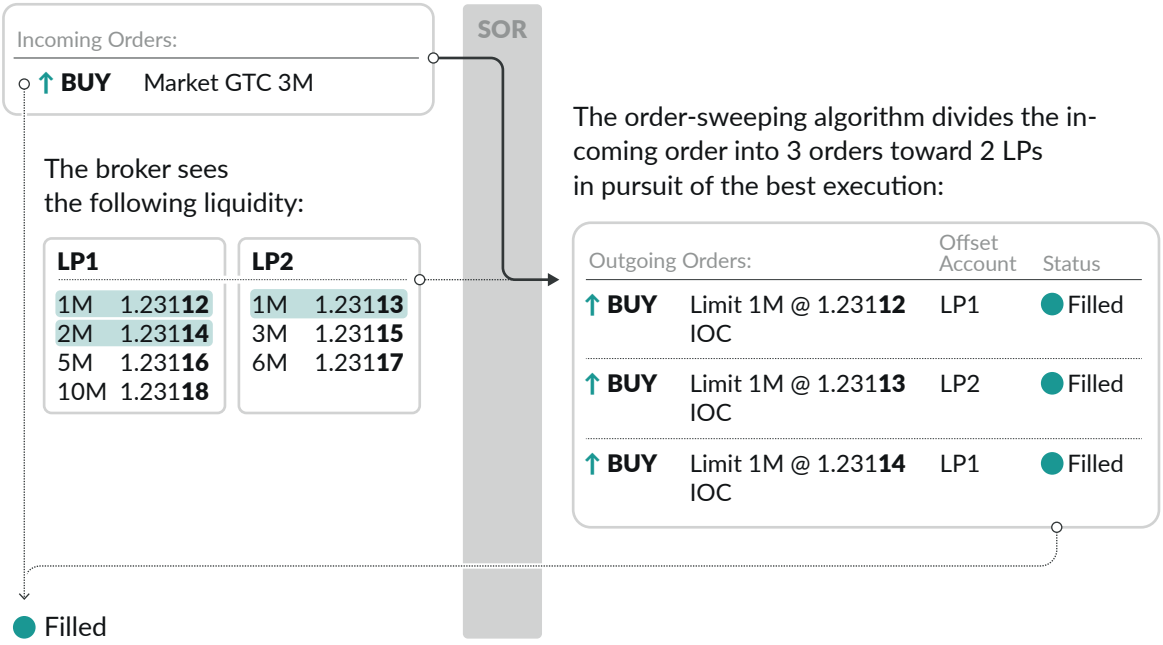
Hit the Same LP Twice If Necessary

In case the aggregator finds that one LP has the best price tiers to execute the trade, it'll apply the Hit the Same LP Twice scenario. For example, the broker has the incoming quotes for EURUSD as in the table below. A trader sends a Buy Market GTC 3M order for execution. Since the tier offer the optimal price to fill the trade, the order-sweeping algorithm shall issue 2 limit orders towards the LP.



Sweep **Buy Market GTC** into Multiple LPs

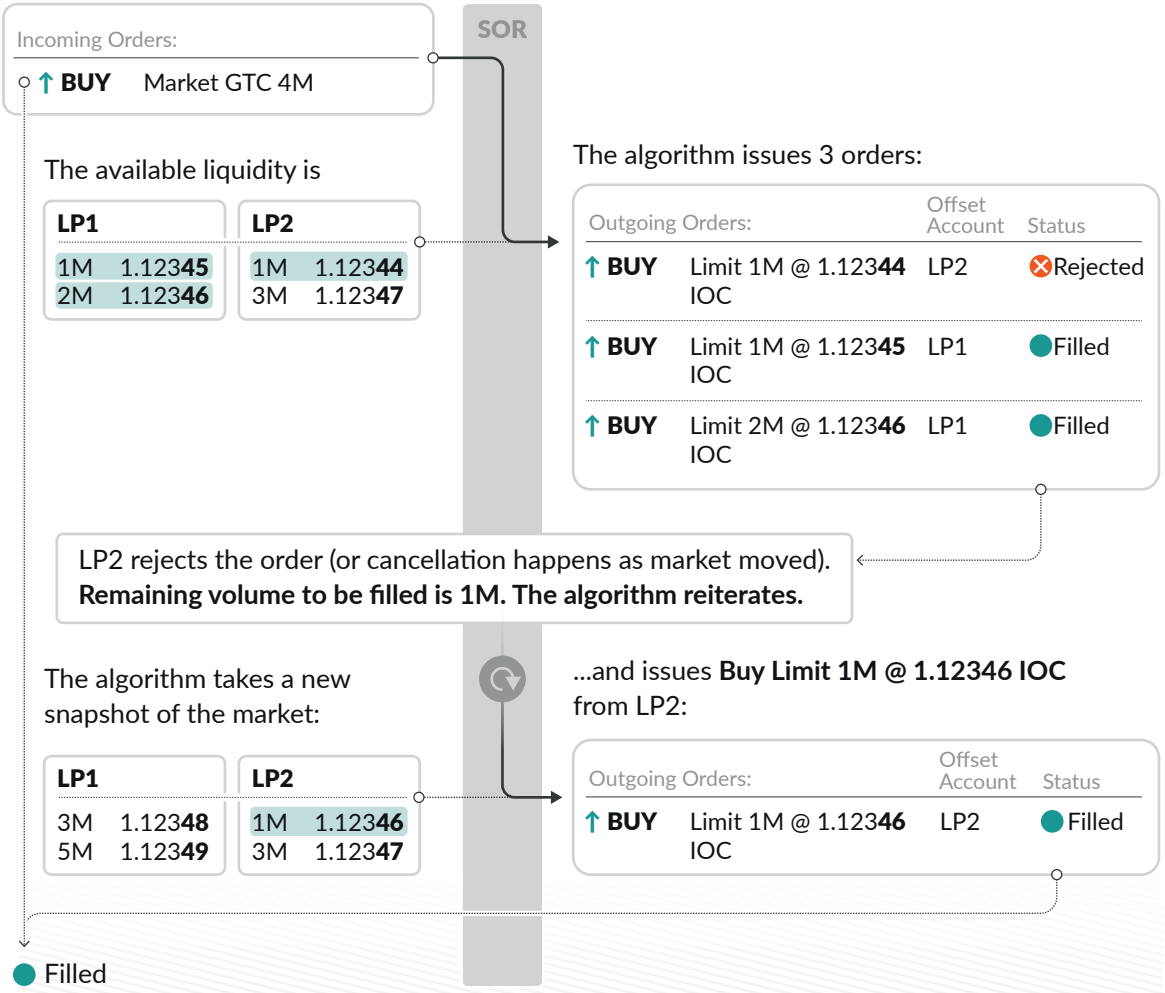
Depending on the order size, competitive prices may be available at different LPs. In this scenario, the order-sweeping algorithm divides the order between multiple LPs to achieve the best execution. For example, a trader sends a Buy Market GTC 3M order, and the order-sweeping algorithm divides it into 3 orders towards 2 LPs:



