Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Distributed Computing

Polymarket's Hybrid Exchange System and Prediction Market Arbitrage

Bachelor's Thesis

Mario Cattaneo

mcattane@ethz.ch

Distributed Computing Group
Computer Engineering and Networks Laboratory
ETH Zürich

Supervisors:
Prof. Dr. Lucianna Kiffer, Oriol Saguillo
Prof. Dr. Roger Wattenhofer

February 12, 2026

Acknowledgements

I want to thank Prof. Dr. Lucianna Kiffer and Oriol Saguillo from IMDEA Networks for giving me the opportunity to do my thesis under their supervision and taking the time to meet me almost every week and guiding me throughout the process, as well as providing me access to their server for running my data collection.

Abstract

Polymarket and Kalshi are exchanges making prediction markets accessible globally, with significant growth in trading volume and popularity. Both use a Central Limit Orderbook as a market mechanism, but Polymarket implements event contracts, settlement, resolution, and more through Ethereum-based smart contracts. The operation of this hybrid system remains largely undocumented.

The purpose of this thesis is to understand the nature of prediction markets and arbitrage opportunities as they exist on Polymarket and Kalshi, as well as break down Polymarket's hybrid exchange system to fill in the missing gaps and compare it to other exchanges. To achieve this, we design data collection clients for the endpoints of Polymarket, Kalshi, and Infura's eth_getLogs service, addressing poorly documented APIs and discovered inconsistencies. Using on-chain data from November 18, 2025 to January 5, 2026 and historical off-chain market objects, we examine the market lifecycle, identifying three distinct reporting oracles (MOOV2 Adapter, Negrisk Adapter, and Centralized Adapter), market creation, registration, and resolution patterns, and the outcome distribution (41.17% YES, 58.58% NO, 0.25% UNKNOWN).

Analyzing over 43 million settlements, we find that slippage y and taker USDC amount x follow the relationship $y = \beta x^{\alpha}$, with $\beta = 2.751 \times 10^{-2} \pm 9.675 \times 10^{-3}$ and $\alpha = 0.6778 \pm 2.487 \times 10^{-2}$, determined through linear regression on log-transformed data with $R^2 = 0.3360$. This deviates from the typical square-root law of market impact, suggesting that slippage increases superlinearly with order size on Polymarket. However, it still compares favorably to the linear scaling slippage in decentralized AMMs like Uniswap V2. Lastly, we use the data collected through the endpoint clients from January 8 to February 2, 2026 to analyze price discrepancies across 1000 cycles of the 15-minute "Bitcoin up or down?" markets, comparing cross-market and cross-exchange discrepancies relative to the market lifecycle timeline. We find that the median discrepancy is highest at the start of the cycle and remains relatively low towards the end, with an increasingly skewed distribution.

Contents

Acknowledgements i

Abstract ii

1 Introduction 1

1.1 Prediction Markets . . . 1
1.2 Growth . . . 2
1.3 US Market . . . 3
1.4 Accessibility . . . 4

2 Objective and Methodology 5

2.1 Methodology . . . 5

3 Endpoints and Clients 7

3.1 Polymarket Event and Market Endpoints . . . 7
3.1.1 Event and Market Objects . . . 7
3.2 Polymarket Orderbook and Trade Endpoints . . . 8
3.2.1 Book Endpoint . . . 8
3.2.2 Falsely Missing Orderbook State Objects . . . 9
3.2.3 Market Feed . . . 9
3.2.4 Ambiguous Event Ordering . . . 10
3.3 Final Polymarket Client . . . 10
3.4 Kalshi Endpoints . . . 11
3.5 Kalshi Client . . . 12

4 Smart Contracts 13

4.1 Conditional Tokens . . . 13
4.2 UMA Framework . . . 14
4.3 UmaCtfAdapter . . . 15

4.4 CTFExchange . . . 15
4.5 Negrisk . . . 16
4.6 Polygon Collection . . . 16

5 Market Lifecycle 18
5.1 Market Creation . . . 18
5.1.1 Reporting Oracles . . . 18
5.1.2 Centralized Adapter . . . 19
5.1.3 Resolution Source . . . 20
5.2 Exchange Registration . . . 21
5.2.1 Unregistered Traded Outcomes . . . 22
5.3 Resolution . . . 22
5.3.1 Disputes . . . 22
5.3.2 Reported Unknown Outcome . . . 23
5.3.3 Manual Resolutions . . . 23

6 Orderbook and Trading 24
6.1 State, Ordering, and Matching . . . 24
6.2 Slippage . . . 25
6.2.1 Power Relation between USDC Taker Amount and Slippage 25
6.2.2 Comparison with Decentralized AMM Slippage . . . 27
6.3 Arbitrage . . . 27

7 Discussion 29
7.1 Data Collection Barriers . . . 29
7.2 Market Integrity and Effectiveness . . . 29
7.3 Slippage and Market Design . . . 29
7.4 Unregistered Markets and Emerging Infrastructure . . . 30

Bibliography 31

A Endpoint Documented JSON responses A-1

B Smart Contracts B-1

C Polymarket Event Object JSON Tree Analysis **C-1**

D Miscellaneous **D-1**

D.1 Slippage Computation Algorithm D-2

Chapter 1

Introduction

In this chapter we introduce prediction markets as they exist on Polymarket and Kalshi, important relations for later analysis and also general information about Polymarket and Kalshi as motivation and context for the rest of the thesis.

1.1 Prediction Markets

A prediction market is derived from the outcome of a real-world event. On both exchanges, a prediction market is exclusively designed for binary events with YES and NO outcomes. These outcomes are represented as shares quoted in USD, or on Polymarket specifically, in USDC, a USD-backed ERC-20 token managed by Coinbase through a reserve ensuring its value is identical to USD. An example is the event "Will the Republican Party win the House in 2026?", which is hosted on both exchanges. A position in a prediction market refers to a specific quantity of YES or NO shares held. The process of deciding the outcome of the underlying event and assigning terminal values for a YES and NO share is called resolution; once the values are determined, the market is deemed resolved.

On Kalshi, every YES or NO share is a contract with an expiration date and a central resolution source. Kalshi states, "Each contract is worth $1 if you're right" [1]. Their legally binding rulebook states that the final outcome is decided at their discretion and that contract modifications are possible [2]. However, for the purpose of this thesis, we assume that markets can only resolve to either YES or NO.

Polymarket uses the Gnosis Conditional Tokens contract (CTF) to define the YES and NO shares as ERC-1155 tokens and to enforce their combined value at 1 USDC. Additionally, it uses UMA's optimistic oracle framework for resolution 4.2. In this system, a prediction can resolve to UNKNOWN, rendering the value of YES and NO shares 0.5 USDC each 4.3.

In both cases, the combined value of a YES and NO share for a prediction is exactly 1 USD at all times, as enforced by the trading contract on Kalshi and by the CTF smart contract on Polymarket 4.1. This does not hold for the market

prices of the YES and NO shares on the exchange, which are determined by a Central Limit Order Book (CLOB) on both exchanges 6. This discrepancy is what makes internal market arbitrage possible.

Events with more than two discrete outcomes can be constructed using multiple prediction markets. For example, the event "Which party will win the House in 2026?" is decomposed into prediction markets for the questions "Will the Democratic Party win the House in 2026?" and "Will the Republican Party win the House in 2026?". This provides powerful information aggregation capabilities built on top of the information efficiency of markets. A common pattern on both exchanges is to define an event with more than two possible discrete outcomes exhaustively, using a separate prediction market for each mutually exclusive outcome. The event "Which party will win the House in 2026?" is an example of this. Polymarket labels these as *Negrisk* and provides a full suite of smart contracts dedicated to these events, including a distinct settlement smart contract deployment *Negrisk CTFE* and an atomic swap operation for converting a set of mutually exclusive outcome positions into the complementary positions of the alternative outcomes 4.5.

1.2 Growth

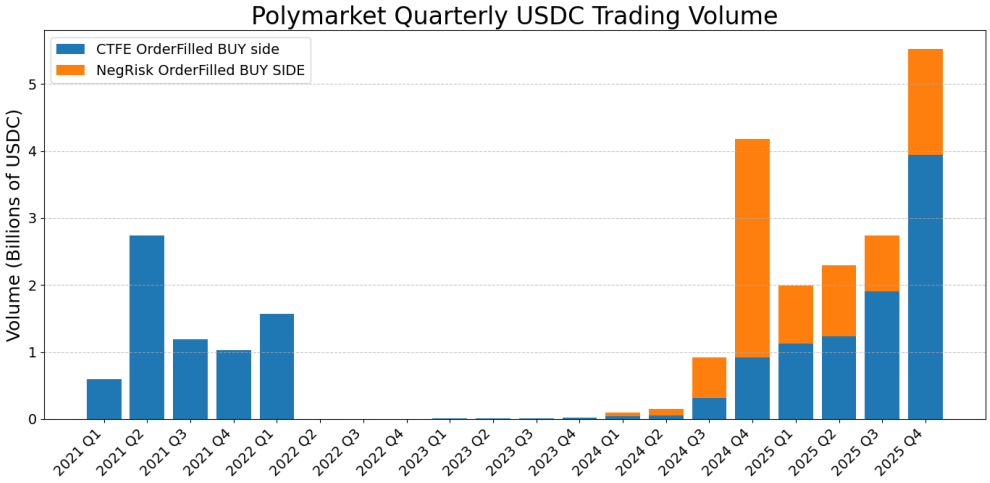

Figure 1.1: Quarterly trade volume on Polymarket. Source: Author's own compilation using Dune Analytics (https://dune.com/queries/6298071).

