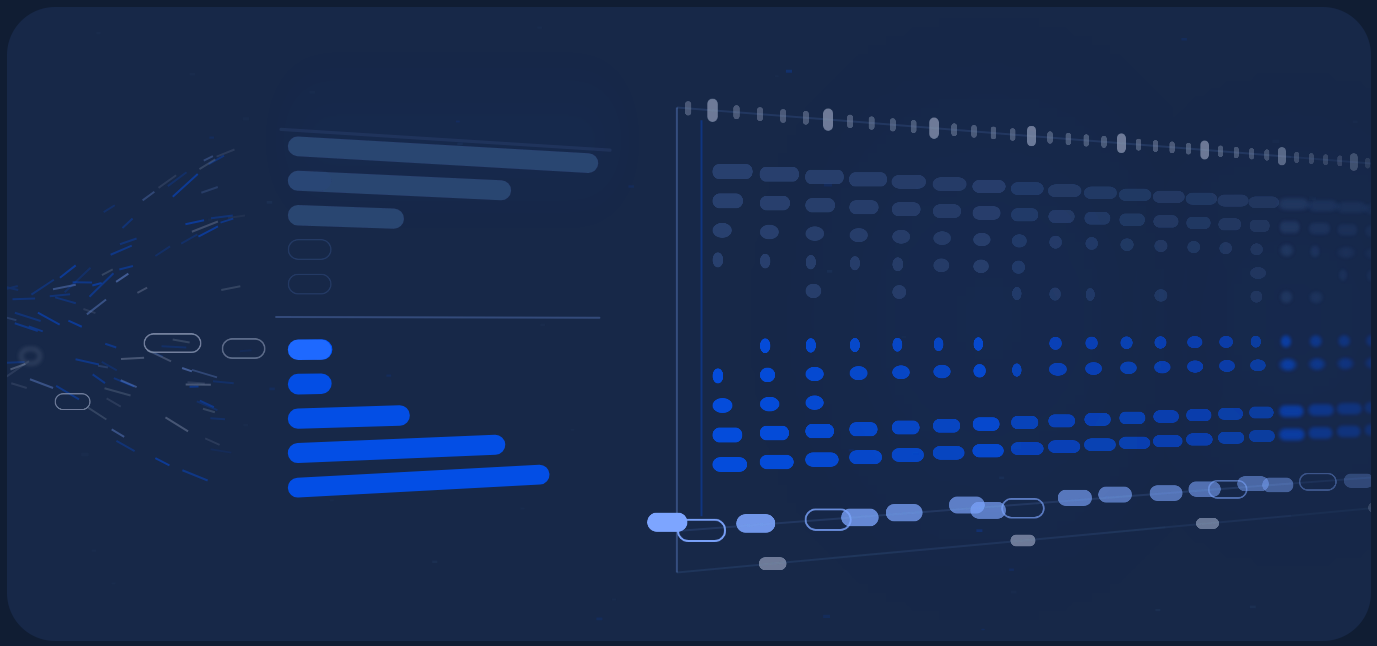
Rillor

METHODS NOTE

Rebuilding an order book from snapshots and diffs

Order book reconstruction applies sequenced diffs to a full snapshot under each venue's own numbering and checks, and one missed update leaves the book wrong until it is resynchronized.



Contents

01 Summary

02 Background

03 Method

04 Findings

05 Implications

06 Limits of this analysis

07 Sources

Summary

- Order book reconstruction applies diffs to a full snapshot in the order the venue published them. Each of the seven venues studied here sets its own rules for what a diff's size means, how the first diff is matched to the snapshot, how updates are numbered and what a client does after a gap [1, 16, 19, 27, 29].
- Coinbase, Binance and Lighter send the new total at a price level, and zero removes it [16, 19, 28]. Kraken removes a level on a quantity of 0 [22]. Nasdaq TotalView-ITCH sends executions and partial cancels as shares to subtract, and CME's market-by-price book addresses levels by position [3, 12].
- Gap tests differ by venue and by product. Binance Spot and Futures chain events by different fields, Lighter checks nonce ranges, and NYSE Daily TAQ and CME's per-product files carry numbers that skip by design [10, 15, 16, 17, 27].
- Among the feeds read, only Kraken documents a book checksum. This note recomputed its worked example and matched the documented value [22]. Depth limits leave levels unknown at Binance and require truncation at Kraken [16, 22].
- Recovery ranges from redundant feeds, snapshot loops and packet replay at CME to a fresh start at Binance and Lighter [1, 7, 8, 16, 28]. Archives have gaps too, and some list them [11, 30, 33].
- Finding 10 turns the rules into one venue-neutral procedure with ten tests.