Scenario 3

Execute **Buy Market GTC** in Iterations Due to Rejection

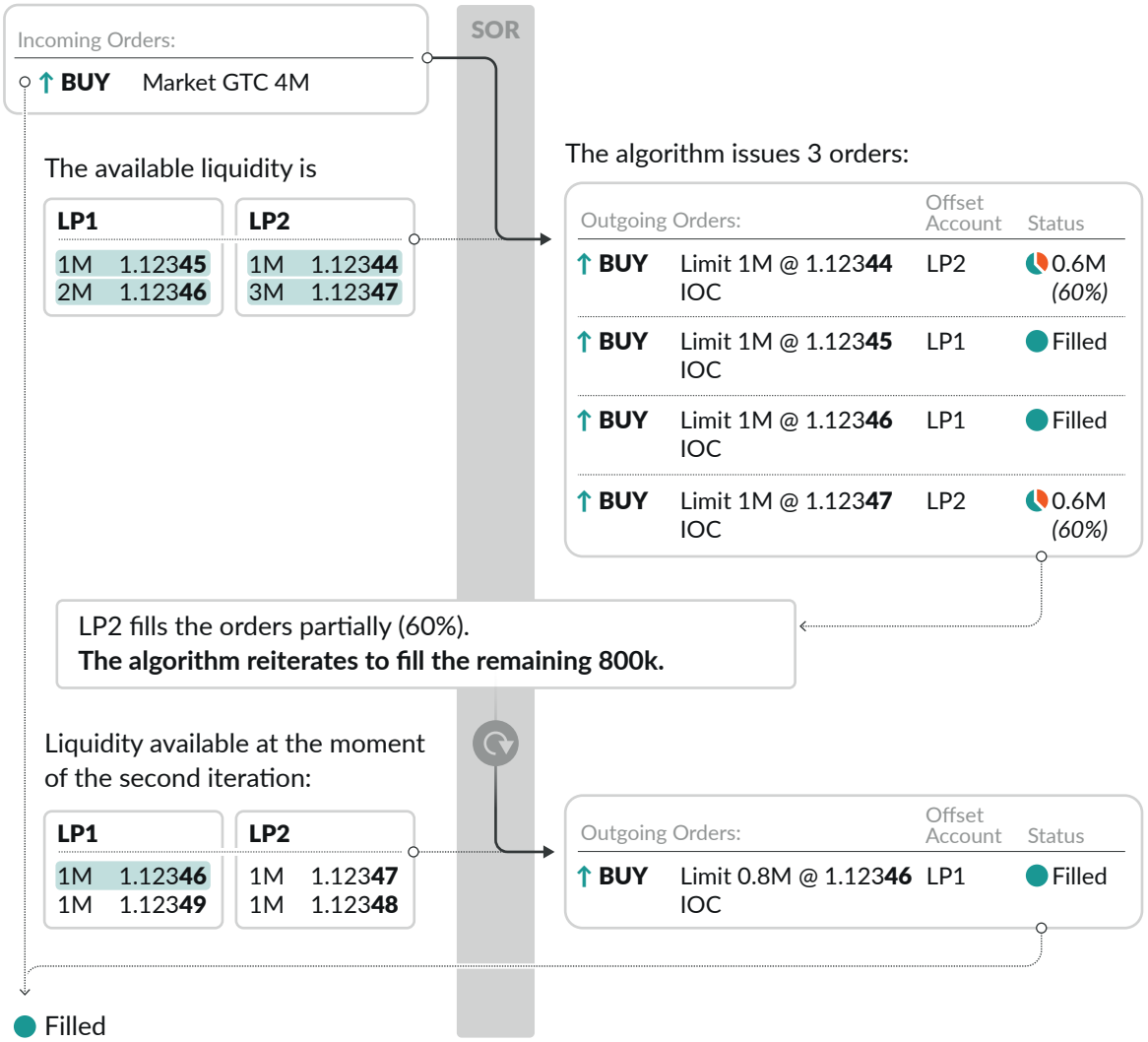
Sometimes the broker receives rejects from an LP. In this case, the aggregator's algorithm reiterates to secure the remaining volume. For example, a broker sends a Buy Market GTC 4M order. The aggregator sees the best execution as 1M from LP2 at the lowest price and the remaining 3M from LP1. LP2, though, rejects the order and the algorithm reiterates to complete the trade.



Scenario 4

Execute **Buy Market GTC** in Iterations Due to **Partial Fill by LP**

A partial fill is typically associated with large trade volumes or results from the LP's risk management controls. In this case, the aggregator reiterates to process the remaining volume. In the below example, LP2 only fills 60% of the order. The algorithm reiterates and completes the execution.

