BMLL Data Feed

Level 3

Date: 2024-11-27
Version: 2.3

1.1. Introduction

The Level 3 dataset offers an order by order view of the order book. Showing every single update to the order book, ilt provides an in-depth understanding of historical market activity, enabling users to make more informed trading decisions.

1.2. Dataset schema

Basic Types

Type	Content	Text representation
bigint	8 byte signed integer	103403403403
char(n)	fixed length string containing letters, digits or '@'	XLON
date	Calendar date	YYYY-MM-DD
enum	string from a defined set	AUCTION
integer	4 byte signed integer	1234
price	8 byte binary floating point. All prices are rounded to a fixed number of decimal places. Trailing decimal zeros are not shown.	123.45
quantity	8 byte binary floating point, rounded to 4 decimal places	125
timestamp	Nanoseconds since 1970 in UTC. Timestamp will be represented in ISO format in text format.	20220302-11:23.57.123 45789
varchar	variable length string up to 500 characters. May contain quotes and special characters. All text will be utf8-encoded unless indicated differently. Ticker and ISO codes should be ASCII compatible.	“Vodafone Group”
bool	Boolean	1, 0

1.3. Schema

One row in the dataset corresponds to a single update of the limit order book during the trading day. An update consists generally of the addition of an order to the limit order book, the removal of an order book of an update on an existing order. A single trading order can cause multiple updates on the limit order book - for instance, a single aggressive sweep order may clear multiple orders that were passively resting on the book.

Whenever we receive that information, we will group updates of the order book related to the same trading order to a unique event number - and this is designed to identify the state of the order book that is actually being interacted with.

Unless preceded by "Old", columns will always represent the value after the event.

Field Name	Type	Description
ExchangeTicker	varchar (ascii)	Ticker as provided by the reporting venue.
ListingId	bigint	The BMLL identifier represents the listing
InstrumentId	bigint	If not 0, the BMLL identifier represents the instrument to which the listing belongs. BMLL Instrument IDs are different if the primary identifier is different, the currency or the region is different.
TradeDate	date	The specific date on which a trade was executed - the date is derived from the LocalTimestamp.
TimestampNanoseconds	bigint	This column contains a representation of the time of the update as the number of nanoseconds since 1970-01-01 00:00:00.0 UTC.
EventNo	integer	An increasing ID per symbol, used to identify atomic updates to the orderbook. Multiple rows can share one EventNo - these should be treated as one update when rebuilding an order book.
Side	integer	Side of the order update 2: Order is updated on the BID side 1: Order is updated on the ASK side

LobAction	enum	What has changed to the order INSERT: A new order added to the book. REMOVE: An existing order on the book is removed, due to either cancellation or order execution. UPDATE: An existing order on the book is modified. Rows with an unknown value of LOB action should be skipped for the purpose of building the order book
OriginalOrderId	varchar	The original ID of the order inserted into the order book. Note that in some exchanges, order Ids can change (e.g due to a priority shift). This ID allows you to track modified and updated orders throughout the day
Price	float	The price of the order after the update. Null if order is removed
Size	int	The size of the order after the update. Null if order is removed
OrderId	varchar	The order ID after the update. Empty if order is removed
OldPrice	float	The price of the order after the update. Null if order is inserted
OldSize	int	The size of the order after the update. Null if order is inserted
OldOrderId	varchar	The order ID before the update. Empty if order is inserted
OrderExecuted	bool	Indicates if the order has been removed due to an execution (rather than a cancellation)
ExecutionPrice	bool	The price that the order executed at. This can be different to the price the order was placed at (e.g during auctions)
ExecutionSize	bool	The size that the order executed at. This can be more than OldSize (e.g with an Iceberg order)
TradeId	varchar	The ID of the trade, if provided by the exchange
PriceLevel	int	The price order of the level after the instruction (1 = best price) where applicable. For a REMOVE event, equals the OldPriceLevel.
OldPriceLevel	int	The level of the order before the instruction (1 = best price) where applicable. For a LobAction of INSERT, this is the price level of the inserted order if the level previously existed, or 0 otherwise. For LobAction of UNKNOWN this is 0.
SizeAhead	Int	For LobAction = INSERT or UPDATE, the total size (quantity) of the orders ahead of the given order in the priority queue, after the event. For LobAction = REMOVE, this is -1.