Background

An order book lists the resting buy and sell interest in an instrument. A price-level book aggregates orders at each price: CME's market-by-price book "reports summarized order quantities and order counts at a given price level" [3]. An order-level book lists each order: CME's market-by-order feed "disseminates individual orders and quotes at every price level" [4], and Kraken's level3 channel "provides visibility of individual orders in the book" [24].

Most feeds send a full snapshot once and then only the changes. Lighter's channel "sends a complete snapshot on subscription, but only state changes after that" [27]. Binance publishes a diff stream "used to locally manage an order book" and a separate snapshot request [16]. The book at a given time is the snapshot plus every change after it, applied in order.

Order matters. Kraken notes that "Updates should always be processed in sequence" [23]. A missed change leaves the book wrong with no certainty that a later message repairs it. After a packet gap, CME says "all books are initially assumed to be in an incorrect state" [1]. A later update to the same level can overwrite a stale value, which CME calls natural refresh, and CME states that it "is not guaranteed and should not be considered a definitive substitute for recovering lost data" [6].

Transport adds failure modes. CME sends incremental data on two UDP feeds because a packet "may be missed from UDP Feed A but will be received from UDP Feed B" [1]. Nasdaq's MoldUDP64 lets listeners "detect and re-request missed packets" over UDP [13]. Coinbase warns that "Even though a WebSocket connection is over TCP", its servers can drop messages [20].

Two kinds of work depend on these rules: keeping a live copy of a book current, and replaying an archive to rebuild past books.

Method

What was examined. For each feed: what a diff's size means, how the snapshot is aligned with the diffs, which fields number updates, how updates are batched, what depth is covered, which state checks exist and how a client recovers. For archives: what each states about coverage and gaps.

Which sources. Venue documentation, read as published: eleven CME MDP 3.0 and DataMine pages on the CME Client Systems Wiki [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11], Nasdaq's TotalView-ITCH 5.0, MoldUDP64 and BX GLIMPSE 5.0 specifications [12, 13, 14], NYSE's Daily TAQ specification for its sequence numbers only [15], Binance's Spot and USD-M Futures order book pages and public data README [16, 17, 18], Coinbase Exchange's WebSocket and REST references [19, 20, 21], five Kraken references and guides [22, 23, 24, 25, 26], Lighter's WebSocket reference and market data guide [27, 28], and four Hyperliquid pages [29, 30, 31, 32]. One research dataset of Hyperliquid book diffs supplies a published coverage statement [33].

How. Every page was opened on 24 October 2026, and each source's version or update date is given where the page states one. CME's wiki pages were read through the wiki's public REST form, Nasdaq's specifications as PDF text and Binance's Spot rules from its GitHub documentation. Field names are quoted as published. Where two pages from one venue disagree, both are reported.

One computation was run. Kraken's checksum example was recomputed from the levels on its guide page with a standard CRC32 (Python's zlib) and gave 3310070434, the documented result [22]. No live feed was captured and no gap rates were measured. The procedure in finding 10 is this note's synthesis.

Findings

"Not stated" means the pages read do not say.

Feed	Size in an update	Sequence field	Depth	State check
CME market by price [2, 3, 6]	Add, Change or Delete at a level index	Packet per channel, RptSeq per instrument	Tag 264 per instrument	Not stated
CME market by order [2, 4]	New displayed quantity per order	Packet per channel	No maximum	Not stated
Nasdaq TotalView-ITCH [12, 13]	Shares to subtract, Replace gives a new total	MoldUDP64 message numbers	Displayable book	Not stated
Binance Spot [16]	New quantity	`U` and `u`	Snapshot up to 5,000 per side	Not stated
Binance USD-M Futures [17]	Absolute quantity	`pu` and `u`	Snapshot limit=1000	Not stated
Coinbase level2 [19]	Updated size	None in the examples	Entire book	Not stated
Coinbase full and level3 [19, 20]	Order messages	`sequence` per product	Entire book	Heartbeat
Kraken Spot book v2 [22, 23]	Quantity, 0 removes	None listed	10 to 1,000	CRC32, top 10
Kraken Spot level3 v2 [24, 25]	Order add, modify, delete	Pages disagree	10, 100 or 1,000	CRC32, top 10
Lighter [27, 28]	New total	`begin_nonce` and `nonce`	Not stated	Not stated
Hyperliquid l2Book [29]	Snapshot each push	None	20, or 5 with `fast`	Not stated
Hyperliquid node [31, 32]	New, update with old and new size, remove	Block number	Full book	Not stated

1. Absolute sizes and deltas

New totals at a price level. Coinbase's level2 size "is the updated size at the price level", and a size of "0" removes the level [19]. Binance Spot tells clients to "set the new quantity in the order book" and to remove a level whose quantity is zero [16], and its Futures page calls the data "the absolute quantity for a price level" [17]. Lighter's WebSocket reference leaves the point open, and its market data guide settles it: "A level's `size` is its new total, and `"0"` means remove it" [27, 28]. Kraken removes a level on a quantity of 0 [22].