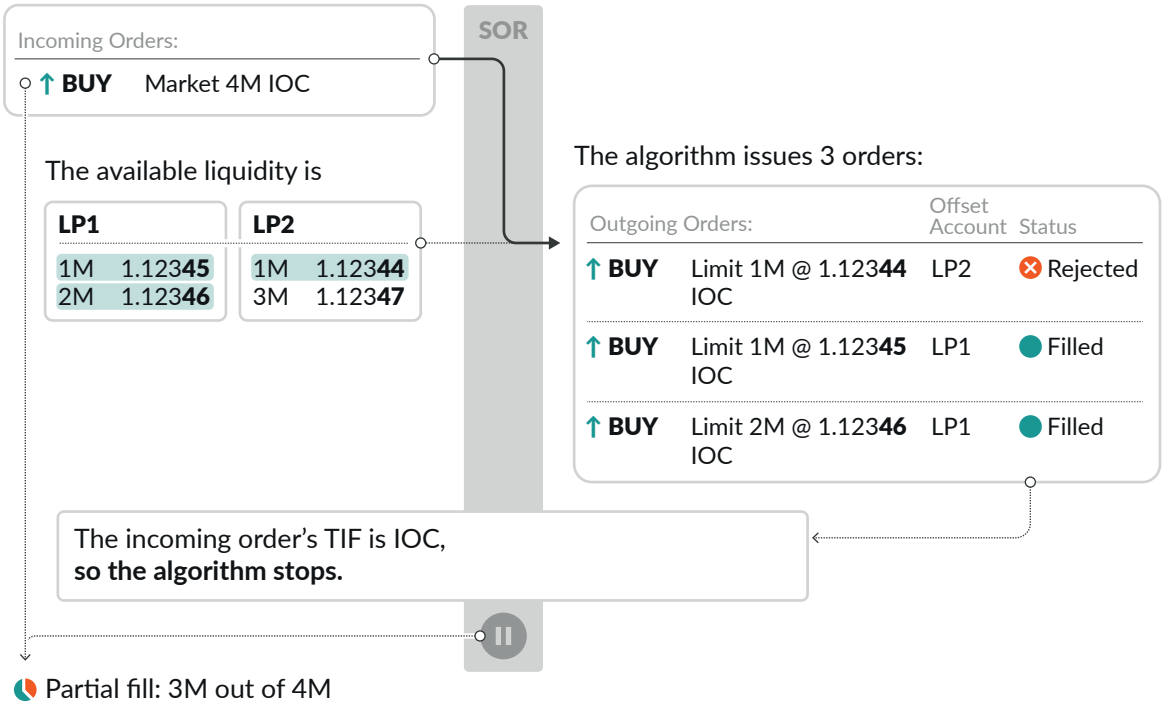


IOC

Scenario 5

Do Not Retry **Buy Market IOC** after **Rejection**

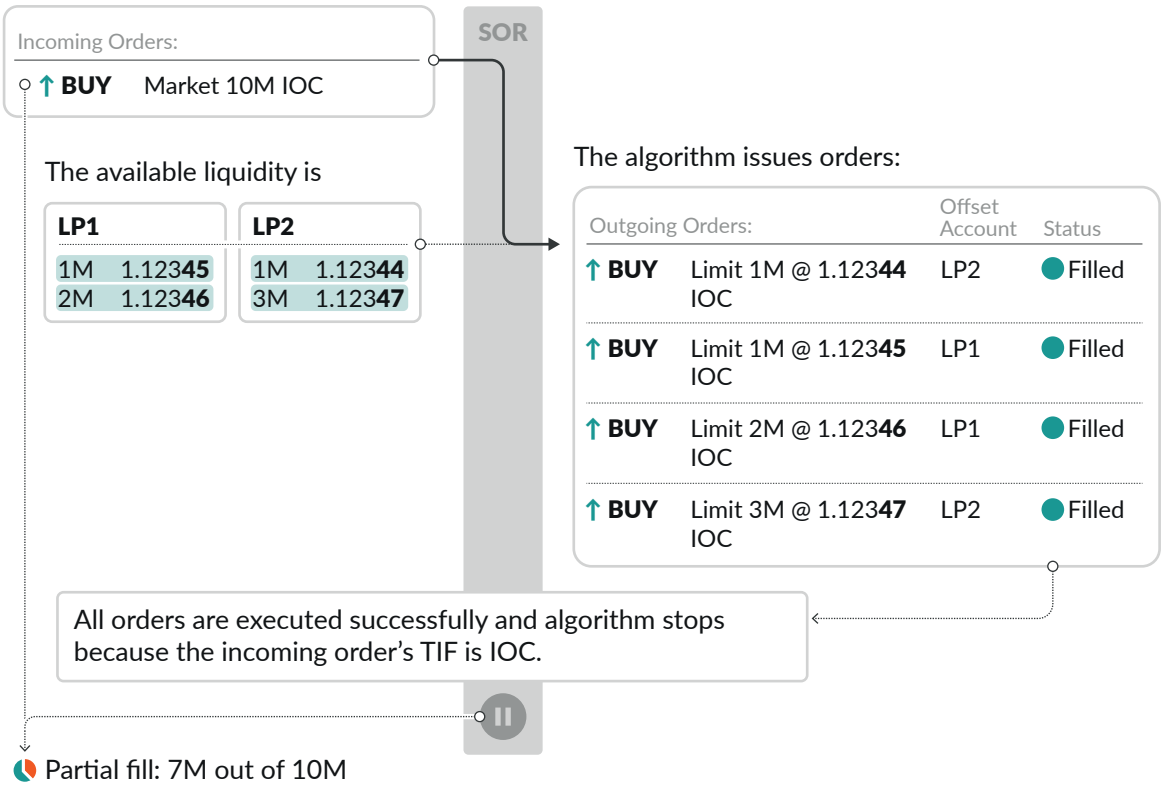
In an immediate-or-cancel (IOC) order, the broker wants to execute the trade immediately. If the market isn't favourable to complete the order in full, the broker doesn't want to make other attempts. In this example, the incoming order is a Buy Market 4M IOC. In this TIF scenario, if the LP rejected the order even though it has enough available liquidity, the algorithm stops.



IOC
Scenario 6

Fill Partially **Buy Market IOC** Due to **Shallow Depth**

Sometimes a broker faces a shallow market where LP offers don't cover the trade in full. In this case, the aggregator will have to settle for a partial fill. For example, a broker issues a Buy Market 10M IOC. In case the LP accepted the order but didn't have enough liquidity, the order would be filled partially.

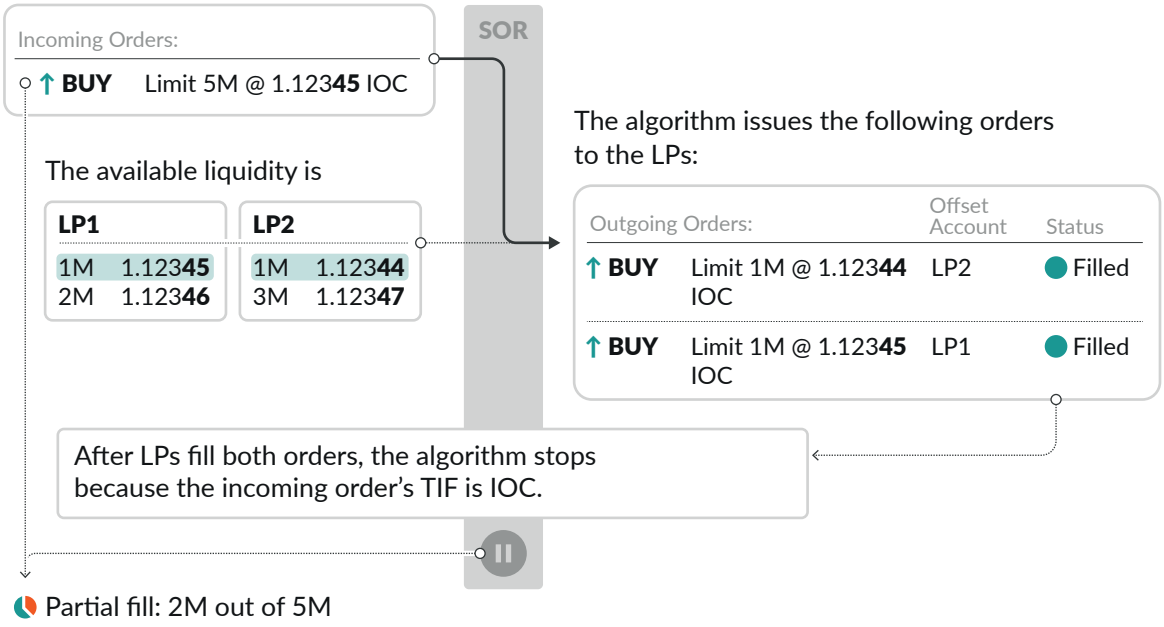


IOC

Scenario 7

Fill Partially **Buy Market IOC** Due to **Limit Price**

When applying a specific strategy, traders might set strict order settings, and the algorithm will try to satisfy their request as much as possible. In this example, the trader sets a limit price for the 5M order. The algorithm will only place the order at a price at or below the limit price, resulting in a partial fill.