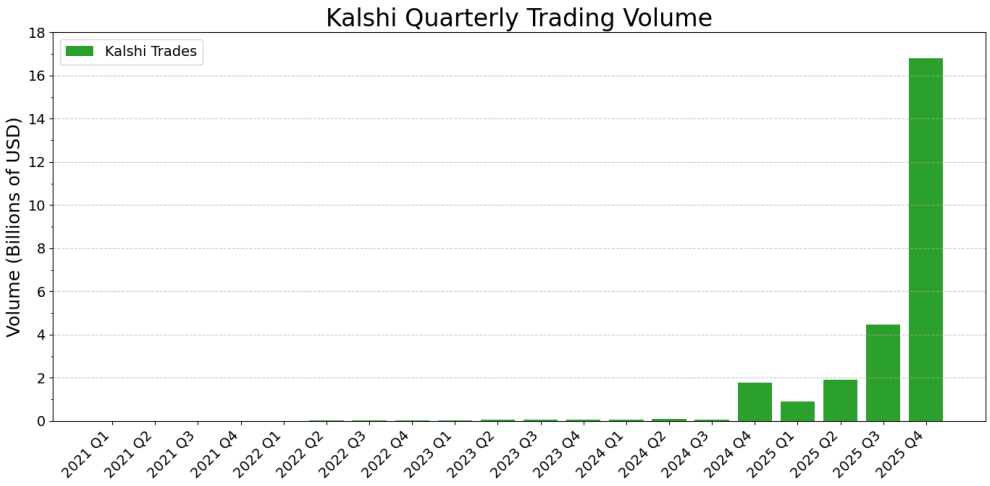

Figure 1.2: Quarterly trade volume on Kalshi. (https://tokenterminal.com/explorer/projects/kalshi?interval=max&granularity=quarter).

Figures 1.1 and 1.2 show the quarterly USD trading volume. For Polymarket, we can also distinguish between trading volume on the Negrisk CTFE and the OOV2 CTFE 4.4.

1.3 US Market

The flat line in Polymarket's trading volume starts right after the CFTC action in January 2022. Kalshi operated only in the U.S. until October 2025, covering the first three quarters of 2025 and the last quarter of 2024 shown in Figure 1.2.

The 47th U.S. presidential election was a major event for both exchanges. Various studies have compared the forecasting ability and information efficiency of Kalshi and Polymarket for this event against other methods like New York Times polls [3, 4]. The volume spike in the fourth quarter of 2024 is also clearly visible. On Kalshi, markets related to politics and the election accounted for roughly 1 billion USD in volume in November alone [5]. On Polymarket, the main market for the presidential election accumulated a total volume of 3.5 billion USDC [6].

However, a recent study using two unsupervised network models suggests that around 18% or less of the volume traded on Polymarket is wash trading, where users trade with themselves to inflate activity. One method reported that approximately 60% of the December 2024 volume was washed [7]. This occurred the month after the U.S. election markets concluded and coincided with a decline of over 50% in total trading volume [8].

1.4 Accessibility

Kalshi expanded to over 140 countries in October 2025 [9], and Polymarket plans to reenter the U.S. market. On July 21, 2025, Polymarket acquired QCX LLC, a company that had registered as a Designated Contract Market (DCM) with the CFTC earlier that month [10]. This entity was subsequently rebranded as "Polymarket US" [11]. As of December 4, 2025, Polymarket was not yet available in the U.S. Both exchanges face regulatory headwinds in many jurisdictions. Gambling authorities in several countries have banned them. This includes Switzerland, where the Swiss Gambling Supervisory Authority (GESPA) has blacklisted their DNS domains, blocking access via Swiss ISPs [12].

Despite CFTC approval, Kalshi faces state-level regulatory issues in the U.S. For example, the Illinois Department of Financial and Professional Regulation (IDFPR) issued a cease-and-desist order preventing Kalshi from operating in Illinois for allegedly defying state gambling regulations. Kalshi argues that national oversight by the CFTC preempts these claims. This ongoing case demonstrates that the regulatory landscape surrounding speculation remains conflicted [13].

Polymarket remains accessible by design, even where banned, because it does not require Know-Your-Customer (KYC) identification on its main platform. Even if KYC were enforced, users could still interact with the underlying smart contracts directly. The ERC-1155 tokens representing YES or NO shares are defined by the Gnosis Conditional Tokens contract. These can be obtained via *splitPosition*, swapped for USDC via *mergePosition*, and redeemed after resolution using *redeemPosition* for the value reported by Polymarket.

Chapter 2

Objective and Methodology

The first objective was to design a data collection client for Polymarket's off-chain endpoints. The second objective was to analyze the market lifecycle on Polymarket. The third objective was to analyze slippage on Polymarket and prediction market arbitrage.

2.1 Methodology

Both Polymarket and Kalshi provide WebSocket endpoints to subscribe to orderbook state changes and trades, as well as HTTP endpoints to request the orderbook state. To use these endpoints, the orderbooks of interest must first be identified. For this, both exchanges define market objects and provide HTTP endpoints to query historical data, as well as WebSocket endpoints to subscribe to market lifecycle events. Both exchanges use JSON for endpoint communication and define rate limits for request throughput.

Due to Polymarket's sparse documentation, significant information had to be discovered through testing and response analysis. These findings, along with continuous endpoint changes and discovered state inconsistencies, resulted in multiple client iterations. The final clients are tailored for the events "Which party will win the House in 2026?" and the cyclical 15-minute event "Bitcoin up or down?". This is covered in Chapter 3.

To analyze the market lifecycle, Chapter 4 first examines the statically available information from on-chain smart contracts. This includes essential smart contract control flow, emitted events, and immutable contract storage. These elements allow for establishing links between contracts and their intended use, and they help identify details that can only be deduced from emitted events. For this purpose, a client for the Infura *eth_getLogs* endpoint was implemented. These events and the historical market objects, retrieved through Polymarket's endpoints, are then used to analyze market lifecycle coordination in Chapter 5.

In Chapter 6, the orderbook for prediction markets is formalized to define how

slippage and arbitrage are measured. Then, data collected from Polymarket's and Kalshi's endpoints, along with emitted events, are used to analyze slippage and arbitrage using these measures.

Finally, Chapter 7 discusses and interprets the findings of this thesis.

Chapter 3

Endpoints and Clients

3.1 Polymarket Event and Market Endpoints

Polymarket defines a market object for every prediction market, identified by a unique market ID. Every market is part of an *event*, and every event can be associated with multiple *tags*. The *gamma-api.polymarket.com* domain hosts several HTTP endpoints to query market, event, or tag objects using pagination. The documentation provides only an example use case, a list of query parameters, and a sample request and response.

The most essential query parameters are *offset* and *limit*, which enable batch access. The limit specifies the number of objects in a batch, while the offset determines the starting point. This implicitly assumes an ordered list of objects starting at offset zero. An *order* parameter also exists but lacks detailed specification. Testing revealed that sorting by many attributes fails to order the results correctly, omits markets, or returns duplicates at different offsets. Querying event objects ordered by event ID avoids these anomalies and was recently added as an example in the documentation. However, ordering market objects by ID is not possible.

Testing also revealed an undocumented constraint: the limit parameter is capped at 500. Any value exceeding this simply returns 500 objects starting at the specified offset. Before December 2025, the response behavior for requesting an offset beyond the available range was unspecified. It was eventually clarified in December 2025 that such requests return an empty JSON array.

3.1.1 Event and Market Objects

The market object is listed with over 100 attributes in the documentation. Initial testing showed a mismatch between the attributes in actual responses and those listed in the documentation; furthermore, the attributes vary between responses. For this reason, this analysis covers all event objects (which include associated market and tag objects) obtained from the *events* endpoint.

Since JSON is specified by a deterministic context-free language, it can be parsed unambiguously into a tree of nodes, where every node has a type. For every event object, a tree is constructed where every node (except the root) is a child of either an object or an array. If a node is a child of an object, it must have an associated string key. A node is a leaf if it is an empty array, an empty object, or a scalar type (number, string, boolean, or null).

For a more representative type analysis, numbers are further categorized as IEEE 754 floating-point values or integers. Strings are classified as empty, whitespace-only, JSON-encoded, standardized timestamps, numeric strings, hexadecimal, case-insensitive literals ("null", "none", "nan", "true", "false"), or *general* strings. The resulting table showing the type distribution and occurrence frequency for every tree path is available in Appendix C.1.

3.2 Polymarket Orderbook and Trade Endpoints

Polymarket provides a *book* HTTP endpoint to request the current orderbook state under the *clob.polymarket.com* domain. It also provides a *market* WebSocket endpoint to receive state change events under the *ws-subscriptions-clob.polymarket.com* domain, referred to as the *market feed*. Every prediction market is associated with two orderbooks: one for the YES outcome and one for the NO outcome. An orderbook has a side for buying the underlying outcome token (bids) and a side for selling the underlying outcome token (asks). The orderbook's underlying outcome is uniquely identified by a *token ID*, also referred to as an *asset ID*. Every market object contains two asset IDs within a JSON-encoded string array under the key "clobTokenIds" C.1.

3.2.1 Book Endpoint

The specification for the book endpoint consists only of a list of request and response JSON attributes, along with samples for both. This endpoint returns the orderbook state objects for the requested asset IDs. The documentation does not specify the maximum number of asset IDs that can be requested, the limit on returned orderbooks, or the behavior when requesting the orderbook of an already resolved market. Testing revealed that orderbook states are returned in the order requested. However, any orderbook state object requested for an asset ID whose market is already resolved is omitted from the response. Additionally, the response contains at most 500 orderbook state objects.

The orderbook state object shown in Appendix A.1 is based on the sample provided in Polymarket's documentation. Notably, the timestamp attribute is described as a string but appears in responses as a string representation of a Unix millisecond timestamp.

3.2.2 Falsely Missing Orderbook State Objects

An initial client making concurrent requests to the book endpoint revealed an anomaly where the endpoint failed to return orderbook state objects for specific asset IDs, only to include them in subsequent responses. This effectively made the orderbook state unavailable, mimicking a resolved market. Crucially, this was not caused by rate limiting, as the endpoint did not return error status codes.

The same setup was run for several hours using strictly serialized requests, and this anomaly was not detected. To investigate further, every asset ID was mapped to a *false miss* counter. This counter incremented whenever the orderbook state object for a requested asset ID was missing from the response; if the object appeared, the counter was reset to 0.