Amounts to subtract. Nasdaq tracks each order by "a day-unique Order Reference Number" [12]. Subscribers "must deduct the number of shares stated in the Modify message from the original number of shares stated in the Add Order message", and repeated Modify and Order Executed messages "are cumulative" [12]. An Order Cancel states "The number of shares being removed from the display size of the order" [12]. The same feed sends a new total in one place: an Order Replace's Shares field is "The new total displayed quantity" [12].

New totals per order. CME's example raises order 555 from 30 to 50. The update carries tag 37706 MDDisplayQty with a value of 50, and "Order Priority changes from 722095 to 722787" because the order loses priority [4]. Hyperliquid's node writes a new order with its size, an update with both `origSz` and `newSz`, and a removal [32].

Levels by position. CME's market-by-price actions sit at a price level index in tag 1023 [3]. After an add, clients "shift price levels down, and delete any price levels past the defined depth" in tag 264. After a delete they shift lower levels up, and an add may fill the last level [3]. A client that keys levels by price alone misreads this feed.

Trades. At CME, "The trade data block itself is not used to update the order book" [3]. Nasdaq sends a Trade Message when a non-displayable order executes, and "Since Trade Messages do not affect the book", firms that only build a book may ignore them [12].

Why it matters. Applying a total as a delta corrupts the book at once: a level of 5 that receives a new total of 3 becomes 8. The convention also decides whether overlap is safe. Binance Futures keeps a first event that straddles the snapshot [17]. With absolute quantities, re-applying a change sets the same value. With deltas it would count twice (this note's reasoning).

2. Aligning the first diff with the snapshot

Where the snapshot comes from a separate request, every documented procedure buffers the diffs, takes the snapshot, drops what the snapshot contains and applies the rest.

Binance. Spot clients note "the `U` of the first event you received", fetch a snapshot with `limit=5000`, and retry "If the `lastUpdateId` from the snapshot is strictly less than the `U`" [16]. They "discard any event where `u` is <= `lastUpdateId`", and the first kept event should span `lastUpdateId` [16]. The Futures page uses

`limit=1000`, drops events where "u is < lastUpdateId", and expects the first event to have "U <= lastUpdateId AND u >= lastUpdateId" [17]. The boundary differs: Spot discards an event whose u equals `lastUpdateId`, and Futures keeps it.

Coinbase. The full channel procedure queues messages, requests the snapshot over REST and plays back the queue, "discarding sequence numbers before or equal to the snapshot sequence number" [19]. The REST book at level 3 returns the entire non-aggregated book with a `sequence` field [21].

CME. The Market Recovery feed loops full snapshots, and "the tag 369-LastMsgSeqNumProcessed value on the Snapshot message corresponds to the packet sequence number on the Incremental feed" [7]. Clients queue incremental data and drop "all cached Incremental feed updates with a packet sequence number < 369-LastMsgSeqNumProcessed" [7]. Where an instrument appears in both feeds, clients "compare the Market Recovery feed tag 60-TransactTime to the Incremental feed tag 60-TransactTime", and an instrument with an unequal value recovers in the next cycle or by natural refresh [7]. In CME's example the snapshot points at packet 104, and queued packets 100 to 103 are dropped [7]. Instruments with no book activity get no snapshot, clients process a full loop "starting at sequence number 1", and an order-level snapshot can span packets, counted by tags 37709 and 37710 [7].

Nasdaq GLIMPSE, as specified for BX. GLIMPSE sends the current book as Add Order messages, then an End of Snapshot message, type "G", with the ITCH sequence number at the time of the request. Firms "should begin to process real-time BX TotalView-ITCH 5.0 messages beginning with the sequence number stated" in it [14]. The TotalView-ITCH specification points to GLIMPSE for current book state [12].

On the venue's side. Lighter and Kraken send the snapshot on the same subscription as the diffs [23, 27], and Kraken's level 3 guide says the channel synchronizes snapshot and updates itself [25].

Each snapshot names its place in the update stream. Binance's `lastUpdateId` and Coinbase's `sequence` are the last update the snapshot includes [16, 19, 21]. CME drops cached packets numbered below tag 369 [7]. GLIMPSE's End of Snapshot sequence is the first ITCH message to process [14].