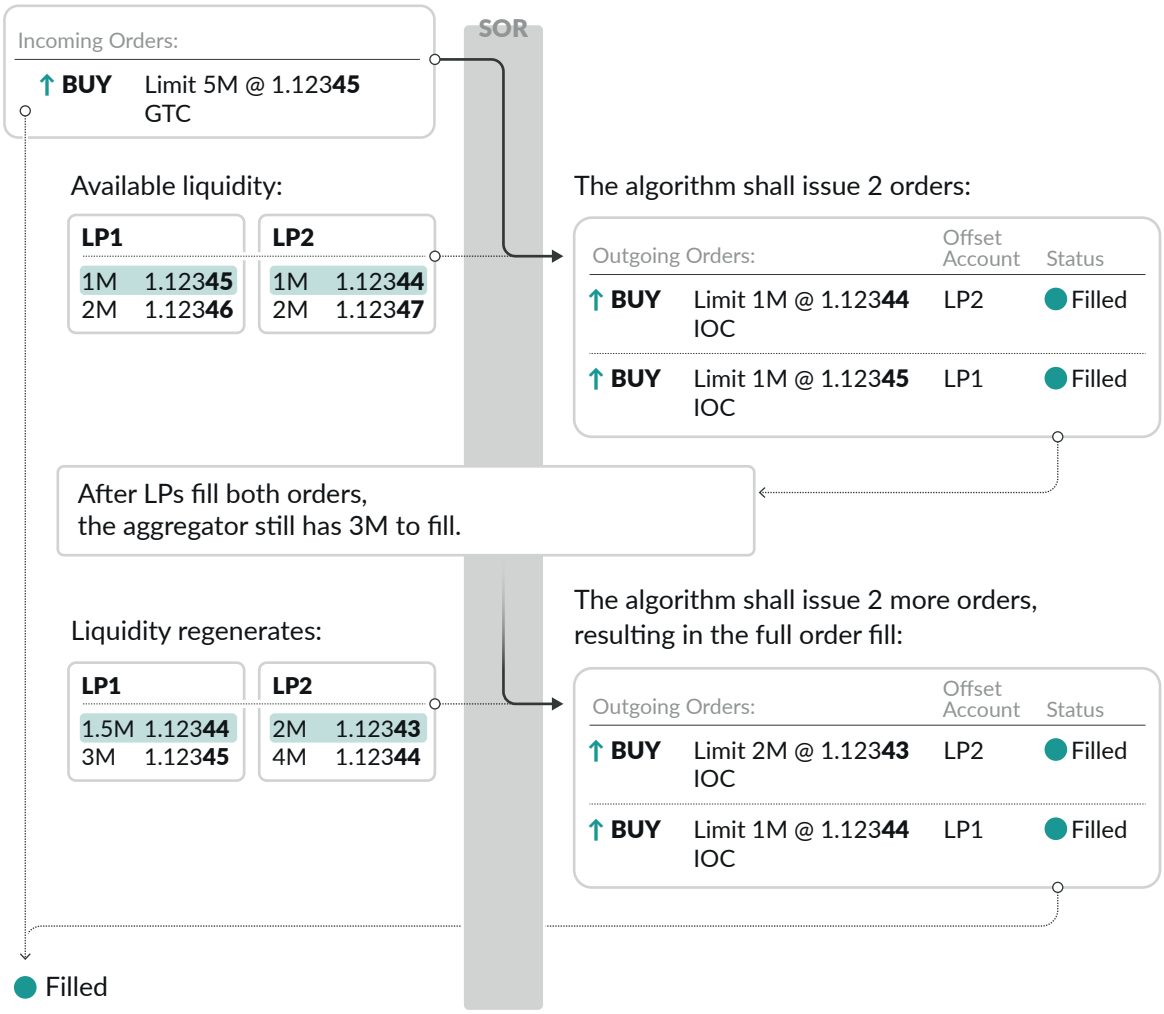


Despite unfavourable prices on the market, the aggregator managed to fill 40% of the order honouring the limit price.

Execute **Buy Limit GTC** Order in Several Steps

(Limit Price Constraint)

In case the time-in-force allows, the algorithm can wait until LPs regenerate their liquidity after the first hit and the tiers meet the criteria to finish the execution. The algorithm will keep reiterating until it fills the order completely. For example, a broker issues a Buy Limit GTC order with a limit price @ 1.12345. The initial market liquidity offers only fill the order partially. That's why the algorithm reiterates until it has a 100% fill.



Select Tier with **Bigger Volume**

In a situation where two LPs offer the same price, and the aggregator needs to decide which one to pick, it needs to resolve a price tie between two or more LPs. The aggregator may prefer to direct the order to the one who sent the quote earlier. But this carries the inherent risks of hitting a stale quote, trade rejection, and inability to provide service to the broker's clients.

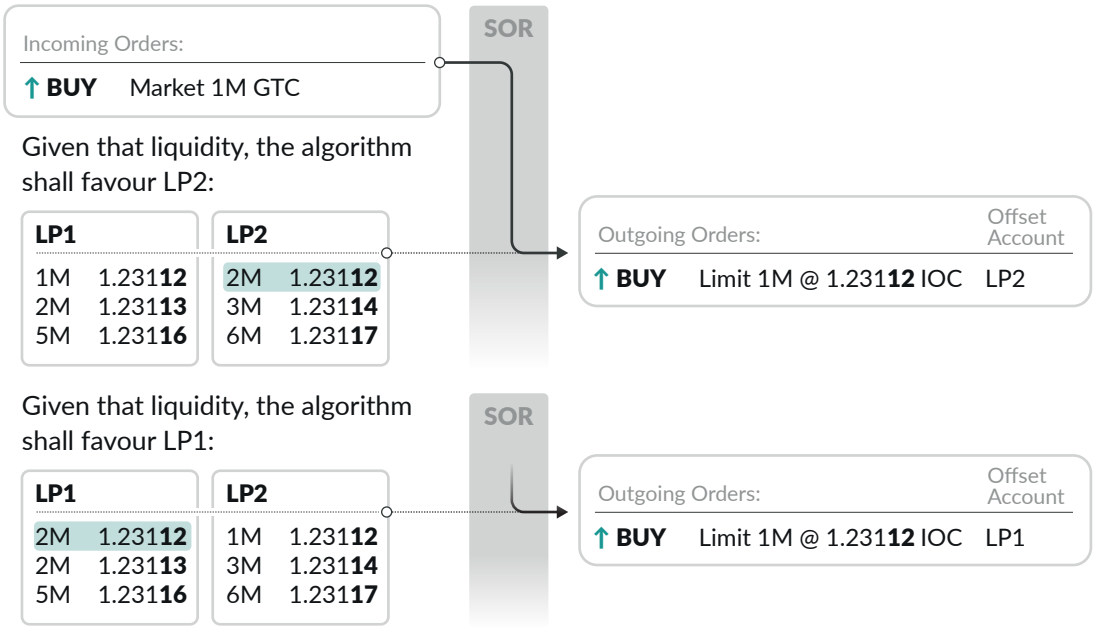
Inversely, the aggregator may prefer to direct the order to the LP who sent the freshest quote. But this setting has its disadvantages too. If the LP finds out that the broker selects the latest quote, the LP will start quoting more often, which will negatively affect the network bandwidth.

Another case is when the broker believes that the quote with the biggest volume is the most reliable. The order-sweeping algorithm shall apply price/size priority; if there is a price tie between LPs, the LP with the higher volume could be chosen. If the price and volume are identical, the system can choose any LP. As the broker grows, additional logic can be added to this resolution in the future.

Often the broker selects an optimal price-time-size quote either heuristically or bound by the aggregator capabilities.

Jon Light,
VP of OTC Platform

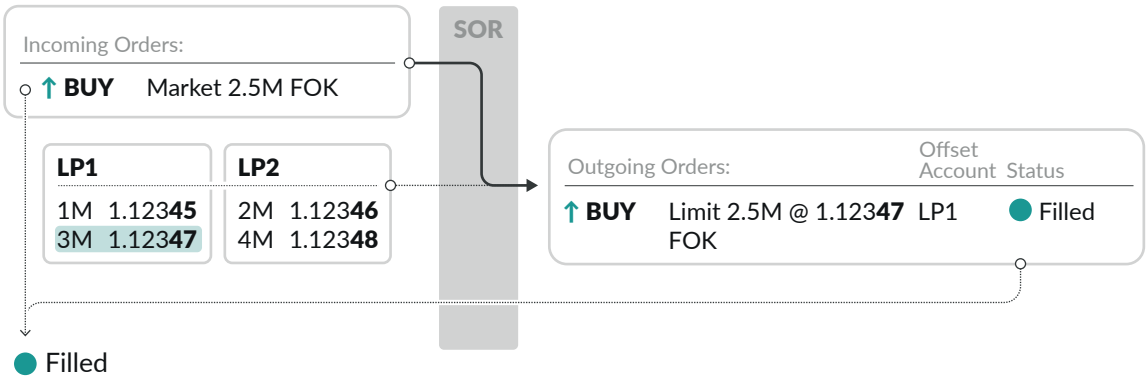
> ***But a more profound way to choose between two or more LPs showing the same quote is to assign weights to the LPs according to their previous fills or rejects. Generally, the weight is based on the amount the LPs charge or their rebates. That could result in a worse weighting if the broker was getting lots of rejects.***



Do Not Split **FOK Market** Order

In the algorithm trading practice, it is common to work with fixed-sized positions to control the risk. A broker resorts to an FOK TIF. The aggregator will search for an offer that fills the order completely in one tier.

