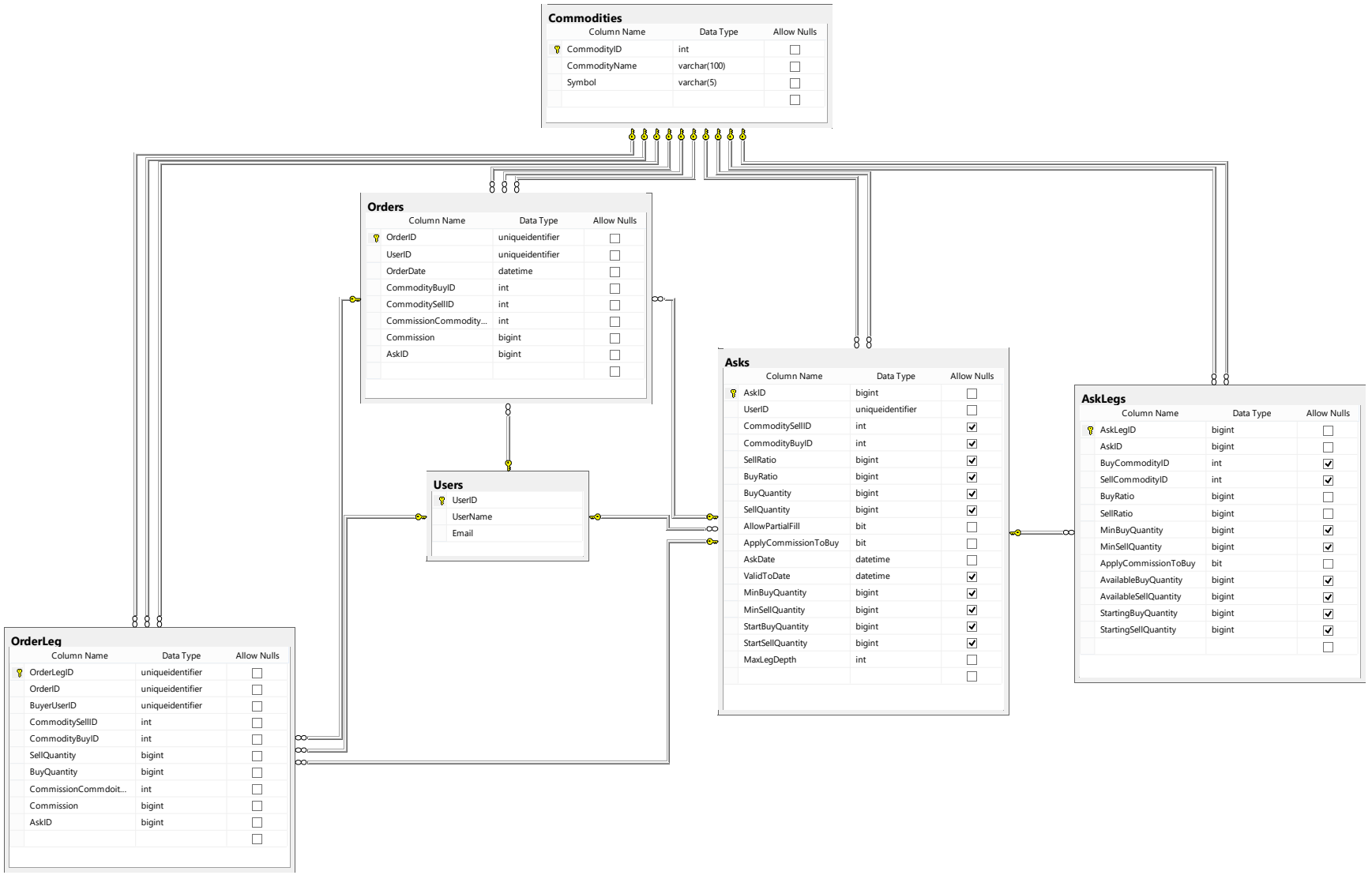
A Generalized Commodity Barter Exchange Engine

Abstract

This document describes a prototype barter exchange system where users buy and sell commodities without using an exchange medium. This system includes mechanisms for finding best available price and multi-tiered transactions.

DATA MODEL


Commodities
Column Name
Data Type
Allow Nulls
CommodityID
int
CommodityName
varchar(100)
Symbol
varchar(5)
Orders
Column Name
Data Type
Allow Nulls
OrderID
uniqueidentifier
UserID
uniqueidentifier
OrderDate
datetime
CommodityBuyID
int
CommoditySellID
int
CommissionCommodity...
int
Commission
bigint
AskID
bigint
Users
UserID
UserName
Email
Asks
Column Name
Data Type
Allow Nulls
AskID
bigint
UserID
uniqueidentifier
CommoditySellID
int
CommodityBuyID
int
SellRatio
bigint
BuyRatio
bigint
BuyQuantity
bigint
SellQuantity
bigint
AllowPartialFill
bit
ApplyCommissionToBuy
bit
AskDate
datetime
ValidToDate
datetime
MinBuyQuantity
bigint
MinSellQuantity
bigint
StartBuyQuantity
bigint
StartSellQuantity
bigint
MaxLegDepth
int
AskLegs
Column Name
Data Type
Allow Nulls
AskLegID
bigint
AskID
bigint
BuyCommodityID
int
SellCommodityID
int
BuyRatio
bigint
SellRatio
bigint
MinBuyQuantity
bigint
MinSellQuantity
bigint
ApplyCommissionToBuy
bit
AvailableBuyQuantity
bigint
AvailableSellQuantity
bigint
StartingBuyQuantity
bigint
StartingSellQuantity
bigint
OrderLeg
Column Name
Data Type
Allow Nulls
OrderLegID
uniqueidentifier
OrderID
uniqueidentifier
BuyerUserID
uniqueidentifier
CommoditySellID
int
CommodityBuyID
int
SellQuantity
bigint
BuyQuantity
bigint
CommissionCommdoit...
int
Commission
bigint
AskID
bigint

Ask Structure

An Ask is a user request for an Order of commodities. Asks must contain either a Buy or a Sell Commodity, if they contain both a Buy Ratio and Sell Ratio must be specified. If either the Buy or Sell commodity is not set corresponding Ask Legs must be set. A Sell Quantity or Buy Quantity must also be specified, if sell commodity is not set a Sell Quantity cannot be specified and the same for buy quantity and buy commodity. Sell Quantity and Buy Quantity will be adjusted for partial fill Asks as they're processed. All quantities are integers as the barter units cannot be fractional.