3. Sequence numbers, nonces and offsets

CME. Every packet carries a sequence number, and "each channel has its own separate set of sequence numbers that increment sequentially with each packet and are reset weekly" [2]. "MDP 3.0 uses the concept of sequencing and recovery by packet instead of by message" [2]. Inside, "tag 83-RptSeq represents the sequence number per instrument", so a client can tell which instrument lost data, and "If there is no gap, the data can be used immediately" [6]. That method applies to market-by-price recovery only [6].

Nasdaq. In MoldUDP64, the header's number is "The sequence number of the first message in the packet", and later messages "are implicitly numbered sequentially" [13]. A Message Count of zero "denotes a heartbeat". Heartbeats typically go once a second and "contain the next expected Sequence Number", so receivers "can sense packet loss even during times of low traffic" [13].

Binance. Spot events carry `U`, "First update ID in event", and `u`, "Final update ID in event" [16]. If `U` is greater than the book's update ID + 1, "you have missed some events" [16]. Futures chains events by another field: "each new event's `pu` should be equal to the previous event's `u`" [17]. One venue uses two tests for two products.

Coinbase. Sequence numbers rise by exactly one per message for each product. A larger jump means a dropped message, and a lower one "can be ignored or represent a message that has arrived out of order" [20]. Heartbeats every second carry "sequence numbers and last trade IDs that can be used to verify that no messages were missed" [19]. The level2 channel "guarantees delivery of all updates", and its examples carry no `sequence` field [19].

Kraken. The Spot book v2 update lists no sequence field [23]. For level 3 the pages disagree: the reference says "no sequencing is required" [24], and the guide lists "Sequence numbers for synchronization" [25]. This note leaves the question open. Kraken's Futures book carries `seq`, "The subscription message sequence number" [26].

Lighter. "While the nonce is tied to Lighter's matching engine, the offset is tied to the API servers", so the offset can change sharply on reconnection and is "not guaranteed to be continuous" [27]. Continuity holds when "begin_nonce on the current update matches the nonce (i.e. last_nonce) of the previous update" [27]. In the published example one update runs from 9182389998 to 9182390020, a span of 22 [27]. The guide's rule: "Use `nonce` for continuity, not `offset`" [28].

Hyperliquid. `WsBook` carries `coin`, `levels` and `time`, with no sequence field [29]. The node's batched output carries `block_number` [31].

Numbers that skip by design. Daily TAQ holds trades and quotes without book depth, and shows the trap. Tape C numbers increment "by 1 with every message" [15]. Tape A and B numbers "do not increase continuously" because they equal "(CTA Block Sequence Number * 100) + CTA Message ID", as in the examples 103, 2,439 and 456,705 [15]. CME's per-product market-by-order files lack some packet numbers because "they were assigned to another product at the PCAP level", while "all instrument level sequence numbers should be in each file" [10].

Resets. In a CME channel reset, "Tag 83-RptSeq for Market by Price (MBP) book updates resets to 1" [9]. A gap test must read the drop to 1 as a reset.

4. Batched and snapshot-only feeds

Events across packets. At CME, "a single message packet can contain multiple updates for multiple instruments" [1], and one event can span several packets [2]. "All book updates in the latest Market Data Incremental Refresh message must be processed before the order book can be considered valid" [1], and the same holds for order-level updates within an event [4]. Implied updates come last, sometimes in a later packet that carries the End of Event indicator [5].

Several updates in one message. Kraken can send "multiple updates to the same price level in a single update message" [23], and its checksum guide says to process all of them before computing the checksum [22].

Timed batches. Coinbase's level2_batch channel sends "every 50 milliseconds", with the time field of the most recent message [19]. Lighter batches "every 50ms" [27]. Binance's Spot diff stream updates every 1,000 ms or 100 ms [16]. Kraken's level3 channel "is not throttled" [24].

Snapshot-only feeds. Hyperliquid's l2Book is a "Snapshot feed, pushed on each block that is at least 0.5 since last push" [29]. The source gives no unit. Levels carry `px`, `sz` and `n`, 20 levels, or 5 with `fast` [29]. Nothing needs sequence checks, and a history built from it holds only the pushed states at the pushed depth.

5. Depth limits and levels outside the snapshot

Binance. With snapshots capped at 5,000 levels per side, "you won't learn the quantities for the levels outside of the initial snapshot unless they change" [16]. On Futures, "Receiving an event that removes a price level that is not in your local order book can happen and is normal" [17]. A client that fails on an unknown removal resyncs without cause.

Kraken. Book depths are 10, 25, 100, 500 or 1,000, default 10 [23]. The client truncates to the subscribed depth after each update because Kraken sends no zero-quantity update for levels that fall out of scope [22]. Level 3 applies the same rule: with a depth of 10, an insert that makes 11 bids means "you must remove the 11th worst bid" [24]. A level that comes into scope arrives as adds for all its orders [24].