Let's consider how the aggregator shall handle such an order:



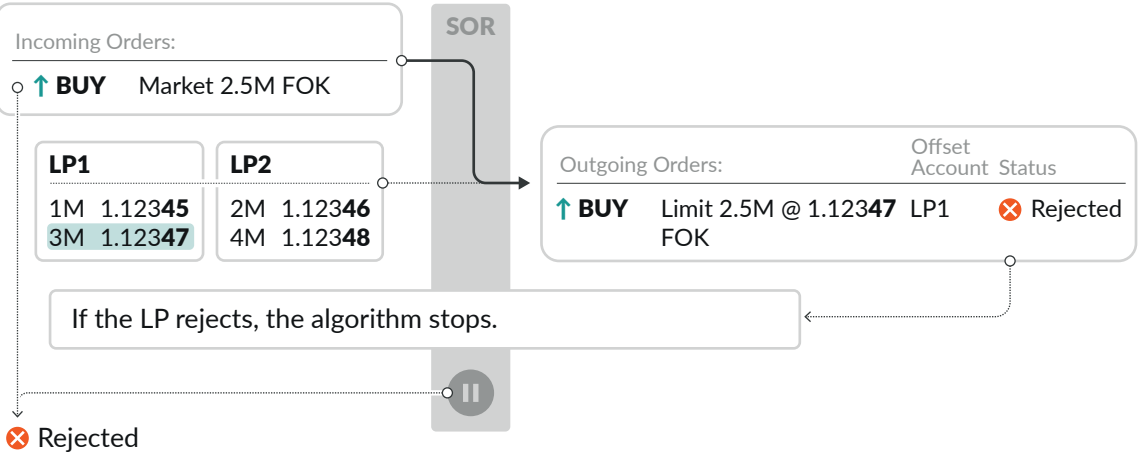
Notice that the order-sweeping algorithm placed the order with the LP that offered the first tier to fill it completely. It didn't search for the lowest price.

FOK

Scenario 11

Do Not Retry **Buy Market FOK** Order after **Rejection**

In this scenario, a broker sends a Buy Market FOK 2.5M order. The aggregator finds an LP offering a tier that ensures full order fill. The LP, though, rejects the order which leads the algorithm to stop.

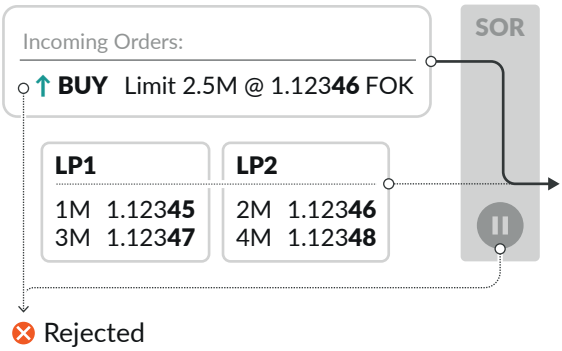


FOK

Scenario 12

Do Not Fill **Buy Limit FOK** Order If No Tiers Meet Size/ Price Constraint

When the FOK order has a limit price, the aggregator will have to find a tier to fill the complete order at or below the limit price. In this example, the broker places a Buy Limit FOK 2.5M. ads the algorithm to stop.

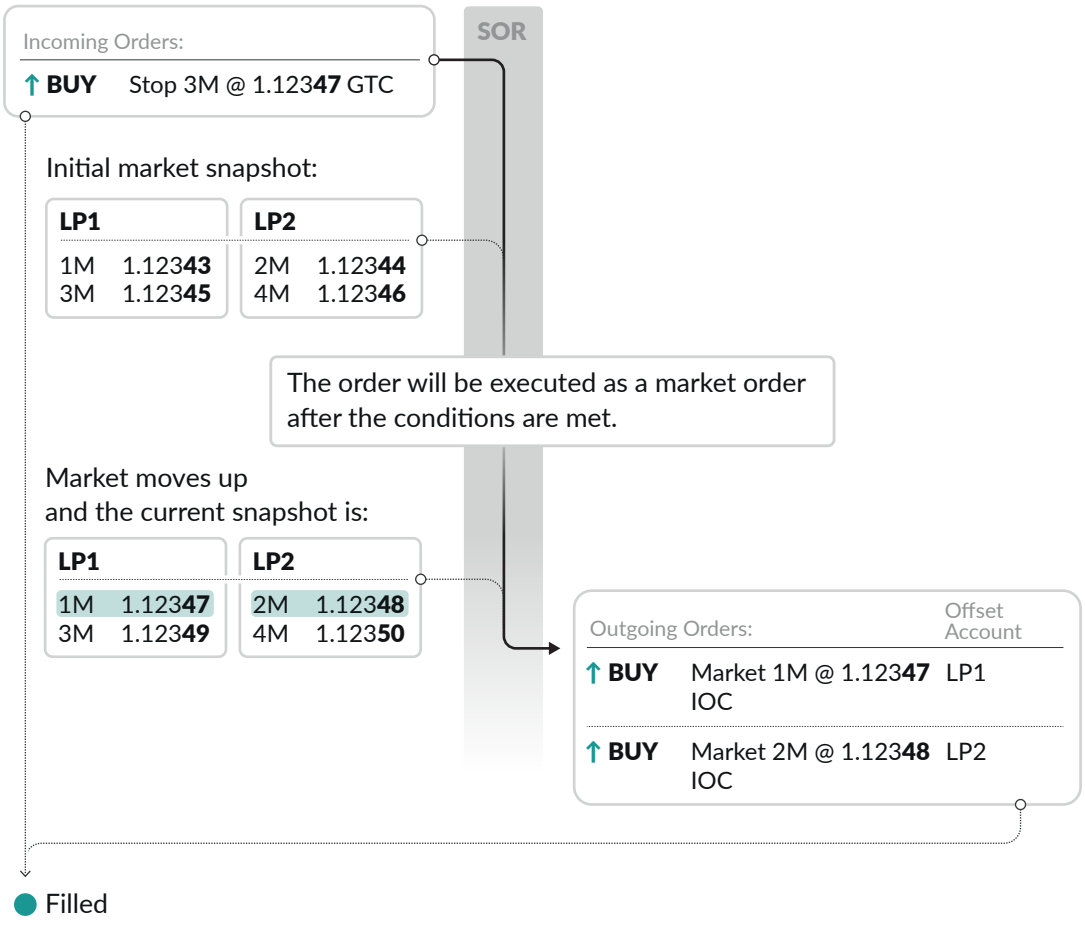


Buy Limit with FOK TIF isn't filled if there is no tier with big enough size AND appropriate price.