This experiment revealed an initial burst of false misses for many asset IDs, though the counts remained below 13. After this initial burst, the number of asset IDs experiencing false misses decreased significantly, but the reported miss counts showed much larger variance, with some exceeding 1,000 false misses.

3.2.3 Market Feed

The market feed broadcasts event messages for subscribed markets. Prior to December 17, 2025, this endpoint allowed only a single subscription per established WebSocket connection. Consequently, if any markets were resolved or created, a new WebSocket connection had to be established to adapt to the change. This limitation was undocumented; testing simply revealed that sending more than one subscription message stopped all event delivery. Even with a 30-second timeout, the server kept the underlying persistent TCP and WebSocket connection alive without sending further messages, effectively turning it into a zombie connection.

On December 17, dynamic subscription and unsubscription messages were announced via Discord, confirming that this functionality was previously unavailable. This was a significant improvement, as the prior client required a new WebSocket connection whenever new markets were created. Another major improvement occurred around January 5, 2026, when lifecycle updates were added to the market feed. Previously, the market or events endpoint had to be scraped continuously to check for new or resolved markets, introducing unnecessary latency and complexity. Comparing market objects from a long-lived connection (continuously scraped for lifecycle changes) against a separate, one-time exhaustive scrape revealed that some market objects were never returned at any offset on the market endpoint for the long-lived connection. This anomaly was only measurable after three days.

Unlike the book, market, and event endpoints, the market feed's documented response attributes appear in every response and are of the correct type. Sample messages can be found in the Appendix (A.3 for price changes, A.2 for

book events, A.4 for trade events, A.5 for new market events, and A.6 for market resolved events). However, the last trade price event contains an undocumented attribute *tx_hash* of type string, which is always a hexadecimal value with the length of an Ethereum transaction hash. The timestamps for market feed events are string representations of Unix millisecond timestamps, matching those found in the book endpoint responses. However, because timestamps only have millisecond resolution and there are no other attributes to order events, the state becomes ambiguous when multiple conflicting events occur within the same millisecond.

3.2.4 Ambiguous Event Ordering

The most obvious conflict arises between events of the same type occurring at the same millisecond with different resulting states. For price change events, this occurs when two or more events share the same timestamp, side, price, and asset ID, but have different sizes. In such cases, it is unclear what the exact size is at the given price, side, and asset ID at or after that timestamp. Book events conflict if they share the same timestamp and asset ID but list different bids and asks. Similarly, last trade events conflict when they share the same timestamp, asset ID, side, and price, making it impossible to differentiate which trade was filled at which size. Table 3.1 illustrates that this occurs in practice.

Table 3.1: Bitcoin Up or Down - January 7, 11:00AM-11:15AM ET

Message Type	**Total**	**Ambiguous (%)**	**Duplicate (%)**
Asset ID: 113615...8317859			
Last Trade Prices	3,562	24 (0.67%)	0 (0.0%)
Price Changes	158,662	3,228 (2.04%)	1,336 (0.84%)
Books	6,074	0 (0.0%)	0 (0.0%)

If we additionally consider the ambiguity arising when a price change message and a book summary message share the same timestamp (i.e., it is unclear whether the price change applied before or after the book state summary), we find that 7,160 price changes are ambiguous.

3.3 Final Polymarket Client

The initial client was designed to subscribe to the market feed and make orderbook state requests for all existing unresolved markets. This involved over 30,000 orderbooks. Since only 500 orderbooks can be requested at once, there is a clear trade-off between the number of orderbooks monitored and the frequency of updates. Additionally, since market feed events cannot be unambiguously ordered,

frequent requests for the actual orderbook state become even more valuable. For these reasons, the final client was designed to monitor fewer than 500 orderbooks and request their states sequentially, with a minimum interval of 500ms between requests. The documentation specifies a rate limit of 500 requests/s, so this setup is very conservative. For the event "Which party will win the House in 2026?", there are four orderbooks, while the cyclical 15-minute event "Bitcoin up or down?" typically has fewer than 100 active orderbooks.

The final client initially scrapes the events endpoint for all currently available 15-minute cyclical "Bitcoin up or down?" markets. It then subscribes to the market feed using these IDs and the four known orderbooks from "Which party will win the House in 2026?". It also listens for newly created or resolved "Bitcoin up or down?" markets to dynamically manage subscriptions. The client also initiates the orderbook state requests as specified above and persistently stores the price change, book, last trade price events, and orderbook state objects in a database.

3.4 Kalshi Endpoints

Kalshi provides endpoints similar to those on Polymarket. These include the *events* endpoint for scraping any market ever created and a WebSocket endpoint with different channels, such as *trades*, *orderbook updates*, and *market lifecycle*. Markets are uniquely identified through their *market_ticker*. These tickers allow for dynamic subscription and unsubscription for the channels of interest on the WebSocket endpoint.

However, unlike Polymarket, Kalshi provides a single orderbook for both outcomes of a prediction market. One side represents the YES prices and the other the NO prices, as seen in the documented orderbook delta A.7 and orderbook snapshot A.8 responses. Other documented responses are available in the Appendix (A.11 for event objects, A.10 for market lifecycle events, and A.9 for trade updates).

The JSON objects within the messages sent by the orderbook update and trade channels contain a unique integer sequence identifier. This identifier can be used to unambiguously order the events for a given channel and spot any missing events, as the first event after subscription has sequence identifier zero, and each subsequent event increments the identifier by one. Although Kalshi's documentation for the trade update message is missing the *seq* attribute, the messages effectively contain it. However, ordering trade events relative to orderbook events remains ambiguous, as it relies on timestamps that are only specified in seconds for trades.

3.5 Kalshi Client

Due to the sequence identifiers provided by Kalshi, there is no need to request the orderbook state, as the identifiers allow for unambiguous maintenance of the orderbook state.

This client, along with the Polymarket client, ran from January 8th 2026 until February 2nd 2026.

Chapter 4

Smart Contracts

Polymarket maintains a GitHub organization with over 90 repositories. Many individual repositories contain the Solidity source code for smart contracts and their deployment details. They also maintain a dedicated repository listing deployment addresses, associated contracts, and security audits, if available [14]. Although not explicitly labeled as exhaustive, a review of the website documentation and GitHub repositories revealed no undeclared deployments. In the absence of another authoritative source, this list serves as the reference for disclosed deployments.

4.1 Conditional Tokens

The Gnosis Conditional Tokens Framework (CTF) extends the ERC-1155 standard to construct tokens representing the outcomes of an event or condition. This enables the transfer, minting, and burning of multiple outcome tokens within a single transaction.

The contract defines the relationship between the value of the n outcome tokens $(O_1, \ldots, O_n)$ for an event and the ERC-20 collateral token Q as:

$$Q = \sum_{i=1}^{n} O_i \tag{4.1}$$

The *splitPositions* method atomically swaps a specific amount of collateral Q for an equal amount of each outcome token and emits the *PositionSplit* event. The *mergePositions* method performs the reverse atomic swap and emits the *PositionsMerge* event.

Before these methods can be used, the condition must be prepared using the *prepareCondition* method with an oracle address, a question ID, and an outcome count. This emits the *ConditionPreparation* event. Every condition is assigned a unique identifier defined as:

$$condition_{id} := keccak(oracle \parallel question_{id} \parallel \text{outcome count}) \tag{4.2}$$

Subsequently, only the oracle can report payouts using the *reportPayouts* method by passing an array of unsigned integers $payouts = (p_1, \ldots, p_n)$ corresponding to the n outcome tokens. This emits the *ConditionResolution* event and assigns each outcome token O_i a new value:

$$O_i := Q \cdot \frac{p_i}{\sum_{k=1}^{n} p_k} \tag{4.3}$$

After this assignment, Equation (4.1) still holds.

Finally, after the payout is reported, anyone can redeem their outcome tokens for the value assigned in Equation (4.3) using the *redeemPositions* method, which emits the *PayoutRedemption* event.

4.2 UMA Framework

UMA designs oracle protocols and deploys Ethereum-based contracts on various blockchains, such as Polygon. These deployments connect to UMA's dedicated web application, allowing users to actively participate in resolving price requests.

Polymarket exclusively used UMA's OptimisticOracleV2 (OOV2) deployment until August 2025, when the UMIP-189 governance proposal passed. Subsequently, UMA deployed a ManagedOptimisticOracleV2 (MOOV2) dedicated exclusively to Polymarket usage. The ManagedOptimisticOracleV2 is based on the same protocol but includes additional access control configurations [15].

Polymarket atomically sets price requests to be event-based within its *UmaCtfAdapter* contract. The process begins with a price request, which emits the *RequestPrice* event. For a price request to be settled, the effective price must first be determined. This requires an initial price proposal suggesting a value for the request, which emits the *ProposePrice* event. Once a price proposal is made, it can be disputed within a liveness period, which defaults to two hours. If the proposal remains undisputed during this period, it becomes the effective price, and the request can be settled.

In OOV2, neither proposals nor disputes have access control restrictions. If the proposal is disputed, the *DisputePrice* event is emitted, initiating a stake-based voting process. The contract governing this process is deployed on the Ethereum blockchain, necessitating an off-chain coordinated message relayer. Once the price is determined, the request is settled. This involves rewarding or slashing the proposer and disputer (if present), depending on the accuracy of their proposals relative to the final outcome. This emits the *Settle* event.

The OOV2 protocol does not inherently specify the semantic meaning of the prices. Therefore, in the context of Polymarket, *ancillary data* must formally specify the semantic meaning of the integer price values. Although Polymarket does not explicitly state this, an analysis of emitted events reveals that virtually

all ancillary data extends the UMIP-107 standard [16]. Specifically, the requested price identifier is *YES_OR_NO_QUERY*, and the ancillary data is a UTF-8 encoded dictionary. This dictionary specifies the question using the *q* key, the associated market via the *marketId* key, and the four possible prices through the *p1*, *p2*, *p3*, and *p4* keys. These keys map specific prices to the outcomes *YES*, *NO*, *UNKNOWN*, and *EARLY_PROPOSAL*.