CME. Market-by-price depth comes from tag 264 per instrument [3]. Market by order has "no maximum number of orders or depth" [4]. CME's example cancels an order "outside the maximum market depth of 10 for the MBP book", so only the order-level feed changes [4]. Implied prices form a separate "2-deep best bid and ask" [5] and are "not sent in MBOFD format" [4], so an order-level rebuild holds no implied prices.

Others. Coinbase's level2 snapshot represents "the entire order book" [19]. Hyperliquid's l2Book stops at 20 levels [29].

A level outside the published depth is unknown, which differs from empty. A rebuilt book should store its depth with each state.

6. Checksums and other state checks

Kraken. "The book channel includes a CRC32 checksum with each update", and it "is always calculated over the top 10 price levels regardless of subscription depth" [22]. Clients take the top 10 asks low to high and the top 10 bids high to low, remove the decimal point and leading zeros from each price and quantity, concatenate asks then bids, and compare the unsigned CRC32 with the message's field [22]. Kraken advises "a decimal or string decoder to preserve full precision" [22]. Its worked example expects 3310070434, and this note's recomputation matched. The level 3 checksum "also verifies the sequence of orders in a price

level" [25]. Some clients generate it only in their development environment, while building their book models [25]. The Kraken pages read document no action after a mismatch, and the procedure below treats one as a gap. No other page read documents a book checksum.

Other checks. Kraken states its level 3 book "will never be crossed" [25]. This note applies that test to every rebuilt book in continuous trading. Coinbase's heartbeat gives the last trade ID as an outside reference [19]. Hyperliquid's `origSz` checks a held order's size [32]. A fresh, aligned snapshot gives a full comparison.

7. Recovery when a gap appears

CME. CME "strongly recommends that client systems process both the A and B Incremental UDP feeds" [1]. After loss on both, clients "must resynchronize all books" [1]. Market Recovery serves "large-scale data recovery (e.g. major outage or late join)", and a recovered instrument can resume while others recover [7]. TCP replay serves small gaps: "A maximum of 2000 packets can be requested at once", with "a 24 hour time limit on the messages available", and recovered packets are applied before queued ones [8]. RptSeq limits the damage: if lost packets held data for 2 of 10 instruments, messages for the others "are still valid" [6]. Natural refresh starts from an emptied book and "should only be used in conjunction with Market Recovery" [6]. In a channel reset, marked by tag 269 MDEntryType=J, "The order book and all statistics for the instrument should be emptied" [9].

Nasdaq. A MoldUDP64 receiver that sees an unexpected number sends a Request Packet and waits [13]. Responses carry only the messages that fit in one packet, so large gaps need several requests [13]. GLIMPSE supplies a fresh book [12, 14].

Binance and Lighter. Binance says "Discard your local order book and restart the process from the beginning" [16]. A Spot connection "is only valid for 24 hours" [16]. Lighter says "Always reconnect and resubscribe automatically", and its sample client resubscribes on a nonce mismatch [28].

Coinbase, Kraken and Hyperliquid. Coinbase asks clients to handle sequence gaps or use channels that guarantee delivery, such as level2 [20]. The Kraken pages read give no recovery step. Hyperliquid's l2Book needs none [29].

8. Order-level books: adds, executions and replaces

Priority. CME orders at one price rank by tag 37707, where "a lower value is a higher priority", and "systems must first sort by price (tag 270-MDEntryPx), then by priority" [4].

Identity. A CME order ID "should stay the same for each order until it has been cancelled or filled" [10]. A Nasdaq Order Replace removes the original order's shares and issues "a new order reference number which will be used henceforth" [12]. Side, symbol and attribution are absent from the replace and must be kept from the original Add Order [12]. Stock locate codes hold within a day, with "no expectation that the assignment will be the same across multiple days" [12]. Timestamps are "nanoseconds since midnight" [12].

Messages that do not touch the book. In Coinbase's full channel, "All open and match messages always result in a change to the order book. Not all done or change messages result in changing the order book" [19]. Applying messages for orders never on the book leaves it "incorrect" [19]. Kraken's level 3 book excludes in-flight orders, unmatched market orders, untriggered stops and hidden iceberg quantity [25]. Both Kraken level 3 feeds carry `OrderID`, so fills can be matched to book entries, which "is not possible with L2 data" [25].

On-chain books. Hyperliquid's raw book diffs carry `user`, `oid`, `coin`, `side` and `px` [32]. A node can compute "L4 book snapshots (full onchain order information)" from a state snapshot [31], and writes "around 100 GB of logs per day" at default settings [32].