The order is 'killed'.

Activate **Buy Stop GTC** Order Even If There Is **Not Enough Liquidity** at the Top

The following example illustrates why slippage might happen when executing Stop orders. After receiving the stop order, the algorithm keeps waiting for when the stop order conditions are met. Once this happens the algorithm executes the order exactly as it does with market orders. For example, a broker issues a Buy Stop GTC 3M order at a stop price of 1.12347:



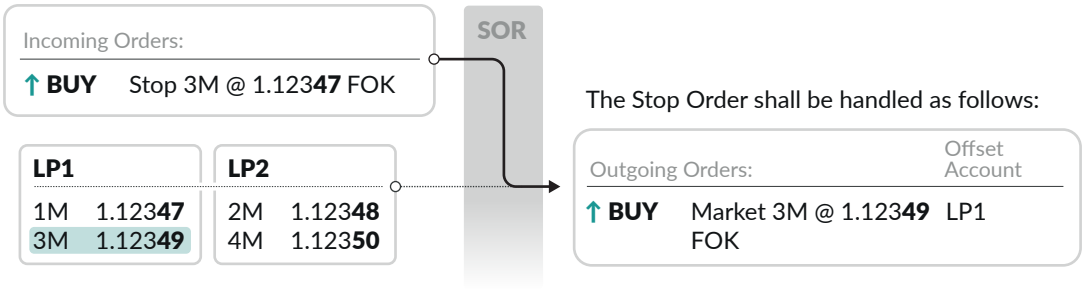
FOK Stop → LP1

Scenario 14

Send Full Amount of **Buy Stop FOK** to the Best Big Enough Size

A full amount execution happens when the aggregator sends the whole order to only 1 LP without splitting it. LPs prefer to receive the full order from a broker instead of participating in the order sweeping and competing with each other. This gives them a better overview of the market.

Implementing the 'send full amount to one LP' scenario, the broker benefits by having higher chances for execution; their LP sees the whole order size and manages the risk better. Assume we have the initial market snapshot as below:



Why would the broker prefer to execute the whole order at 1.12349? Brokers usually agree with several LPs to send them the full order in exchange to guarantee its execution. Surely, the broker could split the order and try to execute it at a better price of 1M @ 1.12347 and 2M @ 1.12348 with 2 LPs, but the risks to receive rejections would be higher, as it would take them some time to get the 'last look'. In that case, the broker could even miss the price of 1.12349 because the market can move against the broker.

Andrew Artamonov,
CTO

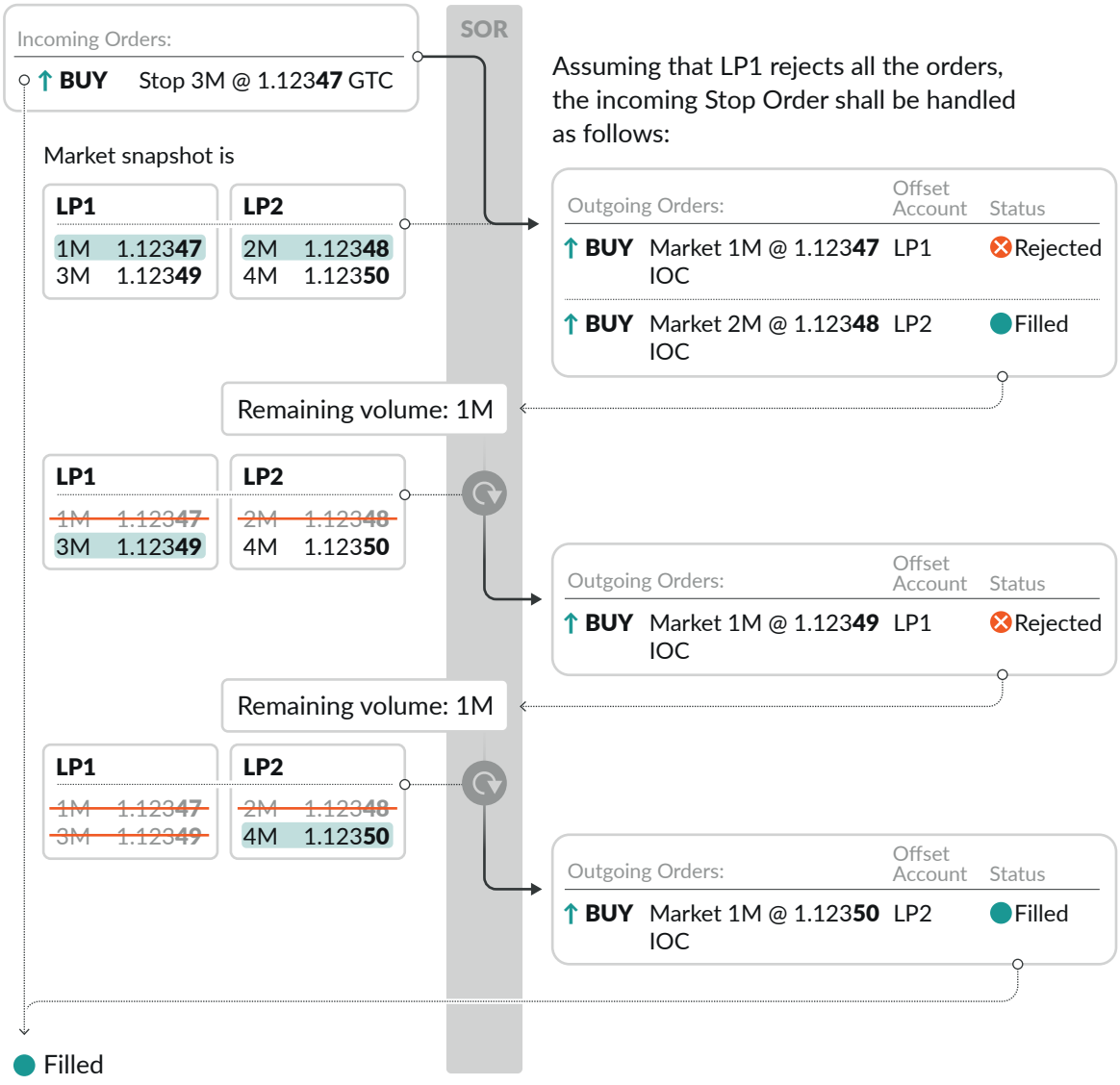
> ***In case 3M is an unusually big order size for this broker, and the LP doubts they see the full order size or wants to reconsider their risk tolerance, they have an opportunity to get the last look.***
>
> ***An LP can lose money without a last look since it may give an outdated price. This offers latency arbitrageurs an opportunity to take advantage of the price difference. Refer to p. 12 of our previous e-book "The Ins and Outs of Forex Liquidity Aggregation" to learn more about Latency Arbitrage.***