Asks can be a partial fill or not, asks that are not a partial fill will not be processed if the full quantity is not available on the market place at the requested buy ratio.

Min Buy Quantity can be populated if there is a Buy Commodity and Min Sell Commodity can be populated if there is a Sell Commodity. It is a good idea to populate these with partial fill asks corresponding to the commission as the commission applied to a small quantity will be vastly overstated.

Max Leg Depth defines how many commodity tiers will be attempted when finding potential orders with the best buy ratio. A higher max leg depth may result in more options and a better price but it takes longer to process. The Ask can also specify that the commission is applied to the buy commodity, if this is the case the commission will be subtracted from the order; if the commission is applied to the sell commodity the seller must provide additional commodities. It is generally a good idea for positive buy ratios to include the commission in the buy because the commission percentage is rounded up and the greater the quantity the closer to the commission is to the ideal rate.

Start Buy Quantity or Start Sell Quantity will be populated if the corresponding Sell or Buy Quantity is populated, this is for accounting and reconciliation purposes. Ask Date will also be automatically populated at the time of Order request.

Ask Leg Structure

Asks' can have ask legs, this allows an order request where the user is buying a commodity but willing to sell multiple commodities or selling a commodity and willing to buy multiple commodities. The Ask Leg's Buy or Sell Commodity must be set, and the corresponding Ask's Buy or Sell commodity must be not set. The Buy Ratio and Sell Ratio must be set, the ratio is in terms of that leg to the parent ask.

Optionally Available Quantity Buy or Available Quantity Sell (depending on if it's a buy or sell ask) can be set to inform the engine the max quantity that can be transacted, these are updated as orders are processed. If they are set the corresponding Starting Quantity will be set for accounting purposes.

Like the parent ask, Apply Commission to Buy can be set.

Order Structure

An order is a transaction for a user buying a commodity and selling another. The Order is tied back to an Ask for accounting purposes. It contains a reference to the Commodity being bought, sold and commissioned and the commission for the order.

Order Leg Structure

Order Legs are the actual transactions comprising an order, an order must have at least one leg and must have at least one leg with the buy commodity of the order and the sell commodity of the order. Order legs define the

commodity being sold and bought, the sell quantity and buy quantity, the commission commodity that the seller is paying and the commission quantity and a reference to the seller.

EXAMPLE

Assume all are partial fill.

EXISTING ASKS

Buy Commodity	Sell Commodity	Buy Ratio	Sell Ratio	Buy Quantity	Buy Ratio
Gold	Silver	20	1295	1000	0.015
Gold	Silver	15	1295	1000	0.012
Silver	Platinum	1466	20	1000000	73.3
Platinum	Copper	3	1466	100000	0.002

ASK TO EXECUTE

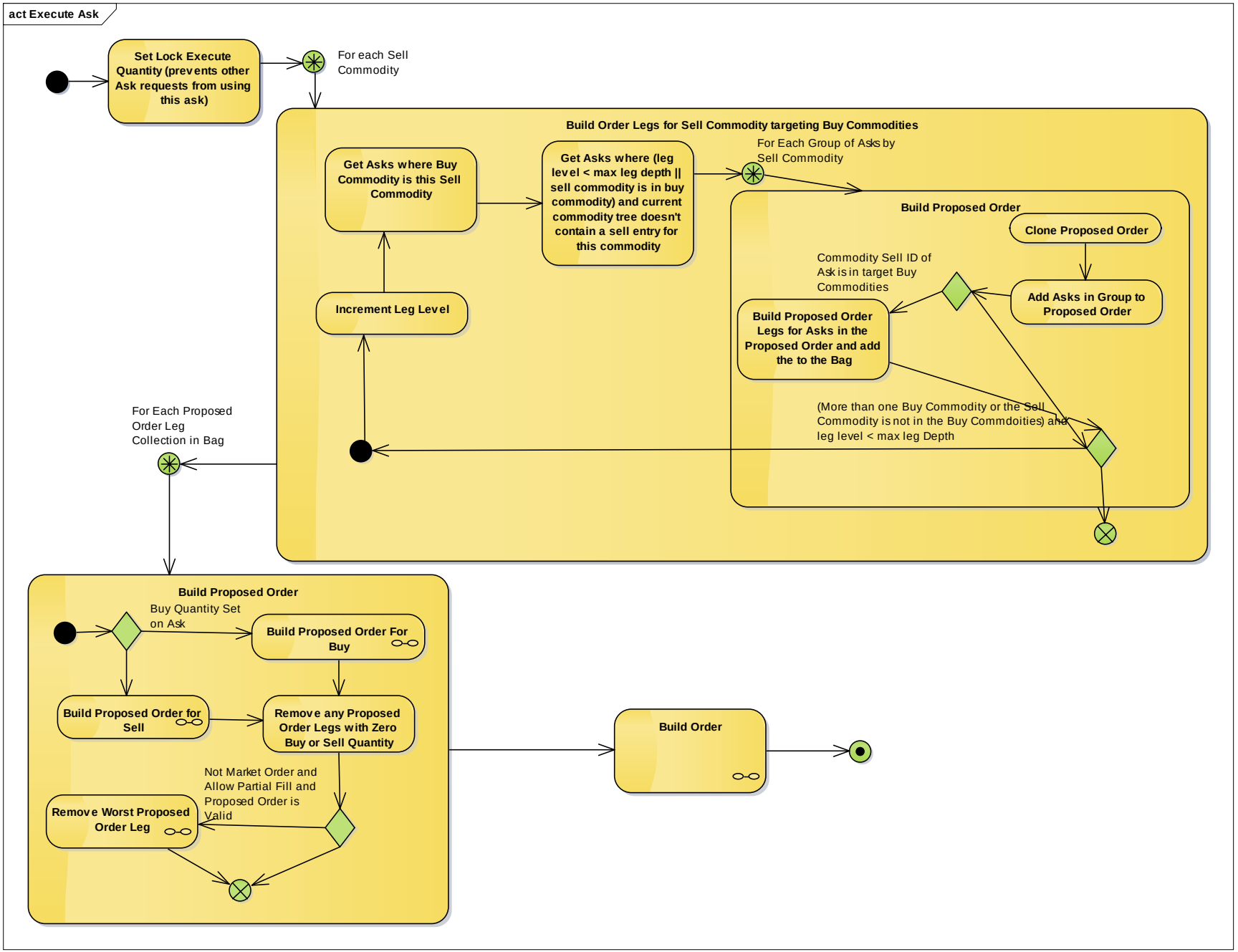
Buy Commodity	Sell Commodity	Buy Ratio	Sell Ratio	Buy Quantity
Copper	Gold	1294	3	1000000

EXECUTED ORDER LEGS

Buy Commodity	Sell Commodity	Buy Quantity	Sell Quantity	Sell Ratio
Silver	Gold	86333	1000	0.012
Silver	Gold	63713	984	0.015
Platinum	Silver	2047	150046	73.3
Copper	Platinum	1000000	2047	0.002

The Ask was successfully executed with 1984 gold sold for 1000000 copper. With a buy ratio of 504 versus the 431 in the quote, this is because there is a favorable ask in there for Gold for Silver that was prioritized by the engine.