9. Historical archives and their gaps

CME DataMine. The market-by-order FIX files "provide all market data messages required to recreate the book" from 2017 [10]. The page states their timestamp precision two ways, so none is given here. The packet capture dataset holds "the complete historical raw data feeds on all channels" [11], and still lists anomalies: a gap on channel 381 from 27 June to 12 July 2021, and "minor packet loss gaps for select channels" on 6 April to 4 May 2017, 24 March 2018, 3 May 2019 and 24 to 27 December 2019 [11]. Each channel day comes as two files, split at midnight Central Time [11].

Binance. The public data README describes `aggTrades`, `klines` and `trades` files and no order book files [18]. It warns that "Archived files may be updated at a later date as a result of recently discovered issues" [18], so a file read twice can differ.

Hyperliquid. Its archive is updated "approximately once a month", and "There is no guarantee of timely updates and data may be missing" [30]. It holds L2 book snapshots, and the requester pays transfer costs [30]. A node writes state snapshots "every 10,000 blocks" and, with `--write-raw-book-diffs`, "every L1 order diff" [31]. That gives an order-level snapshot-and-diff path ordered by block.

A dataset that states its coverage. The Albers et al. dataset on Zenodo publishes Hyperliquid book diffs for BTC, ETH and SOL perpetuals for December 2025: "24 hourly files per day (744 total), no gaps" [33]. Its trade data "has some gaps from collection infrastructure restarts", listed by month: 2 missing hours in October 2025, 10 in November, none in December and 1 in January 2026 [33]. A statement like this shows which windows a rebuilt book can be trusted for.

10. A venue-neutral procedure and its tests

Five venue slots fill one loop: the size rule, the alignment boundary, the continuity test, the depth and the state checks.

```
open the diff stream and buffer every diff
snapshot = request a full snapshot
drop buffered diffs the snapshot already contains (the venue's boundary rule)
check that the first kept diff meets the venue's alignment rule
book = snapshot, last = none

for each kept diff, buffered first and then live:
    if last is set and diff ends at or before last: skip it
    if last is set and not continuous(last, diff): record the gap and resync
    apply each change with the venue's size rule
    last = diff.last_update
    at the end of each message, event or batch:
        cut the book to its depth
        run the checksum and crossed-book checks
        on failure: record it and resync
```

Venue	Drop before applying	continuous(last, diff)
CME MDP 3.0	Packets below tag 369. At the equal packet, compare tag 60 TransactTime [7]	packet number = last + 1, RptSeq + 1 per instrument, reset to 1 handled [2, 6, 9]
Nasdaq ITCH over MoldUDP64	Messages before the GLIMPSE End of Snapshot sequence number [14]	first message number = next expected [13]
Binance Spot	u <= `lastUpdateId` [16]	U <= last + 1 [16]
Binance USD-M Futures	u < `lastUpdateId` [17]	pu = previous u [17]
Coinbase full or level3	`sequence` <= snapshot `sequence` [19]	`sequence` = last + 1 [20]
Lighter	None, snapshot on subscription [27]	`begin_nonce` = previous `nonce` [27, 28]
Kraken Spot book v2	None, snapshot on subscription [23]	checksum only [22, 23]

A rebuilt book passes when:

1. **Continuity.** Every update passes the venue's test above.

2. **Known skips.** Daily TAQ Tape A and B numbers follow the CTA formula, CME per-product files are checked on tag 83, and channel resets are recorded as resets [9, 10, 15].
3. **Alignment.** The first applied diff meets the venue's boundary rule [7, 14, 16, 17, 19].
4. **Whole events.** No check runs before a message, event or batch ends [1, 4, 22].
5. **Checksum.** It matches wherever the venue sends one [22, 24].
6. **No crossing.** The best bid stays below the best ask in continuous trading [25].
7. **Depth.** The book is cut to its depth, and levels beyond it are marked unknown [3, 16, 22].
8. **Order identity.** Every order update refers to a live order, a replace retires the old reference, and messages for orders never on the book are ignored [12, 19].
9. **Snapshot agreement.** A fresh, aligned snapshot equals the rebuilt book at the same point (this note's test).
0. **Coverage record.** Every gap, resync and reset is written down with its instrument and time range, as CME's capture dataset and the Albers et al. dataset do [11, 33].

Implications

For engineers keeping a live book. Treat each product's rules as a separate contract. Binance Spot and Futures differ in continuity field and alignment boundary [16, 17], and Kraken's spot and futures feeds differ on sequencing [23, 26]. Test the field the venue names, and rerun alignment after every reconnect where the venue's steps start from a new connection [16, 28].

For researchers replaying history. A rebuilt book is only as complete as its archive. Check gaps on the right field, such as tag 83 in CME's per-product files and the CTA formula in Daily TAQ [10, 15]. Record each source's depth and cadence, since a snapshot-only feed holds only its pushed states [29]. Expect archived files to change where the provider says so [18].