Stop LP1 LP2

Scenario 15

Execute **Buy Stop GTC** Order in Several Iterations **Due to Rejection**

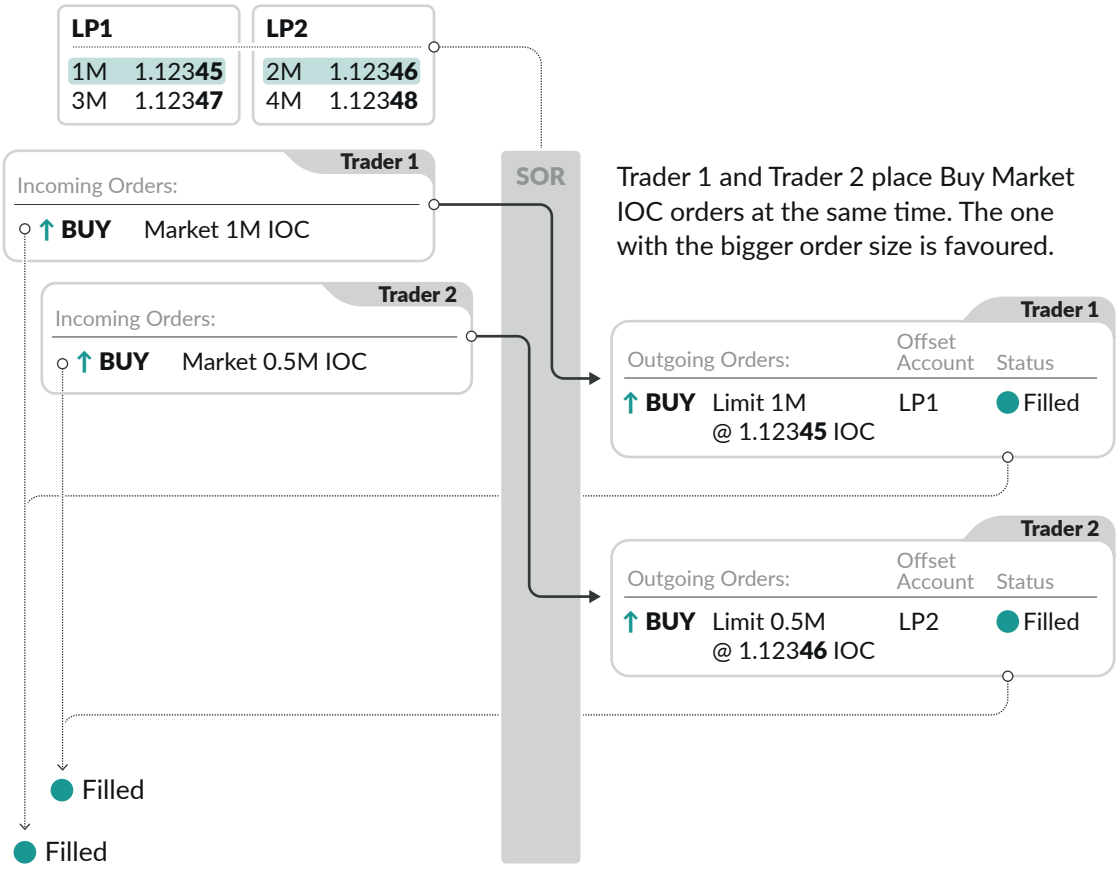
In some situations, an LP rejects all incoming orders. The algorithm memorises and updates the current market snapshot in order not to lose time on double-checking the same price tiers. For example, the aggregator receives a Buy Stop GTC 3M order:



Scenario 16

Market Depth Is Shared between Traders

In this scenario we touch upon a multi-user environment. In this environment the main challenge is to show the existing liquidity, as several clients will see it and want to use it. If a client is late, the order might be filled with a worse price.



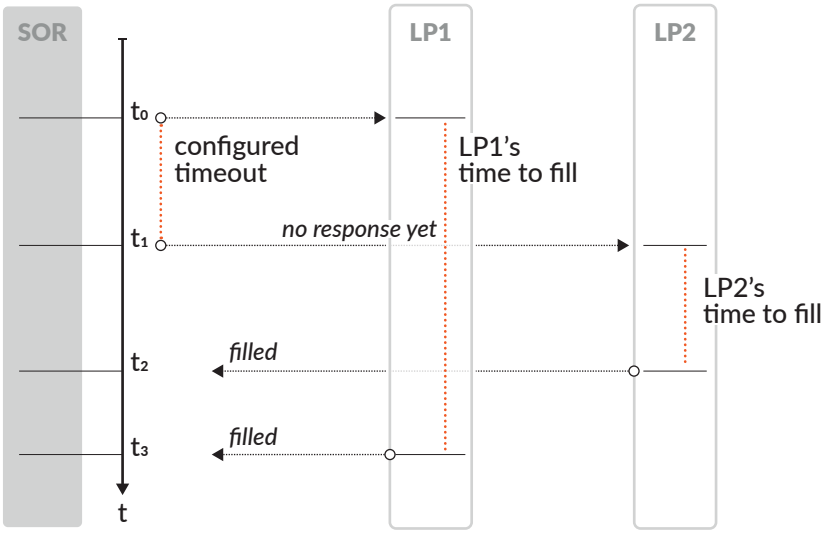
Scenario 17

No Response from the LP

If the LP doesn't respond, the order routing algorithm shall treat it the same way as if the LP sends a rejection. The algorithm would also handle it the same way as for the rejection case. The notable difference is that the status of the timed-out orders has to be different e.g. 'Missed'.

To understand when the order becomes timed-out, the TIF of the outgoing order should have an expiration timestamp, specifying how much time the algorithm should wait. The dealer calculates the appropriate time to wait for each LP empirically based on their previous experience. If this assigned waiting time was not enough, and the algorithm has already redirected to LP2, the order may be false-positively marked as not filled with LP1, and executed at LP2. The worst that could happen is having one order executed at two different LPs.

An example of when the algorithm might show a false-positive execution, in case the time was set incorrectly as too small:



The broker should promptly reconcile this situation as the LP may think the order has taken place. Sometimes, the broker can call the LP and ask to cancel the order, and no other actions are required. In other cases, the broker has to manually book an order (to assign it the filled status) and reconcile it with the LP to adjust the exposure. The broker needs to reduce exposure against the LP so that it coincides with the traders' exposure against the broker.

Andrew Artamonov,
CTO

> ***There has to be a way for the broker's operations team to 1) see a report of these exceptional cases and 2) receive alerts when this happens by email, chatbot messenger push, SMS, or other notification services.***