ALGORITHM


act Execute Ask
Set Lock Execute Quantity (prevents other Ask requests from using this ask)
For each Sell Commodity
Build Order Legs for Sell Commodity targeting Buy Commodities
Get Asks where Buy Commodity is this Sell Commodity
Get Asks where (leg level < max leg depth || sell commodity is in buy commodity) and current commodity tree doesn't contain a sell entry for this commodity
For Each Group of Asks by Sell Commodity
Build Proposed Order
Clone Proposed Order
Commodity Sell ID of Ask is in target Buy Commodities
Add Asks in Group to Proposed Order
Build Proposed Order Legs for Asks in the Proposed Order and add the to the Bag
Increment Leg Level
(More than one Buy Commodity or the Sell Commodity is not in the Buy Commdoities) and leg level < max leg Depth
For Each Proposed Order Leg Collection in Bag
Build Proposed Order
Buy Quantity Set on Ask
Build Proposed Order For Buy
Build Proposed Order for Sell
Remove any Proposed Order Legs with Zero Buy or Sell Quantity
Not Market Order and Allow Partial Fill and Proposed Order is Valid
Remove Worst Proposed Order Leg
Build Order

HIGH LEVEL

The first step in the algorithm of executing an ask or getting a quote is setting the asks Lock Execute Quantity equal to the available quantity of the Ask. Available Quantity is defined as the Ask's Buy or Sell Quantity minus Lock Quantity (the amount currently being used by other potential orders) minus Lock Execute Quantity.

Then the engine attempts to find commodity trees which will represent Proposed Orders. For each Commodity that can be sold by the Ask it will enter a recursive function that builds a tree of commodity buy/sell combinations that result in a sale of the commodity id and a buy of one of the commodities that can be bought. This is described in the diagram above under "Build Order Legs for Sell Commodity targeting Buy Commodities".

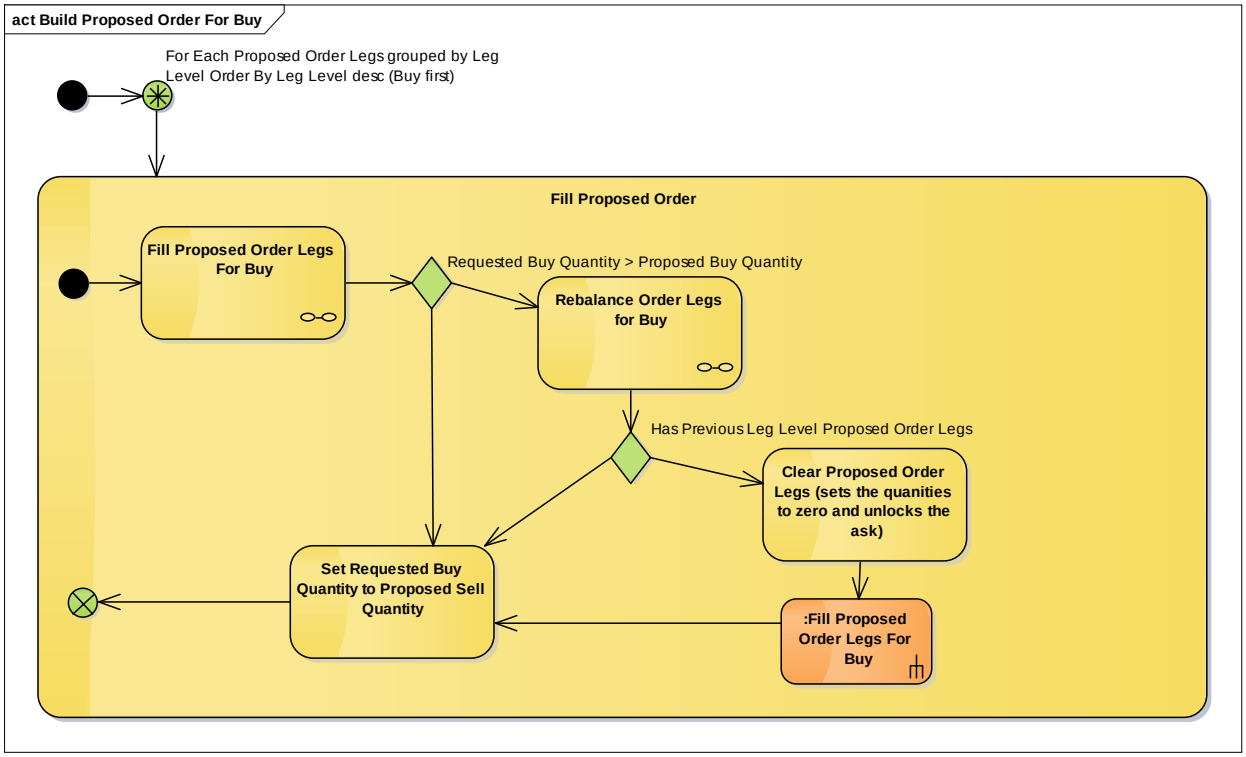
Then for each commodity tree it will attempt to build a Proposed Order. Depending on whether the ask being executed or quotes has a buy or sell quantity it will enter different logic to build the initial Proposed Order.

For non-market orders that allow partial fill and are valid the engine will attempt to remove Proposed Order Legs until the Proposed Order's Buy Ratio (Proposed Quantity Buy / Proposed Quantity Sell) is greater or equal to the Ask's Real Buy Ratio (Buy Ratio / Sell Ratio). Valid Orders are defined as those where the Order Quantities are in sync (the corresponding buy and sell order leg quantities for the same commodity are equal), there contains at least one order leg that has a Sell Commodity in the Ask's Sell Commodities, there contains at least one order leg that has a Buy Commodity in the Ask's Buy Commodities and Proposed Quantity Buy and Sell are greater than zero.