For buyers of order book data. Ask how a dataset fills the five venue slots, and ask for its gap list by instrument and window, in the form CME's capture dataset and the Albers et al. dataset publish theirs [11, 33]. Rillor's market data holds on-chain perpetual futures order books rebuilt from snapshots and book diffs [34, 35]. Book sequence numbers are checked on every rebuild, and gaps are written into each instrument's coverage notes [34]. Its published update record uses absolute sizes: "Size resting at the level after the update", with zero removing the level [34].

Limits of this analysis

- Venue documentation changes without notice. Every rule here is as read on 24 October 2026.
- This note reads documentation only. It measured no gap rates, resync frequencies or latencies.
- Kraken's level 3 pages disagree on sequencing, and CME's market-by-order FIX page gives two timestamp precisions. Neither point is treated as settled.
- Nasdaq's GLIMPSE mechanism comes from the specification for its BX market, because the Nasdaq-market specification could not be opened. Binance's Futures page was read through a fetch tool's rendering.
- Venues document more products than those read here, and rules for one product may not hold for another.
- The crossed-book and snapshot-agreement tests are this note's own, and auctions can follow different rules.
- This is a methods note about data handling, with no investment, legal or trading advice.

Sources

1. MDP 3.0 - Recovery Services. CME Group Client Systems Wiki, page version 2, 26 December 2024. https://cmegroupclientsite.atlassian.net/wiki/spaces/EPICSANDBOX/pages/457325847/MDP+3.0+-+Recovery+Services

2. MDP 3.0 - Packet Structure with Event Based Messaging. CME Group Client Systems Wiki, page version 4, 13 May 2026. https://cmegroupclientsite.atlassian.net/wiki/spaces/EPICSANDBOX/pages/457326496

3. MDP 3.0 - Market by Price - Multiple Depth Book. CME Group Client Systems Wiki, page version 2, 26 December 2024. https://cmegroupclientsite.atlassian.net/wiki/spaces/EPICSANDBOX/pages/457574047

4. MDP 3.0 - Market by Order - Book Management. CME Group Client Systems Wiki, page version 2, 23 December 2024. https://cmegroupclientsite.atlassian.net/wiki/spaces/EPICSANDBOX/pages/457605721/MDP+3.0+-+Market+by+Order+-+Book+Management

5. MDP 3.0 - Implied Book. CME Group Client Systems Wiki, page version 3, 10 January 2025. https://cmegroupclientsite.atlassian.net/wiki/spaces/EPICSANDBOX/pages/457422235

6. MDP 3.0 - Market by Price - Book Recovery Methods for Concurrent Processing. CME Group Client Systems Wiki, page version 3, 10 January 2025. https://cmegroupclientsite.atlassian.net/wiki/spaces/EPICSANDBOX/pages/457705188

7. MDP 3.0 - MBP and MBOFD Market Recovery. CME Group Client Systems Wiki, page version 4, 1 October 2025. https://cmegroupclientsite.atlassian.net/wiki/spaces/EPICSANDBOX/pages/457672425

8. MDP 3.0 - TCP Recovery. CME Group Client Systems Wiki, page version 4, 10 January 2025. https://cmegroupclientsite.atlassian.net/wiki/spaces/EPICSANDBOX/pages/457574209

9. MDP 3.0 - Channel Reset. CME Group Client Systems Wiki, page version 5, 9 June 2026. https://cmegroupclientsite.atlassian.net/wiki/spaces/EPICSANDBOX/pages/457326922

10. MBO FIX (CME DataMine dataset). CME Group Client Systems Wiki, page version 7, 20 October 2026. https://cmegroupclientsite.atlassian.net/wiki/spaces/EPICSANDBOX/pages/457223111/MBO+FIX

11. Packet Capture Dataset (CME DataMine). CME Group Client Systems Wiki, page version 8, 20 October 2026. https://cmegroupclientsite.atlassian.net/wiki/spaces/EPICSANDBOX/pages/457323061/Packet+Capture+Dataset

12. Nasdaq TotalView-ITCH 5.0 Specification. Nasdaq, latest revision 28 April 2023. https://www.nasdaqtrader.com/content/technicalsupport/specifications/dataproducts/NQTVITCHSpecification.pdf

13. MoldUDP64 Protocol Specification, Version 1.00. Nasdaq, last revised 2 August 2024. https://www.nasdaqtrader.com/content/technicalsupport/specifications/dataproducts/moldudp64.pdf