4.3 UmaCtfAdapter

This is Polymarket's custom contract, built on top of the CTF and OOV2, whose addresses are stored as immutable references. It provides an *initialize* method that atomically prepares the condition on the CTF and requests the price from the OOV2. When preparing the condition, it uses itself as the oracle and a hardcoded outcome count of two. The *initialize* method is permissionless and emits the *QuestionInitialized* event.

The other permissionless method is *resolve*, which emits the *QuestionResolved* event. It can only be invoked if the market's requested price has been determined and the contract is not paused. This method calls the OOV2 to settle and retrieve the *int256* price. If the price does not correspond to the *EARLY_PROPOSAL* outcome, it is used to report the payouts on the CTF. The *EARLY_PROPOSAL* outcome is hardcoded as the minimum *int256* value for comparison. A price of 0 is interpreted as *NO*, a price of 0.5 Ether is interpreted as *UNKNOWN*, and any other price is considered *YES*. Based on the outcome, a specific payout array is passed to the CTF: '[0,1]' for *NO*, '[1,1]' for *UNKNOWN*, and '[1,0]' for *YES*.

All other public methods in this contract are restricted to either an *onlyAdmin* or *onlyOracle* role. This includes the *resolveManually* method, which allows an authorized party to report a payout array directly without settling a price request.

4.4 CTFExchange

The CTFExchange is another custom contract by Polymarket that provides the *matchOrders* method. This method is used for settling a match on the CLOB between a taker order and n maker orders. It emits one *OrdersMatched* event for the overall match and $n + 1$ *OrderFilled* events. One *OrderFilled* event corresponds to the taker's side of the trade, while the other n events correspond to the makers' sides. There are three types of matches: split, merge, and complementary.

A split match occurs when the taker and makers are both buyers, one for YES shares and the other for NO shares. Analogously, a merge match occurs when the taker and makers are both sellers, one for YES shares and the other for NO shares. A complementary match happens when the taker is selling the shares

that the makers are buying, or vice versa. The match type and the amount of instruments exchanged can be deduced from the ‘OrderFilled‘ events.

4.5 Negrisk

The Negrisk framework builds on top of the UmaCtfAdapter to create multiple markets with mutually exclusive outcomes. It provides a *convertPositions* method, which enables atomically swapping a set of YES shares for the complementary set of NO shares, or vice versa. This functionality is implemented in the Negrisk Adapter.

However, the framework also contains a Negrisk Operator, which the Negrisk UmaCtfAdapter uses as an immutable reference for the conditional tokens contract interface. Notably, the Negrisk Operator's ‘prepareCondition‘ method is empty, and its ‘reportPayouts‘ method does not call any other contract. This implies condition preparation and price requests for a Negrisk market are not atomically executed within the same transaction, unless a hidden contract facilitates this.

Additionally, there is a distinct Negrisk CTFExchange deployment. This is simply another instance of the CTFExchange contract that uses the Negrisk Adapter for splitting and merging. It also uses its own collateral token, which is a wrapper around USDC.

Similarly to the UmaCtfAdapter, the Negrisk Operator has a *emergencyResolveQuestion* method to manually report payouts without settling a price request through UMA.

4.6 Polygon Collection

The data collection process uses the Infura eth_getLogs endpoint. Infura has a rate limit of 500 credits/s, with a request to the eth_getLogs requiring 255 credits according to the documentation. However, testing showed that the actual rate limit between consecutive requests without rate limiting is approximately 1s. The round-trip time (RTT) for a single eth_getLogs request can exceed 30s, making it suitable for concurrent requests. Additionally, there is a limit of 10,000 events per request. Requests are specified using a block range. To manage this constraint, we use block density estimation with a margin of 500 blocks to target approximately 9,500 events per request. This seems to be effective as for most events the block density does not seem to have strong variance. But the margin can be increased if some events have a greater block density variance. As a sanity check, the number of events returned was compared against those found in Dune Analytics over the same block range. For consistency, the block

range from 79,172,086 (inclusive) to 81,225,971 (inclusive), spanning November 18, 2025 07:00:03 UTC to January 5, 2026 00:00:01 UTC, was used for all analytics in Chapters 5 and 6. The complete list of collected events is provided in Appendix B.1.

Chapter 5

Market Lifecycle

5.1 Market Creation

5.1.1 Reporting Oracles

As explained in Section 4.1, the reporting oracle for a condition is passed as an argument during condition preparation on the CTF. This address is the only one that can report payouts to determine the final YES and NO share values after resolution. This makes it essential to analyze the reporting oracles used by Polymarket.

Out of 128,003 ‘ConditionPreparation‘ events, 124,340 have a matching off-chain market object with the same condition ID. These matched events use three distinct reporting oracles, two of which are undocumented: the *MOOV2 Adapter* and the *Centralized Adapter*, detailed in Appendix B.2.

The MOOV2 Adapter is an ‘UmaCtfAdapter‘ instance as described in Section 4.3, but it uses the MOOV2 instead of the OOV2. Interestingly, no condition was prepared with the OOV2 Adapter as the reporting oracle. This aligns with the UMIP-189 proposal and suggests that the MOOV2 Adapter has replaced the OOV2 Adapter, although Polymarket has not disclosed this change.

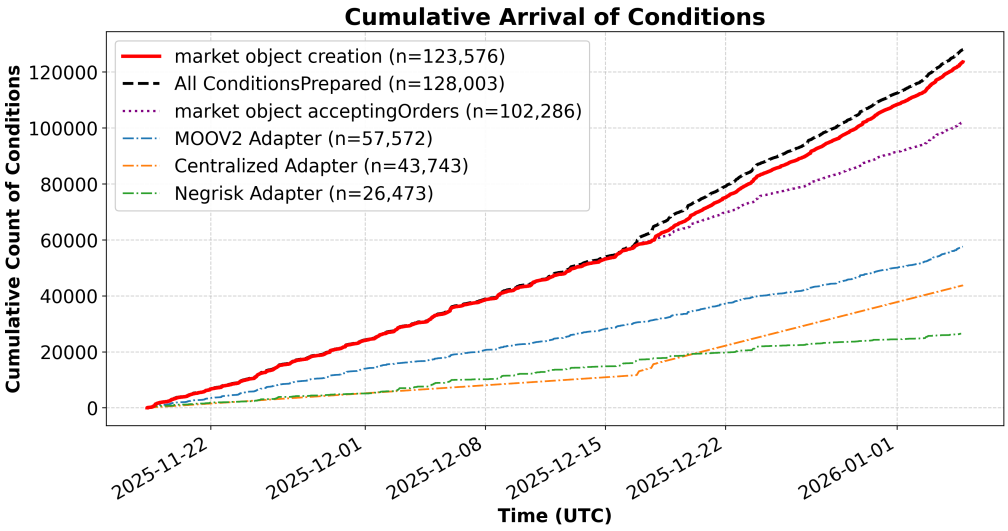

Figure 5.1: Cumulative count of conditions found over time

Figure 5.1 shows that around December 17, 2025, some market objects began to appear without an ‘acceptingOrders‘ timestamp, a detail also noted in Appendix C.1. This period coincides with the 3,663 conditions that were prepared on the CTF with no matching off-chain market object. Virtually all of these use the Centralized Adapter as the reporting oracle, and the rate of new conditions prepared with this adapter also changed at this time.

5.1.2 Centralized Adapter

The bytecode for the Centralized Adapter is not recognized by prominent bytecode-to-source indexers like Polygonscan or Etherscan. However, we can still use the matched market objects to analyze this adapter in more detail.

Table 5.1: Label frequency in 41,016 market objects matched with the Centralized Adapter as reporting oracle

Label	**Appeared in (%)**
Up or Down	41,016 (100%)
Crypto Prices	41,016 (100%)
Hide From New	41,016 (100%)
Recurring	41,016 (100%)
Crypto	41,016 (100%)
5M	21,630 (52.7%)
15M	18,270 (44.5%)
Solana	10,263 (25.0%)
Ethereum	10,257 (25.0%)
XRP	10,256 (25.0%)
Ripple	10,256 (25.0%)
Bitcoin	10,240 (25.0%)
4H	1,116 (2.7%)

Every market object has a set of labels assigned by Polymarket to categorize the event. Table 5.1 shows the label frequency in market objects matched with conditions using the Centralized Adapter. Clearly, all labels are tied to crypto-asset, price-related events. When viewing these matched markets on Polymarket's web application, none of them link to the UMA resolution status page, which exists for other markets. Instead, they link to a source that tracks the underlying crypto asset price, and this source is used directly as the ground truth for resolution. The cyclical 15-minute "Bitcoin up or down?" markets are an example of this.

5.1.3 Resolution Source

While the MOOV2 Adapter makes a price request atomically with condition preparation, a condition prepared for a Negrisk does not, as explained in Section 4.5. 'ConditionPreparation' and 'RequestPrice' events have no direct, shared identifier. However, virtually every ancillary data field contains a 'marketId' key, which can be used to match price requests to market objects and subsequently to 'ConditionPreparation' events. Out of 84,684 'RequestPrice' events, 83,743 have a matching off-chain market object, and 83,652 have a matching 'ConditionPreparation' event.

Table 5.2: Resolution Source and Reporting Oracle Pairs

Resolution Source (ids)	Matches	Oracle (Conditions)
MOOV2	57,502	MOOV2 Adapter
OOV2	70	MOOV2 Adapter
OOV2	26,080	Negrisk Adapter

A curious anomaly is that 70 price requests made to the OOV2 matched with conditions using the MOOV2 Adapter as the reporting oracle. Of these, 50 do not share the same transaction hash, indicating they were not executed atomically via the ‘initialize‘ method.

The conditions prepared with the Negrisk Adapter as the reporting oracle matched exclusively with price requests made to the OOV2. None of these matches share the same transaction hash, confirming they are not executed atomically. However, 21,672 (83.10%) share the same block number, which demonstrates that while the operations are coordinated through separate transactions, they are typically executed in a timely manner.