After the proposed orders are built the final order will be built from the best (by Buy Ratio) Proposed Orders. This will be detailed later.

If it is an execute the locks will be realized and the available quantities on the Ask and Ask Legs adjusted according to the order.

BUILD PROPOSED ORDER FOR BUY

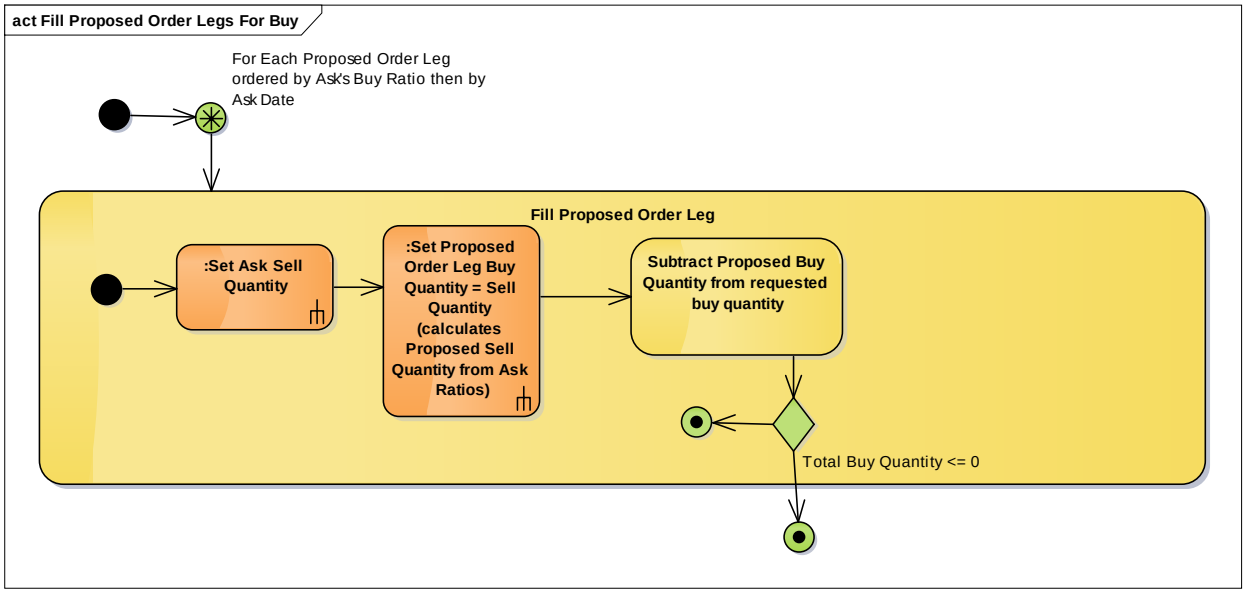


For a buy ask the Buy Quantity must be applied to the commodity tree first since the final Buy Ratio is market driven and therefore the Sell Quantity is derived from the order itself. The engine groups the Proposed Order Legs by Leg Level and orders them descending; leg level is an indicator of where in the tree the Proposed Order Leg is with 1 being a sell and the max number being the buy. The Requested Buy Quantity for the first iteration is the order quantity of the order.

The First step is to “Fill Proposed Order Legs for Buy”, this will attempt to fill, based on each Proposed Order Leg’s Ask’s Available Quantity, the order from a buy perspective. If Requested Buy Quantity is greater than the sum of Proposed Buy Quantities of the Filled Proposed Order Legs then the order must be rebalanced, unless it’s the first iteration and then it is not necessary. The “Rebalance Order Legs for Buy” method iterates over all previously built Order Leg groups and adjusts their quantities downward to reflect the short fall in the current iteration. The current iteration’s Proposed Order Legs are cleared and recalculated from the new sum of the previous order leg’s sell quantity.

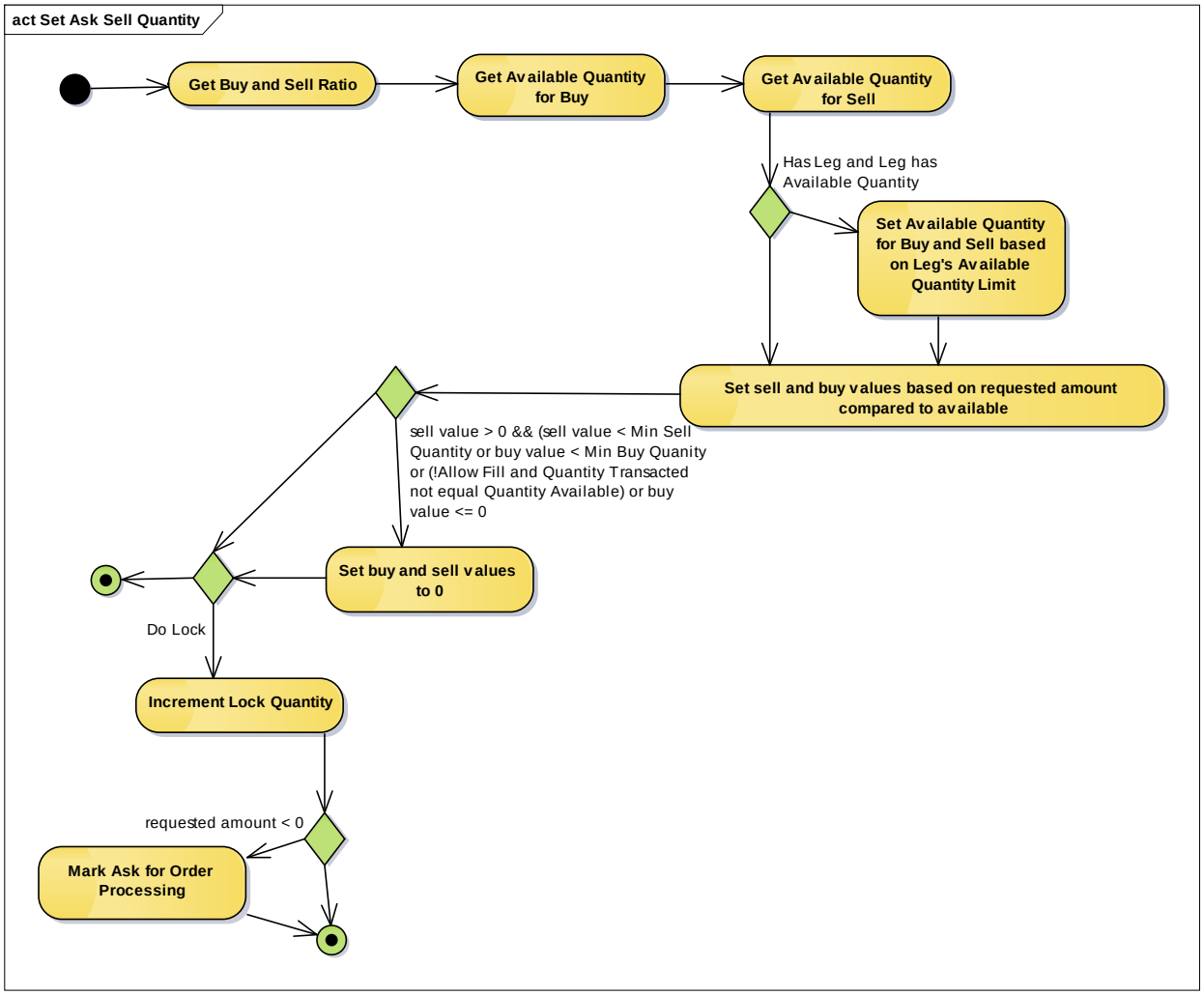
The Requested Buy Quantity variable is then set to the Proposed Sell Quantity of the current iteration and the next group of legs is processed.