OrdersAhead	Int	For LobAction = INSERT or UPDATE, the number of the orders ahead of the given order in the priority queue, after the event. For LobAction = REMOVE, this is -1.
PosChange	bool	Indicates whether the order has changed priority with the update
EndOfEvent	bool	Indicates whether this is an end of event
MarketState	enum	The market state of the update
Printable	bool	Indicates whether the execution is printable. Printable is used to stop double counting (e.g auction executions)
DelOrderIndex	int	Used in the order book rebuild. Provides the position of the order in all priority before the order book update -1 for an INSERT
AddOrderIndex	int	Used in the order book rebuild. Provides the position of the order in all priority after the order book update. -1 for a REMOVE
MPIDAttribution	varchar	The market participant id (if provided by the exchange)
ExchangeSequenceNo	bigint	The sequence number of the market state update, if provided by the exchange.
BMLLSequenceNo	bigint	A synthetic sequence number as provided by BMLL. This ensures the correct ordering of all messages from a particular source and should be used as the standard to sort and join all messages.
BMLLSequenceSource	bigint	Source of the data, where multiple order book sources are provided together. Example would be out of hours orderbook and main session (e.g Borsa Italiana)
OrderType	int	The type of order 0: Unknown 1: Limit 2: Market
OriginalExchangeMessage	varchar	The original message type

1.4. Market State Normalisation

BMLL provides a consistent normalisation for market state messages, in order to make cross-venue analysis quick and easy. For full details, please refer to the BMLL Data Feed: Market State documentation.

1.5. File format and delivery structure

Data is delivered as a single file per market per date, as a parquet file.

Equity-L3/{MIC}/{yyyy}/{mm}/{dd}/l3-{MIC}-{yyyy}{mm}{dd}.parquet

1.6. Data arrival times and coverage

Full product coverage and arrival times are available at https://data.bmlltech.com.

1.7. Frequently Asked Questions

We can notice some orders that execute although they were not at the top of the book - is this expected ?

Many venues have a special type of orders called `Discretionary Orders`. These orders will be sitting on a book at a predefined price but may have additional instructions, which can correspond to a second hidden limit against which the order may go aggressive. These instructions will often cause the order to sit passively on the book and, then, to execute before those that were before other orders of the order book, they can generally be identified by the fact that the execution price was ultimately different from the price at which they were sitting.

Another situation that can occur is that a locked NBBO quote protection may prevent the immediate of execution of orders on venue (orders flagged as `Do Not Route` (DNR) will stay blocked if the NBBO is crossed), and as NBBO unlocks, and the book evolves the order the execution that was previously blocked by a locked NBBO, orders may adapt their price, matching may become acceptable, and matched execution may then occur though book that has evolved - but where order previously present on the book may benefit from specific rules allowing them to jump back ahead.

We can see some orders increasing their size and keeping their position in the queue - how is it possible ?

In US markets, Reg NMS and best-execution obligations have technical implications that are important to understand when looking at limit order book data: An order or a part of an order that was sitting on the book may need to be routed from one venue to another venue to ensure best-exchange based on protected quotes. However, as routing and information dissemination inevitably take time to be transferred between trading venues, it can sometimes happen that, when the routed order arrives on the venue, the quantity is not available anymore as other venues may have also routed orders to the most competitive venue. When this happens, the quantity will be routed

back and reinserted back in the limit order book of the original venue - when this quantity comes back, it is reinserted where it was (without losing its priority). This feature relies on the PosChange which is populated based on exchange data for BATS, BATY, EDGX, EDGA and EPRL.

We can see orders being added during trading halts - is this expected?

Yes, this can occur for multiple reasons, though many venues will reject order entry during trading halt, this is not always the case and some venues will still accept order change during halt. Also, even if order entry is closed during that period, there can be race conditions, and as trading halt can occur for multiple reasons, and as trading venues need to ensure the book is in a valid state before resuming trading - they may need to adapt the book with latest updates before this.