5.2 Exchange Registration

For matched orders on the CLOB to be settled on the on-chain exchange, the condition ID and the two token IDs of the prediction market must first be registered, as explained in Section 4.4.

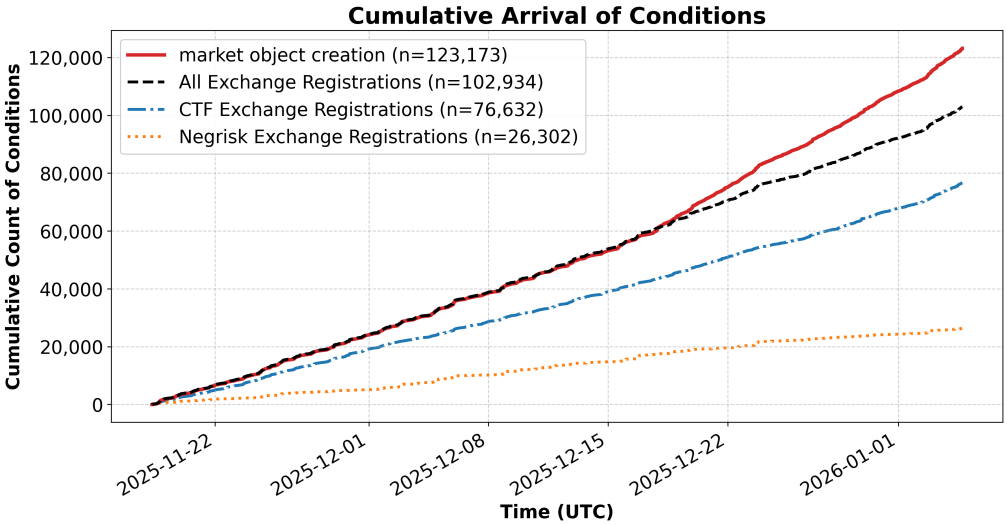

Figure 5.2: CLOB markets and registered tokens unique condition count

Figure 5.2 shows that after a certain point in time, markets are consistently being created without being registered with the on-chain exchange. This trend begins around the same time that market objects start being created without ‘acceptingOrders‘ timestamps.

5.2.1 Unregistered Traded Outcomes

When matching the ‘ConditionsPrepared‘ events to the ‘RegisteredToken‘ events, we find that virtually all unregistered markets were prepared using the Centralized Adapter as the reporting oracle. Overall, less than half of the conditions prepared using the Centralized Adapter as the reporting oracle are registered with an exchange.

Table 5.3: Matched conditions of 134,424 Conditions Prepared and 215,152 TokensRegistered over Nov 18 - Jan 08

Oracle (Conditions)	**Matches**	**Exchange (Conditions)**
Centralized Adapter (48,896)	20,425	CTFE (83,262)
MOOV2 Adapter (62,857)	61,439	CTFE (83,262)
Negrisk Adapter (27,728)	23,545	Negrisk CTFE (24,209)

Some of these markets are visible on the Polymarket web application, listed as active in the market object, and show trading activity on the WebSocket market feed, yet they lack any corresponding ‘TokensRegistered‘, ‘OrdersMatched‘, or ‘OrderFilled‘ events emitted from either the CTFE or the Negrisk CTFE. The cyclical 15-minute “Bitcoin up or down?” markets serve as an example of this behavior, which demonstrates that Polymarket is silently becoming more centralized.

5.3 Resolution

5.3.1 Disputes

As described in Section 4.2, the UMA OOV2 and MOOV2 protocols allow price proposals to be disputed within a liveness period.

There is a total of 74,833 matched proposals and 340 matched disputes. Table D.1 shows the ten most frequent labels assigned to the market objects matched with the disputes. Out of the 340 matched disputes, 145 occurred on the OOV2 and 195 on the MOOV2. This is notable, considering that the MOOV2 was explicitly designed with a proposer whitelist managed by Polymarket under the UMIP-189 proposal. Despite the whitelist, the MOOV2 has a higher raw num-

ber of disputes than the OOV2. However, it is important to consider the total volume: the MOOV2 also has 53,818 proposals compared to the OOV2's 21,015.

5.3.2 Reported Unknown Outcome

The terminal value for the YES and NO shares is determined by the payout array passed from the reporting oracle to the CTF, which then emits a 'ConditionResolution' event containing the condition ID and the payout array.

Out of 112,638 'ConditionResolution' events, 283 (0.25%) reported an UNKNOWN outcome, 46,373 (41.17%) reported a YES outcome, and 66,982 (58.58%) reported a NO outcome. If the YES and NO outcomes were uniformly distributed, they would be reported equally. However, most Negrisk events are designed such that one prediction market resolves to YES and the remaining ones to NO. In fact, out of the 19,541 'ConditionResolution' events with a Negrisk Adapter as the reporting oracle, only 2,616 (13.39%) reported a YES outcome, 16,925 (86.61%) reported a NO outcome, and none reported an UNKNOWN outcome. All reported UNKNOWN outcomes were generated by the MOOV2 Adapter.

Table D.2 shows the proportion of labels associated with markets reported to have an UNKNOWN outcome.

5.3.3 Manual Resolutions

Sections 4.3 and 4.5 mentioned that the UmaCtfAdapter and Negrisk Operator contracts include administrative methods that allow for manual market resolution without requiring UMA price requests to be settled. All relevant administrative methods emit events, including the 'resolveMarketManually' method in the UmaCtfAdapter and the 'emergencyResolveQuestion' method in the Negrisk Operator. However, no events for either of these manual resolution methods were emitted in any of the UmaCtfAdapter instances or the single Negrisk Operator instance during the collection period. This observation does not fully exclude the possibility of manual resolution, but if it occurred, it must have been executed through a delegate call from a hidden proxy contract, which would result in the events being emitted from that proxy contract instead.

Chapter 6

Orderbook and Trading

Section 1.1 explained how a prediction market consists of YES and NO shares whose price is denoted in USD. The Central Limit Order Book (CLOB) used by Polymarket and Kalshi captures the intention of traders through market and limit orders. These orders are placed via HTTPS requests to *order* endpoints and are matched based on price-time priority [17, 18]. A limit order allows the trader to specify the price at which they want to buy or sell shares. This price cannot cross other resting limit orders; otherwise, the order becomes marketable. A marketable order is either executed immediately against the best possible resting limit orders by price-time priority or is aborted [19, 20].

6.1 State, Ordering, and Matching

A trader can buy or sell YES shares or buy or sell NO shares, which creates four possible order types: Sell Yes, Sell No, Buy Yes, and Buy No. However, these four types represent only two core intentions: speculating on the YES outcome by buying YES shares or selling NO shares, or speculating on the NO outcome by buying NO shares or selling YES shares.

Kalshi maintains a single orderbook for each prediction market, capturing order intentions through a YES and a NO side. Polymarket, on the other hand, provides separate orderbooks for the YES and NO shares, each with bid and ask sides. However, analyzing the orderbook state reveals that the bid share amount at any given price p for the YES shares is always the same as the ask share amount for the NO shares at price $1 - p$. The analogous relationship also holds for the YES asks and NO bids.

The state of the CLOB is defined by the resting limit orders. A limit order can be defined as a tuple (p, v, s, i), where $p \in \mathbb{R}^+$ is the price per share for the taker, $v \in \mathbb{R}^+$ is the quantity of shares, $s \in \{\text{YES}, \text{NO}\}$ is the outcome, and $i \in \mathbb{N}$ is the priority for selection when matching with market orders at price p and side s. A lower i indicates a higher priority. This guarantees a unique ordering for both buy and sell limit orders used for matching. The taker price p is only the

same the matched maker price if their complementary. Otherwise, in the case that the match is a merge or split, the taker price p equals one minus the maker price. In any case, the order for limit orders by taker price and time priority is:

$$((p_1, v_1, s_1, i_1) > (p_2, v_2, s_2, i_2)) \iff p_1 < p_2 \vee (p_1 = p_2 \wedge i_1 < i_2) \tag{6.1}$$

The best price for the taker obviously is the lowest, which by ordering 6.1 is the price of the greatest limit order:

$$p_{\text{market}} = \max\{p|(p, v, s, i)\} \tag{6.2}$$

If the quantity for a marketable order cannot be fully filled at the best market price, the taker suffers from price slippage if the order is executed. If the marketable order cannot be filled by limit orders at any price, the taker order remains partially or fully unfilled. Depending on the market order specification, a partially unfilled market order is either executed or canceled.

6.2 Slippage

The CLOB is designed to allow this, with the core principle being that the taker is guaranteed timely order execution but not price, while the maker is guaranteed order price but not timely execution. We can define the slippage for a taker buying or selling x base assets and getting matched with maker orders M as:

$$\text{Slippage}(x, M) = \left(\sum_{(p,v,s,i) \in M} p \cdot v \right) - p_{\text{market}} \cdot x \tag{6.3}$$

The slippage can only be positive or zero by definition 6.2, and the least limit order $(p, v, s, i) = \min\{M\}$ might only be partially consumed. The total quantity executed, x, is the sum of the quantities from all matched maker orders:

$$x = \sum_{(p,v,s,i) \in M} v \tag{6.4}$$

6.2.1 Power Relation between USDC Taker Amount and Slippage

We can match the OrderFilled events emitted by the CTFExchange by their transaction hash, to obtain 43,970,817 settlements. Then we can use the OrderFilled events to obtain the maker orders M, share quantity x and p_{market} for a

given settlement using algorithm D.1, such that slippage can be computed using definition 6.3.

There have been many studies on the power relation $y = \beta x^{\alpha}$ between order size x and slippage y in traditional financial markets with an orderbook as market mechanism [21, 22, 23].