FILL PROPOSED ORDER LEGS FOR BUY



The Total Buy Quantity starts out at the target buy quantity for the Order Leg from the previous iteration or initial Ask Quantity. The engine orders the Proposed Order Legs by the Order Leg's Ask's Buy Ratio (lower being more favorable to the requester of the quote) and then by Ask Date and iterates over them. It attempts to set the Ask's Sell Quantity equal to the Buy Quantity of the Order Leg, the Ask is in reverse because it is the other side of the order leg being executed. The Proposed Order Leg's Proposed Buy Quantity is then set to the quantity calculated in the "Set Ask Sell Quantity". The Proposed Buy Quantity is then subtracted from the requested buy quantity and processing continues until the Requested Buy Quantity is zero or there are no more legs to fill.

SET ASK SELL QUANTITY



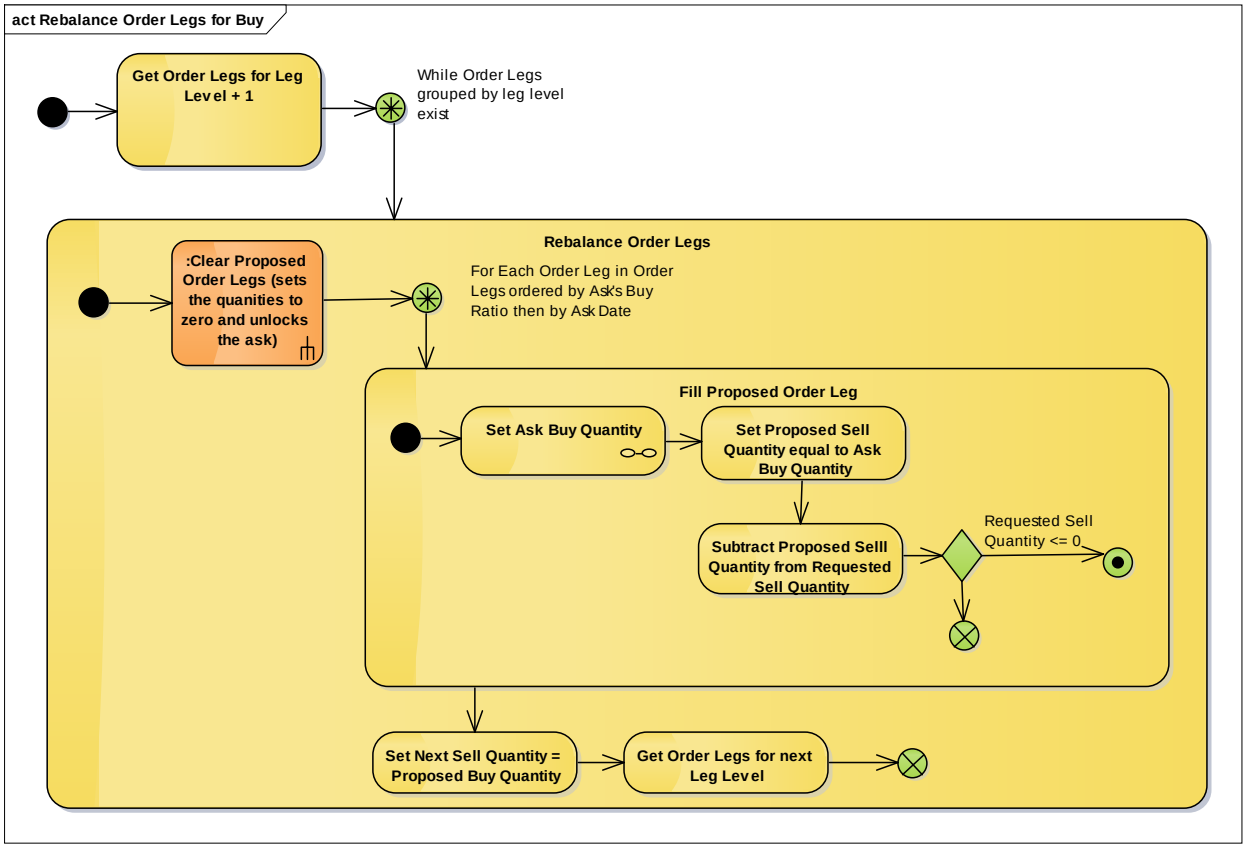
The first step is to get the Buy/Sell Ratio, which is either on the Ask or the Ask Leg. Then the available quantity buy/sell is set, if the Ask is a buy the buy available quantity comes from the set value and the sell is calculated from that, and the opposite if the Ask is a Sell. Then if the Ask has a Leg and Available Quantity on the Leg the available quantities are adjusted accordingly.

After the available quantities are set the target value is compared to the available sell quantity and the lesser of the two is applied as the value to get set. The buy value is then calculated from the sell value using the buy/sell ratio. Then the engine checks the sell value against the Min Sell Quantity and buy value against Min Buy Quantity and checks if the Ask is not a Partial Fill and the buy or sell quantity is equal to the available quantity and buy value is greater or equal to zero. If all those conditions are met the values are left alone otherwise they are set to zero.

If the set request is set to lock the lock quantity of the Ask is incremented by the transacted quantity. If the request has a leg then that leg's lock quantity is incremented as well.