14. BX GLIMPSE 5.0 Interface Specification, Version 5.00. Nasdaq, 1 October 2013. https://www.nasdaqtrader.com/content/technicalsupport/specifications/dataproducts/BXGLIMPSESpecification_5.0.pdf

15. Daily TAQ Client Specifications, Version 4.0. NYSE, 2 November 2022.

https://www.nyse.com/publicdocs/nyse/data/Daily_TAQ_Client_Spec_v4.0.pdf

16. WebSocket Streams for Binance (web-socket-streams.md). Binance, binance-spot-api-docs repository on GitHub, last changed 15 October 2026. https://github.com/binance/binance-spot-api-docs/blob/master/web-socket-streams.md

17. How to manage a local order book correctly (USDⓈ-M Futures). Binance Developer Docs, viewed 24 October 2026. https://developers.binance.com/docs/derivatives/usds-margined-futures/websocket-market-streams/How-to-manage-a-local-order-book-correctly

18. Binance Public Data (README). Binance, binance-public-data repository on GitHub, last changed 30 September 2026. https://github.com/binance/binance-public-data/blob/master/README.md

19. Exchange WebSocket Channels. Coinbase Developer Documentation, viewed 24 October 2026. https://docs.cdp.coinbase.com/exchange/websocket-feed/channels

20. Exchange WebSocket Overview. Coinbase Developer Documentation, viewed 24 October 2026. https://docs.cdp.coinbase.com/exchange/websocket-feed/overview

21. Get product book. Coinbase Developer Documentation, Exchange REST API, viewed 24 October 2026. https://docs.cdp.coinbase.com/api-reference/exchange-api/rest-api/products/get-product-book

22. Book checksum (WebSocket v2). Kraken Developers, viewed 24 October 2026. https://docs.kraken.com/api/docs/guides/spot-ws-book-v2

23. Book (Level 2), WebSocket v2. Kraken Developers, viewed 24 October 2026. https://docs.kraken.com/api/docs/websocket-v2/book

24. Orders (Level 3), WebSocket v2. Kraken Developers, viewed 24 October 2026. https://docs.kraken.com/api/docs/websocket-v2/level3

25. Spot Level 3 Market Data. Kraken Developers, viewed 24 October 2026. https://docs.kraken.com/api/docs/guides/spot-l3-data

26. Book (Futures WebSocket v1). Kraken Developers, viewed 24 October 2026. https://docs.kraken.com/api/docs/futures-api/websocket/book

27. WebSocket. Lighter API documentation, updated 18 October 2026. https://apidocs.lighter.xyz/docs/websocket-reference

28. Market data. Lighter API documentation, updated 18 October 2026. https://apidocs.lighter.xyz/docs/market-data

29. WebSocket: Subscriptions. Hyperliquid Docs, viewed 24 October 2026. https://hyperliquid.gitbook.io/hyperliquid-docs/for-developers/api/websocket/subscriptions

30. Historical data. Hyperliquid Docs, viewed 24 October 2026. https://hyperliquid.gitbook.io/hyperliquid-docs/historical-data

31. hyperliquid-dex/node README. Hyperliquid, GitHub, last changed 10 August 2026. https://github.com/hyperliquid-dex/node/blob/main/README.md

32. L1 data schemas. Hyperliquid Docs, viewed 24 October 2026. https://hyperliquid.gitbook.io/hyperliquid-docs/for-developers/nodes/l1-data-schemas

33. An Open Book: Level 4 Order Book Data from the Hyperliquid Exchange, version 1.0 (dataset), by Jakob Albers, Mihai Cucuringu, Sam Howison and Alexander Y. Shestopaloff. Zenodo, 25 March 2026. https://zenodo.org/records/18184441

34. Market data. Rillor, viewed 7 November 2026. https://rillor.com/markets/market-data

35. Data services. Rillor, viewed 7 November 2026. https://rillor.com/data

About this report

Published by Rillor on 7 November 2026. Superintelligence (SI) tools assist research and drafting, and every figure cites its source. Rillor builds and secures SI systems, supplies the data they learn from and delivers the infrastructure they run on. We apply SI to financial and compute markets.

Cite as: Rillor. Rebuilding an order book from snapshots and diffs. 7 November 2026. https://rillor.com/insights/rebuilding-order-books-from-snapshots-and-diffs

Notices

Rillor is not a registered investment adviser or commodity trading advisor and does not provide investment or trading advice. Research, forecasts, data and software described on this site are for research and engineering use. Nothing here is an offer or recommendation to buy or sell any security, commodity interest or digital asset. Past or simulated results do not indicate future results.

Work with Rillor

Send the instruments, the history window, the delivery format and the intended use. Contact us at rillor.com/contact?type=market-data or contact@rillor.com. More on market data: rillor.com/markets/market-data