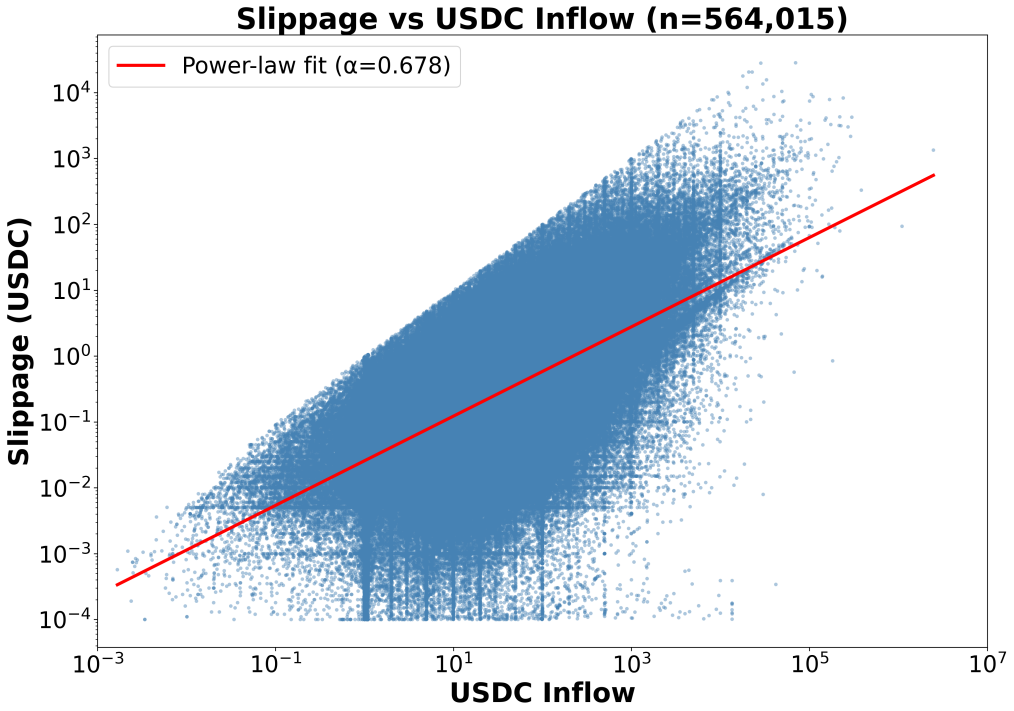

Figure 6.1: Linear Regression on Log Slippage and Log USDC Inflow, with slippage starting at 10^{-4} USDC

In the figure, we can see the linear regression on $\log(y) = \alpha \cdot \log(x) + \log(\beta)$. The tolerance of 10^{-4} USDC slippage resulted from an initial observation when plotting all the log-transformed USDC taker amounts and slippage for settlements with more than two OrderFilled events. The plot in Appendix D.1 shows this, where we can see three clearly distinguishable clusters. One reason could be that Polymarket dynamically adjusts the smallest price per share possible, which according to documentation usually is 0.01 but in case the prices are close to 0 or 1 can also become smaller. Another could be quantization and rounding errors related to how numbers are converted, since the prices and share quantities are in floating-point numbers on the API, but stored as integers on chain by multiplying the USDC amounts and share quantities by 10^6. There have been complaints about numeric discrepancies in order execution directly as well.

The regression results in $\beta = 2.571 \times 10^{-2} \pm 9.675 \times 10^{-3}$, $\alpha = 0.6778 \pm 2.487 \times 10^{-2}$, $R^2 = 0.3360$. This suggests that, as an aggregate, the settlements clearly follow a power-law relation, but as the R^2 suggests, it is arguably a poor predictor for individual settlements. The α is relatively high for a CLOB, where values around 0.5 are expected, also known as the square-root law of market impact [21, 22, 23]. The coefficient β follows an even more extreme trend, being orders of magnitude higher than what is found in comparable markets [24]. However, it is important to consider that less than 2% of settlements have a slippage above 10^{-4} USDC, which would suggest that a slippage occurring is a rare event, but if it does occur its expected to be greater than what is observed in comparable markets.

6.2.2 Comparison with Decentralized AMM Slippage

Polymarket's CLOB differs from decentralized AMMs like Uniswap V2 in slippage behavior. In Uniswap V2, slippage follows the constant product formula $x \cdot y = k$ and scales linearly with trade size relative to liquidity, whereas Polymarket exhibits lower slippage for most trades. Polymarket's CLOB achieves a slippage of less than 10^{-4} for over 98% of settlements, though exhibiting high variance. Additionally, on-chain AMMs suffer from MEV extraction via sandwich attacks and flashbots, where validators and bots front-run and back-run transactions for profit [25, 26]. Polymarket's off-chain CLOB with on-chain settlement avoids protocol-level MEV extraction, meaning observed slippage reflects only order book consumption without adversarial extraction.

6.3 Arbitrage

For the 15-minute cyclical "Bitcoin up or down?" markets, there are four shares in total. We can partition these into the YES share partition {YES_Kalshi, YES_Polymarket} and the NO share partition {NO_Kalshi, NO_Polymarket}. Every share within the same partition shares the same value, and the values of any pair of shares from different groups are complementary, summing to 1.

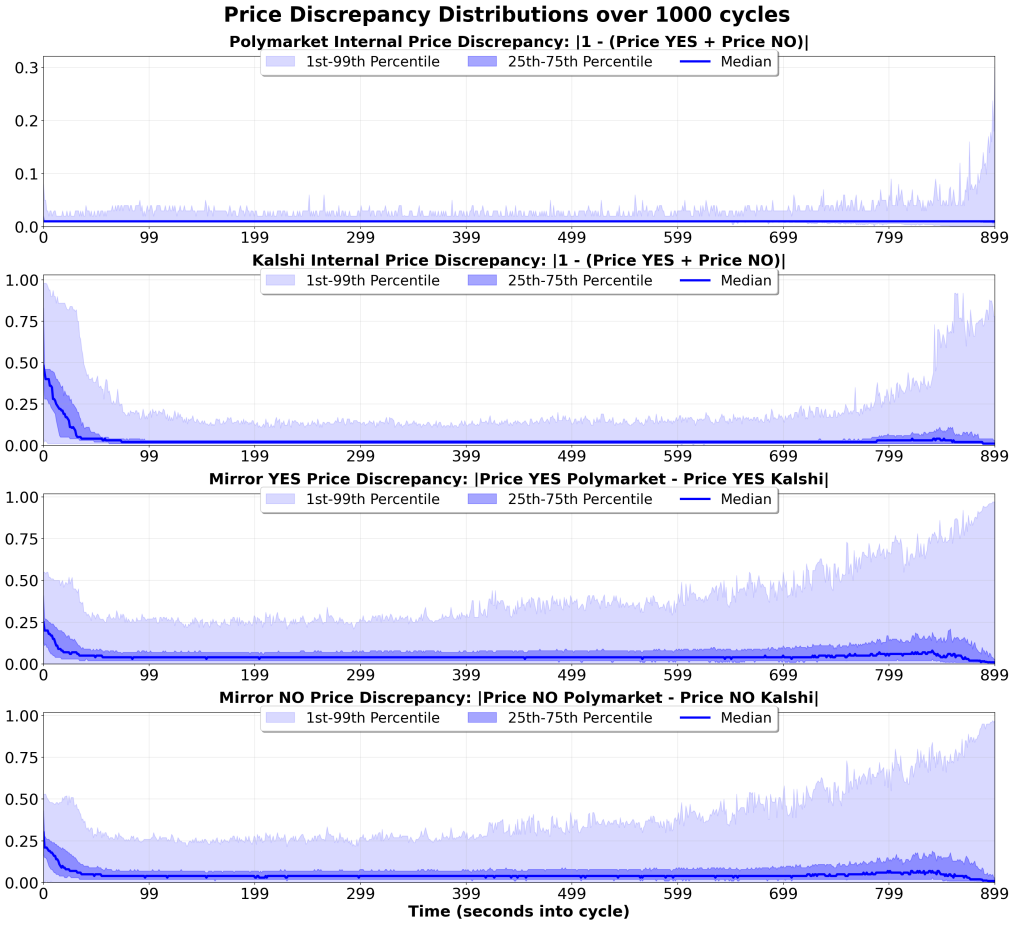

Figure 6.2: Price Discrepancy Distribution for 15-minute cyclical "Bitcoin up or down?" markets

Figure 6.2 shows the distribution of the price discrepancy between four of the six possible pairs of shares. It is important to note that for both Polymarket and Kalshi, the orderbook for a specific 15-minute interval was available before the start of the interval. The price discrepancy is lower when more Polymarket shares are involved, regardless of the point in time, but especially at the start of the interval. In all cases, the median price discrepancy is highest at the start of the interval. Towards the end of the cycle, the median remains relatively low, but the distribution becomes increasingly skewed.

Chapter 7

Discussion

7.1 Data Collection Barriers

The barrier for reliable data collection on Polymarket is relatively high as a result of the incomplete, fragmented and even incorrect documentation. This applies to both the off-chain endpoints and the on-chain smart contracts, where the exact coordination is abstracted away. Analyzing the on-chain events and off-chain market objects revealed just how much of the system actually is undocumented, including the transition to the MOOV2 Adapter without public disclosure, the existence of the Centralized Adapter with unrecognized bytecode, and recurring event ambiguities from millisecond-resolution timestamps. Additionally, the off-chain endpoints have demonstrated inconsistent behavior, such as false misses in orderbook requests and zombie connections on the WebSocket feed, which is difficult to assert since there is no specification for expected endpoint behavior in the first place.

7.2 Market Integrity and Effectiveness

Despite these documentation and consistency challenges, the market mechanisms appear effective. Comparing the market internal price discrepancy between Polymarket and Kalshi for the 15-minute cyclical "Bitcoin up or down?" markets suggests that arbitrageurs are maintaining lower price discrepancies on Polymarket compared to Kalshi, indicating tighter and more efficient pricing. However, this efficiency may be partially attributed to wash trading activity, as a recent study suggests approximately 18% or potentially up to 60% of December 2024 volume on Polymarket involved wash trades.