If the requested quantity is < 0 then an event is raised to mark the Ask for additional processing. The engine will eventually execute this ask again, in case while it had a locked amount an Ask was executed that matches this Ask.

REBALANCE ORDER LEGS FOR BUY

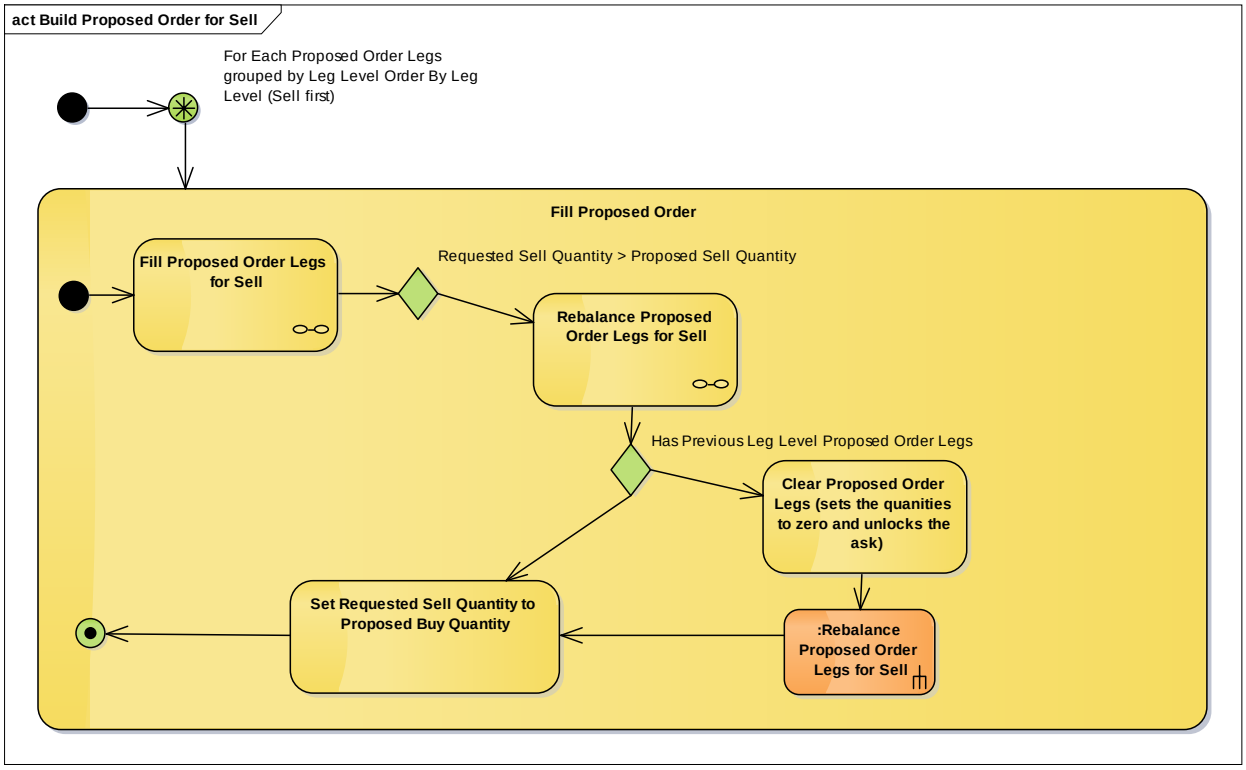


The rebalance starts by fetching the Proposed Order Legs for the previous level and then entering a loop that will iterate over all previous Proposed Order Legs grouped by leg level.

Inside that loop it will clear the Proposed Order Legs, this will set all quantities on the proposed order equal to zero and release any locks on the order legs' asks'. The engine will then iterate over the Proposed Order Legs in the Proposed Order ordered by the Ask's Buy Ratio. It will attempt to set the Ask's Buy Quantity equal to the requested Sell Quantity of the previous Leg Level's buy quantity and then set the Proposed Order Leg for that Ask's Sell Quantity equal to the Buy Quantity calculated. The Proposed Sell Quantity is then subtracted from the requested Sell Quantity and if the Requested Sell Quantity is less than or equal to zero no more Proposed Order Legs are processed.

The next requested sell quantity is then set to the sum of the Proposed Order Leg's buy quantities and the next leg level's Proposed Order Legs are fetched and the loop continues.

BUILD PROPOSED ORDER FOR SELL

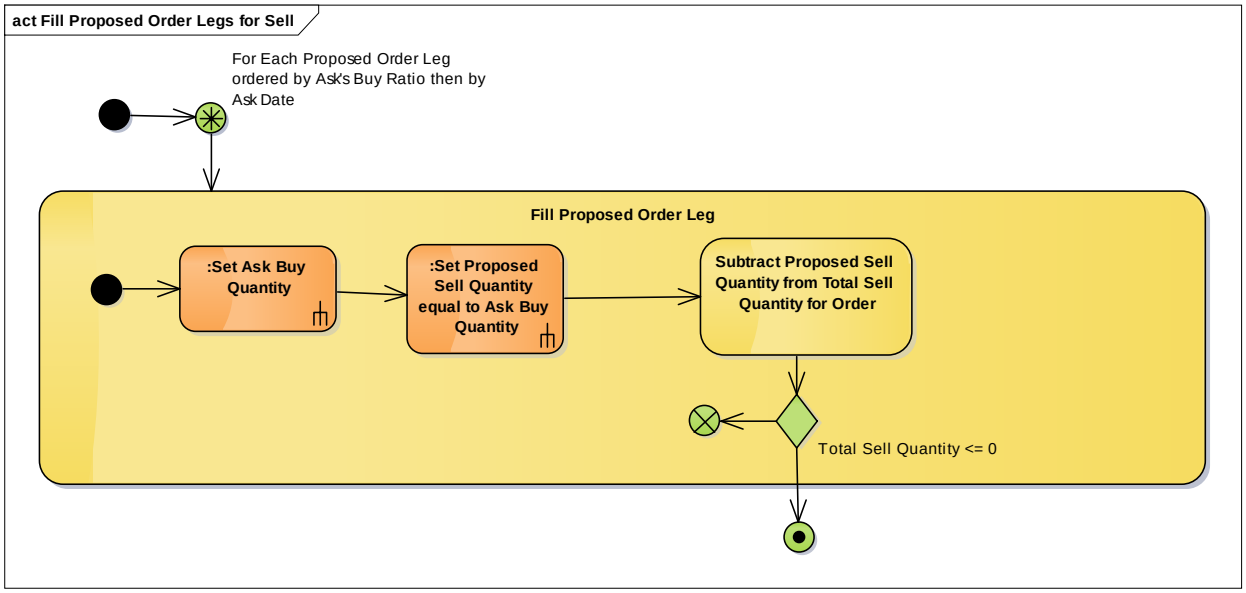


For a sell ask the Sell Quantity must be applied to commodity tree first since the final Buy Ratio is market driven and therefore the Buy Quantity is derived from the Order itself. The engine groups the Proposed Order Legs by Leg Level and orders them ascending. Please refer to "Build Proposed Order for Buy" for a description of Leg Level. The Requested Sell Quantity for the first iteration is the order quantity of the order.