Scenario 18

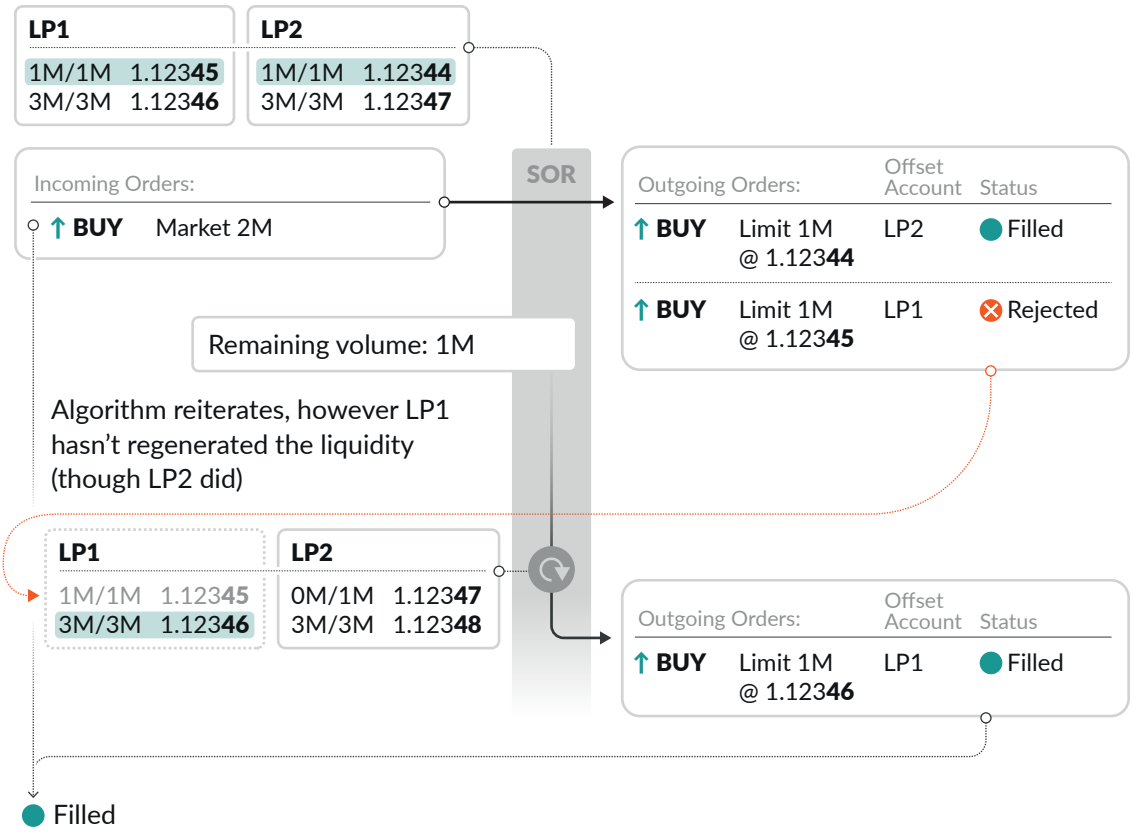
Avoiding the **Double Hit**

If an LP rejected an order, and hasn't replenished its liquidity yet, it's better to avoid hitting the same tier again. Similarly, if the order has been filled, but the prices weren't updated yet, it's better to take into account the volume we already used to minimize the rejection probability.

It is especially important while working with market makers and not as much for aggregation venues or cryptoexchanges.

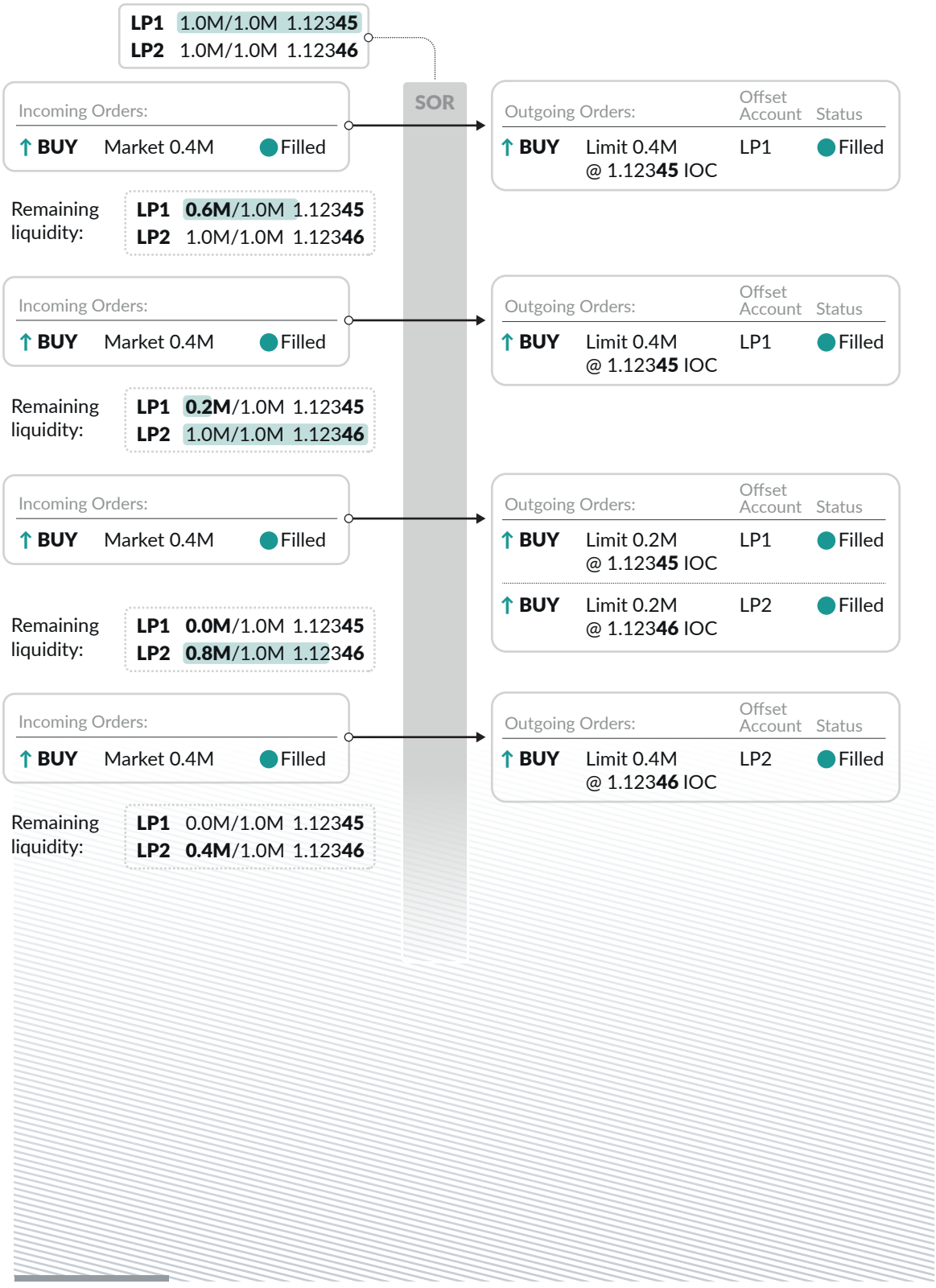
In order to make this effective, the algorithm shall track the volume tiers the LP provides and the available volume (or locked volume if it is more convenient, tier volume = locked volume + available volume).

Here, we use X/Y notation (1M/1M), where X is the volume available from the initial volume Y.



Note that even though 1.12345 is still present in the market depth, the algorithm skips it as it already hit this tier and the order was rejected before.

Here's another example with no LP regeneration: there are 4 incoming orders, Buy Market 400k each. The algorithm shall issue orders as follows:



CONCLUSION

Order management and execution management in Forex trading are much more complicated areas than one might think. This book is just the tip of the iceberg, and we hope it will show you how deep the rabbit hole is.

It is not that difficult to build an in-house forex or crypto aggregator using your own efforts, especially at the demo, concept design, or MVP stage. But if you are planning a big construction project to build a real-world high-load system to process billions of trading volume daily, it's a different story. This is where you might need professional expertise to bring your project to life.

Write to us if you decide to undertake to develop your own high-load trading system, we've been in financial software development for 20 years.

Yours sincerely,
Devexperts

About Devexperts

Devexperts is a global fintech company. We specialise in software development of trading systems for investing in stocks, futures, options, warrants, FX, and cryptocurrencies, and in market data distribution.

As technical partners of some of the world's top brokerage houses, Devexperts is committed to delivering financial technology solutions tailored to clients' fast-evolving needs. Our core trading and risk management solutions help clients gain competitive advantage by having the right data and the tools to analyse it easily and quickly.