7.3 Slippage and Market Design

Polymarket's CLOB achieves over 98% of settlements incurring minimal slippage below 10^{-4} USDC. While the measured power relation exhibits worse impact co-

efficients than traditional markets, the off-chain CLOB with on-chain settlement avoids protocol-level MEV extraction inherent to decentralized AMMs, meaning observed slippage reflects only mechanical order book consumption. However, high variance in the slippage-to-volume ratio indicates that large orders face unpredictable price impact from order book depth constraints.

7.4 Unregistered Markets and Emerging Infrastructure

A notable finding is the emergence of unregistered prediction markets on Polymarket around December 2025, primarily using the Centralized Adapter as the reporting oracle. These markets are visible on the web platform, show trading activity, yet lack corresponding smart contract registrations and on-chain trade settlement. This suggests Polymarket is experimenting with markets that may not require on-chain settlement, potentially introducing infrastructure variability that complicates end-to-end analysis.

Bibliography

[1] Kalshi, "Payout on yes and no outcomes," 2026. [Online]. Available: https://help.kalshi.com/kalshi-101/what-are-prediction-markets

[2] Kalshi, "Kalshi rulebook," 2026. [Online]. Available: https://kalshi-public-docs.s3.amazonaws.com/regulatory/rulebook/Kalshi%20Rulebook%20v1.21.pdf

[3] J. Clinton and T.-F. Huang, "Prediction markets? the accuracy and efficiency of $2.4 billion in the 2024 presidential election," OSF Preprints, December 2025. [Online]. Available: https://osf.io/preprints/socarxiv/a3v5p

[4] J. Wolfers and J. E. Stiglitz, "Prediction markets: The lessons from the 2024 election," NBER Working Paper, December 2024, working Paper w33005. [Online]. Available: https://www.nber.org/papers/w33005

[5] G. Gottsegen, "In 2024, prediction markets called the presidential election before the polls could. now, they're mostly betting on sports," Morningstar, December 2024, https://www.morningstar.com/news/marketwatch/20241220282/in-2024-prediction-markets-called-the-presidential-election-before-the-polls-could-now-theyre-mostly-betting-on-sports.

[6] N. De, "Polymarket's main 2024 u.s. presidential election market sees $3.5b in volume," CoinDesk, November 2024. [Online]. Available: https://www.coindesk.com/policy/2024/11/06/polymarkets-main-2024-us-presidential-election-market-sees-35b-in-volume/

[7] A. Sirolly, H. Ma, Y. Kanoria, and R. Sethi, "Network-based detection of wash trading (november 06, 2025)," SSRN, November 2025. [Online]. Available: https://ssrn.com/abstract=5714122orhttp://dx.doi.org/10.2139/ssrn.5714122

[8] M. Cattaneo, "Polymarket's trading volume in november and december of 2025," Dune, January 2026. [Online]. Available: https://dune.com/queries/6452777?utm_source=share&utm_medium=copy&utm_campaign=query

[9] Kalshi, "Kalshi is now available in 140+ countries," Kalshi Blog, October 2025. [Online]. Available: https://kalshi.com/blog/kalshi-is-now-available-in-140-countries

[10] Markets Wiki. Qcx llc. [Online]. Available: https://www.marketswiki.com/wiki/QCX,_LLC

[11] CFTC. Qcx llc dcm exchange registry. [Online]. Available: https://www.cftc.gov/IndustryOversight/IndustryFilings/TradingOrganizations/49571

[12] Swiss Gambling Supervisory Authority (GESPA), "Liste der gesperrten zugänge (blocklist)," Official Publication, October 2025, accessed January 2, 2026. The specific PDF is dated October 21, 2025. [Online]. Available: https://www.gespa.ch/de/bekaempfung-illegaler-aktivitaeten/zugangssperre

[13] D. Stabile, "Kalshi sues illinois regulators over cease-and-desist order," Gaming Today, November 2025. [Online]. Available: https://www.gamingtoday.com/news/kalshi-sues-illinois-regulators-over-cease-and-desist-order/

[14] Polymarket, "Polymarket contract security," Github, November 2025. [Online]. Available: https://github.com/Polymarket/contract-security

[15] UMAprotocol, "Umip-189," Github, November 2025. [Online]. Available: https://github.com/UMAprotocol/UMIPs/blob/master/UMIPs/umip-189.md

[16] UMAprotocol, "Umip-107," Github, November 2025. [Online]. Available: https://github.com/UMAprotocol/UMIPs/blob/master/UMIPs/umip-107.md

[17] Kalshi, "Kalshi order queue position," February 2026. [Online]. Available: https://docs.kalshi.com/api-reference/orders/get-queue-positions-for-orders

[18] Polymarket, "Polymarket limit orders," February 2026. [Online]. Available: https://docs.polymarket.com/polymarket-learn/trading/limit-orders

[19] Kalshi, "Kalshi create orders," February 2026. [Online]. Available: https://docs.kalshi.com/api-reference/orders/create-order

[20] Polymarket, "Polymarket order placement," February 2026. [Online]. Available: https://docs.polymarket.com/developers/CLOB/orders/create-order

[21] R. Almgren and N. Chriss, "Optimal execution of portfolio transactions," *Journal of Risk*, vol. 3, no. 2, pp. 5–39, 2001.

[22] B. Tóth, Z. Patart, and J. D. Farmer, "The short-term dynamics of the price impact," *arXiv preprint arXiv:1112.4555*, 2011. [Online]. Available: https://arxiv.org/abs/1112.4555

[23] J. D. Farmer, B. Tóth, and R. Seager, "Hacienda: market impact and power laws," *SSRN Electronic Journal*, 2012. [Online]. Available: https://ssrn.com/abstract=2090352

[24] A. Hewelke-Fischl, B. Tóth, and J. D. Farmer, "On the market impact and limit order book dynamics," *arXiv preprint arXiv:1402.2348*, 2014. [Online]. Available: https://arxiv.org/abs/1402.2348

[25] P. Daian, S. Goldfeder, T. Kell, Y. Li, X. Zhao, I. Bentov, L. Breidenbach, and A. Juels, "Flash boys 2.0: Frontrunning, transaction reordering, and consensus instability in decentralized exchanges," in *2020 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2020, pp. 910–927.

[26] L. Heimbach and R. Wattenhofer, "Eliminating sandwich attacks with the help of game theory," in *Proceedings of the 2022 ACM on Asia Conference on Computer and Communications Security*. ACM, 2022, pp. 612–626.

Appendix A

Endpoint Documented JSON responses

Listing A.1: Polymarkets Documented Orderbook State Object Sample

```
{
  "market": "0x1b6f76e5b8587ee896c35847e12d
  11e75290a8c3934c5952e8a9d6e4c6f03cfa",
  "asset_id": "1234567890",
  "timestamp": "2023-10-01T12:00:00Z",
  "hash": "0xabc123def456...",
  "bids": [
    {
      "price": "1800.50",
      "size": "10.5"
    }
  ],
  "asks": [
    {
      "price": "1800.50",
      "size": "10.5"
    }
  ],
  "min_order_size": "0.001",
  "tick_size": "0.01",
  "neg_risk": false
}
```

Listing A.2: Polymarkets Documented Book Event

```
{
  "event_type": "book",
  "asset_id": "65818619657568813474341868652308942079804919
  2873804221928922111314087931254 22",
  "market": "0xbd31dc8a20211944f6b70f31557f1001557
  b59905b7738480ca09bd4532f84af",
  "bids": [
    { "price": ".48", "size": "30" },
    { "price": ".49", "size": "20" },
    { "price": ".50", "size": "15" }
```

```
  ],
  "asks": [
    { "price": ".52", "size": "25" },
    { "price": ".53", "size": "60" },
    { "price": ".54", "size": "10" }
  ],
  "timestamp": "123456789000",
  "hash": "0x0...."
}
```

Listing A.3: Polymarkets Documented Price Change Event

```
{
    "market": "0x5f65177b394277fd294cd75650044e32
    ba009a95022d88a0c1d565897d72f8f1",
    "price_changes": [
        {
            "asset_id": "71321045679252212594626385532706912
            75033272857194253228963137931245558399256 3",
            "price": "0.5",
            "size": "200",
            "side": "BUY",
            "hash": "56621a121a47ed9333273e21c83b660cff37ae50",
            "best_bid": "0.5",
            "best_ask": "1"
        },
        {
            "asset_id": "52114319501245915516055106046884 2
            09969926127482827954674443846427813813222426",
            "price": "0.5",
            "size": "200",
            "side": "SELL",
            "hash": "1895759e4df7a796bf4f1c5a5950b748306923e2",
            "best_bid": "0",
            "best_ask": "0.5"
        }
    ],
    "timestamp": "1757908892351",
    "event_type": "price_change"
}
```

Listing A.4: Polymarkets Documented Last Trade Price Event

```
{
"asset_id":"114122071509644379678018727908709560
226618148003371446110114509806601493071694",
"event_type":"last_trade_price",
"fee_rate_bps":"0",
"market":"0x6a67b9d828d53862160e470329ffea5246f3
38ecfffdf2cab45211ec578b0347",
"price":"0.456",
"side":"BUY",
"size":"219.217767",
"timestamp":"1750428146322"
```

```
}
```