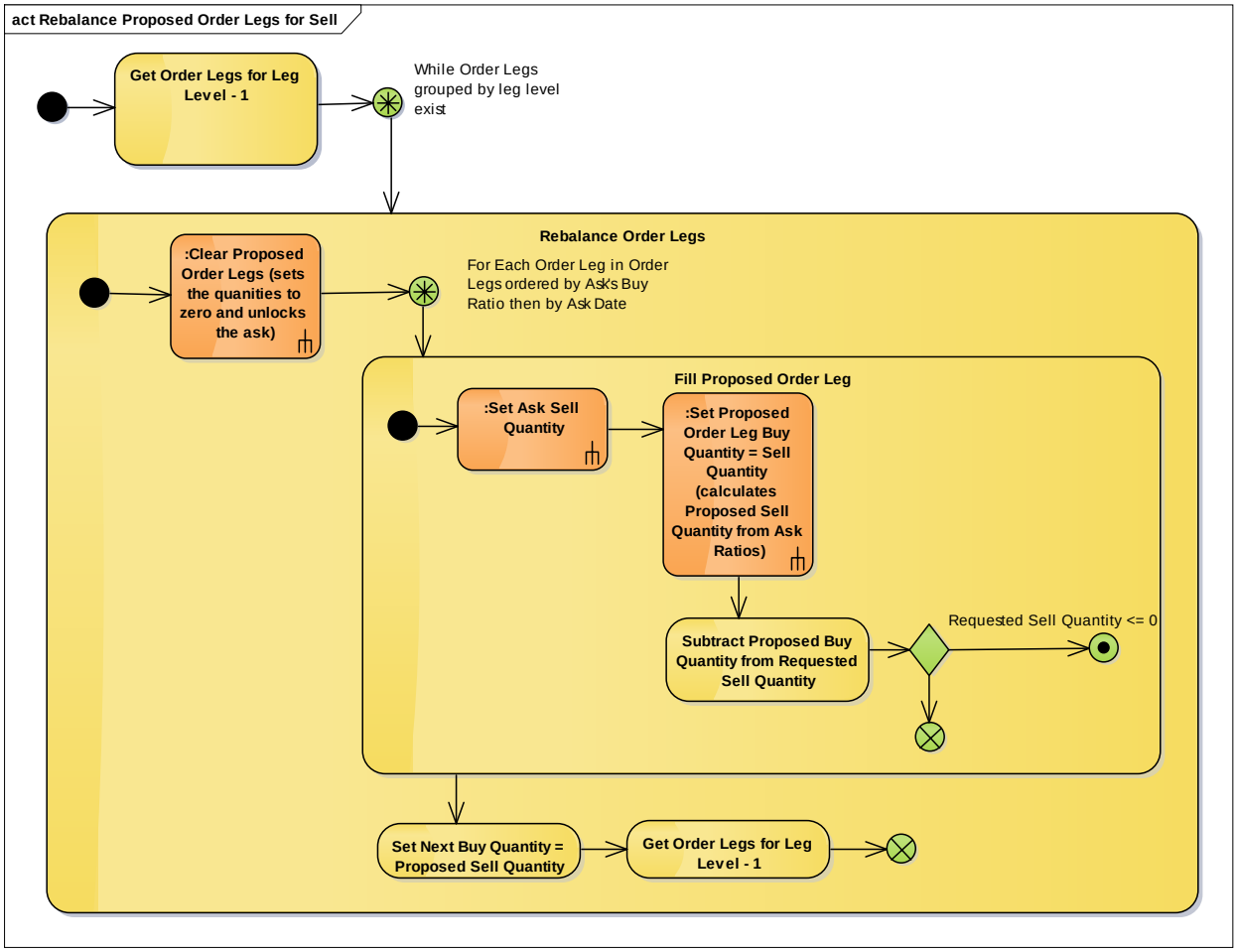
The first step is to "Fill Proposed Order Legs for Sell", this will attempt to fill, based on each Proposed Order Leg's Ask's Available Quantity, the order from a sell perspective. If Requested Sell Quantity is greater than the sum of Proposed Sell Quantities of the Filled Proposed Order Legs then the order must be rebalanced, unless it's the first iteration and then it's not necessary. The "Rebalance Order Legs for Buy" method iterates over all previously built Order Leg groups and adjusts their quantities downward to the reflect the shortfall in the current iteration. The current iteration's Proposed Order Legs are cleared and recalculated from the sum of the previous order leg's buy quantity.

The Requested Sell Quantity variable is then set to the Proposed Buy Quantity of the current iteration and the next group of legs is processed.

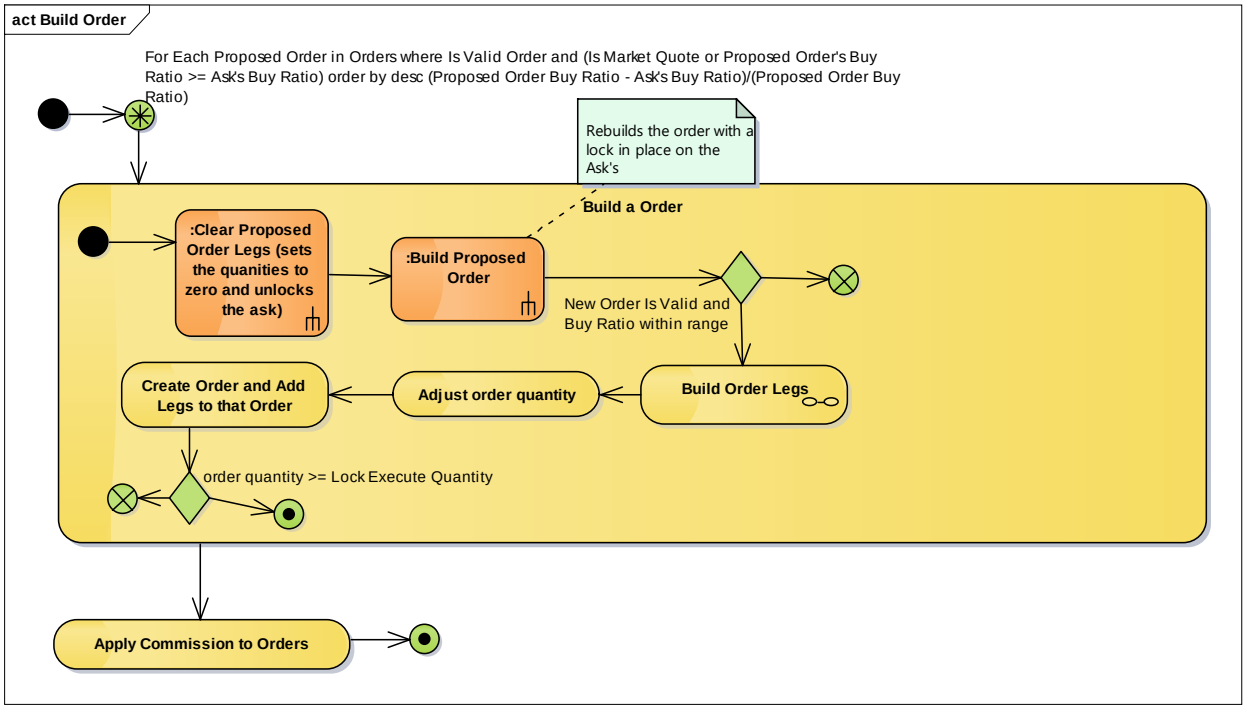
Fill Proposed Order Legs for Sell



Rebalance Proposed Order Legs for Sell



BUILD ORDER

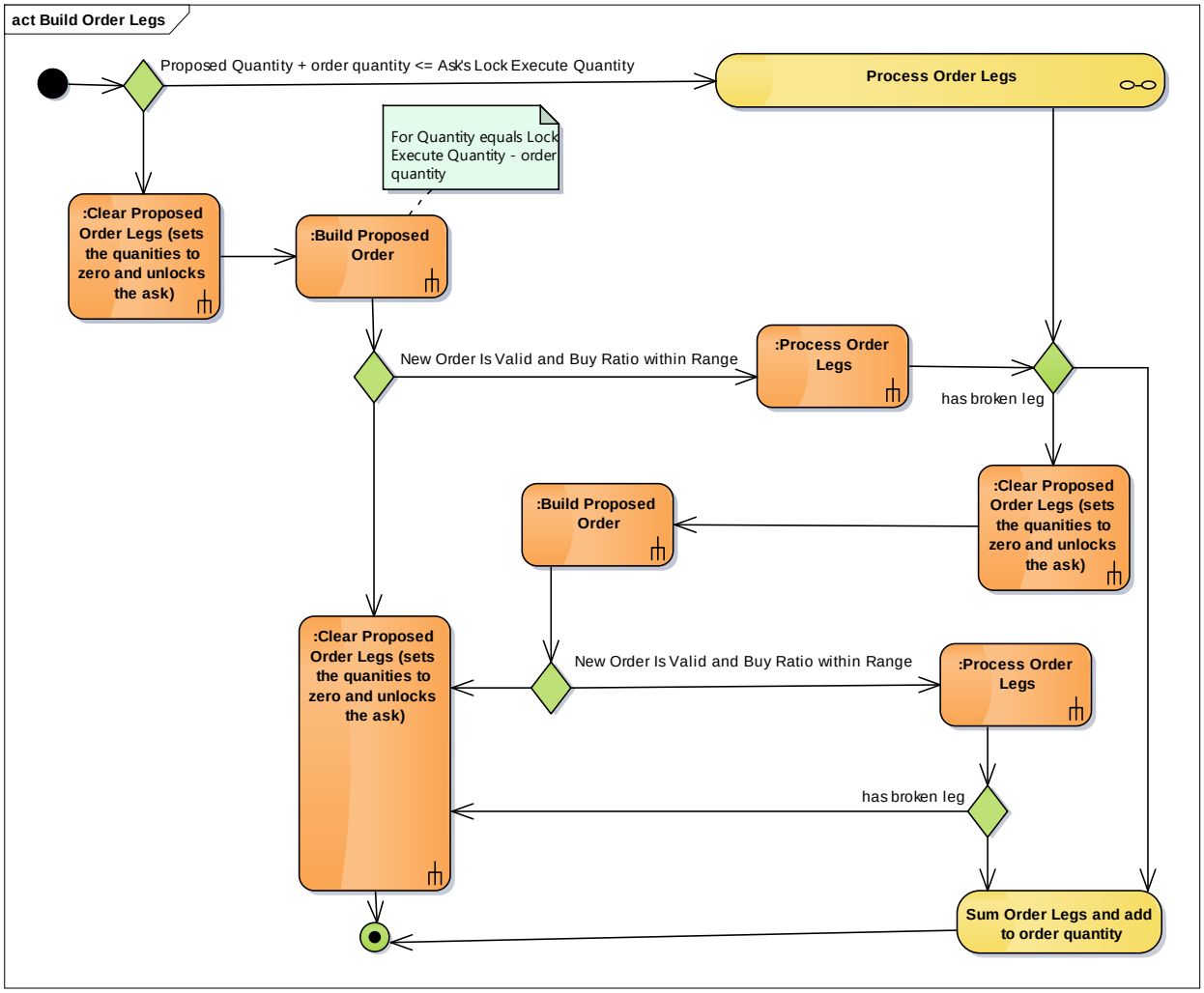


For Each Proposed Order that is valid and has a Buy Ratio greater or equal to the executing Ask's Buy Ratio for that commodity buy/sell pair ordered descending by the difference in buy ratio from the Ask's Buy ratio for that commodity buy sell pair an Order is built from that Proposed Order.

The first step of Build an Order is to Clear the Proposed Order Legs and then build a new Proposed Order, this time locking the underlying Asks. The engine then checks to ensure the new Proposed Order is Valid and the Buy Ratio is within range and then Build Order Legs, adjust the Order Quantity (the buy or sell amount of the order) and then Create the Order and Add Legs to that Order. If Order Quantity is greater or equal to Lock Execute Quantity of the Ask no more Orders will be created.

The last step is to apply the commission to the Orders.

BUILD ORDER LEGS



PROCESS ORDER LEGS