Listing A.5: Polymarkets Documented New Market Event

```
{
    "id": "1031769",
    "question": "Will NVIDIA (NVDA) close above
    $240 end of January?",
    "market": "0x311d0c4b6671ab54af4970c06fcf
    58662516f5168997bdda209ec3db5aa6b0c1",
    "slug": "nvda-above-240-on-january-30-2026",
    "description": "This market will resolve to \"Yes\" if the
        official closing price for NVIDIA (NVDA) on the final
        trading day of January 2026 is higher than the listed price
        . Otherwise, this market will resolve to \"No\".\n\nIf the
        final trading day of the month is shortened (for example,
        due to a market-holiday schedule), the official closing
        price published for that shortened session will still be
        used for resolution.\n\nIf no official closing price is
        published for that session (for example, due to a trading
        halt into the close, system issue, or other disruption),
        the market will use the last valid on-exchange trade price
        of the regular session as the effective closing price.\n\
        nThe resolution source for this market is Yahoo Finance
        specifically, the NVIDIA (NVDA) \"Close\" prices available
        at https://finance.yahoo.com/quote/NVDA/history, published
        under \"Historical Prices.\"\n\nIn the event of a stock
        split, reverse stock split, or similar corporate action
        affecting the listed company during the listed time frame,
        this market will resolve based on split-adjusted prices as
        displayed on Yahoo Finance.",
    "assets_ids": [
        "76043073756653678226373981964075571318
        26728924813471736928451899592278932642 5",
        "31690934263385727664202099278545688007 7
        99199447969475608906331829650099442770"
    ],
    "outcomes": [
        "Yes",
        "No"
    ],
    "event_message": {
        "id": "125819",
        "ticker": "nvda-above-in-january-2026",
        "slug": "nvda-above-in-january-2026",
        "title": "Will NVIDIA (NVDA) close above ___ end of January
            ?",
        "description": "This market will resolve to \"Yes\" if the
            official closing price for NVIDIA (NVDA) on the final
            trading day of January 2026 is higher than the listed
            price. Otherwise, this market will resolve to \"No\".\n
            \nIf the final trading day of the month is shortened (
            for example, due to a market-holiday schedule), the
            official closing price published for that shortened
```

```
            session will still be used for resolution.\n\nIf no
            official closing price is published for that session (
            for example, due to a trading halt into the close,
            system issue, or other disruption), the market will use
             the last valid on-exchange trade price of the regular
            session as the effective closing price.\n\nThe
            resolution source for this market is Yahoo Finance
            specifically, the NVIDIA (NVDA) \"Close\" prices
            available at https://finance.yahoo.com/quote/NVDA/
            history, published under \"Historical Prices.\"\n\nIn
            the event of a stock split, reverse stock split, or
            similar corporate action affecting the listed company
            during the listed time frame, this market will resolve
            based on split-adjusted prices as displayed on Yahoo
            Finance."
    },
    "timestamp": "1766790415550",
    "event_type": "new_market"
}
```

Listing A.6: Polymarkets Documented Market Resolved Event

```
{
    "id": "1031769",
    "question": "Will NVIDIA (NVDA) close above $240
    end of January?",
    "market": "0x311d0c4b6671ab54af4970c06fc
    f58662516f5168997bdda209ec3db5aa6b0c1",
    "slug": "nvda-above-240-on-january-30-2026",
    "description": "This market will resolve to \"Yes\" if the
        official closing price for NVIDIA (NVDA) on the final
        trading day of January 2026 is higher than the listed price
        . Otherwise, this market will resolve to \"No\".\n\nIf the
        final trading day of the month is shortened (for example,
        due to a market-holiday schedule), the official closing
        price published for that shortened session will still be
        used for resolution.\n\nIf no official closing price is
        published for that session (for example, due to a trading
        halt into the close, system issue, or other disruption),
        the market will use the last valid on-exchange trade price
        of the regular session as the effective closing price.\n\
        nThe resolution source for this market is Yahoo Finance
        specifically, the NVIDIA (NVDA) \"Close\" prices available
        at https://finance.yahoo.com/quote/NVDA/history, published
        under \"Historical Prices.\"\n\nIn the event of a stock
        split, reverse stock split, or similar corporate action
        affecting the listed company during the listed time frame,
        this market will resolve based on split-adjusted prices as
        displayed on Yahoo Finance.",
    "assets_ids": [
        "76043073756653678226373981964075571318 2
        67289248134717369284518995922789326425",
        "31690934263385727664202099278545688007 7
        99199447969475608906331829650099442770"
```

```
    ],
    "winning_asset_id": "76043073756653678226373981964075571318267289248134717369284518995922789326425",
    "winning_outcome": "Yes",
    "event_message": {
        "id": "125819",
        "ticker": "nvda-above-in-january-2026",
        "slug": "nvda-above-in-january-2026",
        "title": "Will NVIDIA (NVDA) close above ___ end of January?",
        "description": "This market will resolve to \"Yes\" if the official closing price for NVIDIA (NVDA) on the final trading day of January 2026 is higher than the listed price. Otherwise, this market will resolve to \"No\".\n\nIf the final trading day of the month is shortened (for example, due to a market-holiday schedule), the official closing price published for that shortened session will still be used for resolution.\n\nIf no official closing price is published for that session (for example, due to a trading halt into the close, system issue, or other disruption), the market will use the last valid on-exchange trade price of the regular session as the effective closing price.\n\nThe resolution source for this market is Yahoo Finance specifically, the NVIDIA (NVDA) \"Close\" prices available at https://finance.yahoo.com/quote/NVDA/history, published under \"Historical Prices.\"\n\nIn the event of a stock split, reverse stock split, or similar corporate action affecting the listed company during the listed time frame, this market will resolve based on split-adjusted prices as displayed on Yahoo Finance."
    },
    "timestamp": "1766790415550",
    "event_type": "new_market"
}
```


Listing A.7: Kalshi Documented Orderbook Delta


```
{
  "type": "orderbook_delta",
  "sid": 2,
  "seq": 3,
  "msg": {
    "market_ticker": "FED-23DEC-T3.00",
    "market_id": "9b0f6b43-5b68-4f9f-9f02-9a2d1b8ac1a1",
    "price": 96,
    "price_dollars": "0.960",
    "delta": -54,
    "delta_fp": "-54.00",
    "side": "yes",
    "ts": "2022-11-22T20:44:01Z"
  }
}
```

Listing A.8: Kalshi Documented Orderbook Snapshot


```
{
  "type": "orderbook_snapshot",
  "sid": 2,
  "seq": 2,
  "msg": {
    "market_ticker": "FED-23DEC-T3.00",
    "market_id": "9b0f6b43-5b68-4f9f-9f02-9a2d1b8ac1a1",
    "yes": [
      [
        8,
        300
      ],
      [
        22,
        333
      ]
    ],
    "yes_dollars": [
      [
        "0.080",
        300
      ],
      [
        "0.220",
        333
      ]
    ],
    "yes_dollars_fp": [
      [
        "0.0800",
        "300.00"
      ],
      [
        "0.2200",
        "333.00"
      ]
    ],
    "no": [
      [
        54,
        20
      ],
      [
        56,
        146
      ]
    ],
    "no_dollars": [
      [
        "0.540",
        20
      ],
```

```
      [
        "0.560",
        146
      ]
    ],
    "no_dollars_fp": [
      [
        "0.5400",
        "20.00"
      ],
      [
        "0.5600",
        "146.00"
      ]
    ]
  }
}
```

Listing A.9: Kalshi Documented Trade Update

```
{
  "type": "trade",
  "sid": 11,
  "msg": {
    "trade_id": "d91bc706-ee49-470d-82d8-11418bda6fed",
    "market_ticker": "HIGHNY-22DEC23-B53.5",
    "yes_price": 36,
    "yes_price_dollars": "0.360",
    "no_price": 64,
    "no_price_dollars": "0.640",
    "count": 136,
    "count_fp": "136.00",
    "taker_side": "no",
    "ts": 1669149841
  }
}
```

Listing A.10: Kalshi Documented Market Lifecycle V2

```
{
  "type": "market_lifecycle_v2",
  "sid": 13,
  "msg": {
    "market_ticker": "INXD-23SEP14-B4487",
    "event_type": "created",
    "open_ts": 1694635200,
    "close_ts": 1694721600,
    "additional_metadata": {
      "name": "S&P 500 daily return on Sep 14",
      "title": "S&P 500 closes up by 0.02% or more",
      "yes_sub_title": "S&P 500 closes up 0.02%+",
      "no_sub_title": "S&P 500 closes up <0.02%",
      "rules_primary": "The S&P 500 index level at 4:00 PM ET...",
      "rules_secondary": "",
```

```
      "can_close_early": true,
      "event_ticker": "INXD-23SEP14",
      "expected_expiration_ts": 1694721600,
      "strike_type": "greater",
      "floor_strike": 4487
    }
  }
}
```

Listing A.11: Kalshi Documented Event Object

```
{
  "event": {
    "event_ticker": "<string>",
    "series_ticker": "<string>",
    "sub_title": "<string>",
    "title": "<string>",
    "collateral_return_type": "<string>",
    "mutually_exclusive": true,
    "category": "<string>",
    "available_on_brokers": true,
    "product_metadata": {},
    "strike_date": "2023-11-07T05:31:56Z",
    "strike_period": "<string>",
    "markets": [
      {
        "ticker": "<string>",
        "event_ticker": "<string>",
        "market_type": "binary",
        "title": "<string>",
        "subtitle": "<string>",
        "yes_sub_title": "<string>",
        "no_sub_title": "<string>",
        "created_time": "2023-11-07T05:31:56Z",
        "updated_time": "2023-11-07T05:31:56Z",
        "open_time": "2023-11-07T05:31:56Z",
        "close_time": "2023-11-07T05:31:56Z",
        "expiration_time": "2023-11-07T05:31:56Z",
        "latest_expiration_time": "2023-11-07T05:31:56Z",
        "settlement_timer_seconds": 123,
        "status": "initialized",
        "response_price_units": "usd_cent",
        "yes_bid": 123,
        "yes_bid_dollars": "0.5600",
        "yes_ask": 123,
        "yes_ask_dollars": "0.5600",
        "no_bid": 123,
        "no_bid_dollars": "0.5600",
        "no_ask": 123,
        "no_ask_dollars": "0.5600",
        "last_price": 123,
        "last_price_dollars": "0.5600",
        "volume": 123,
        "volume_fp": "10.00",
```

```
        "volume_24h": 123,
        "volume_24h_fp": "10.00",
        "result": "yes",
        "can_close_early": true,
        "open_interest": 123,
        "open_interest_fp": "10.00",
        "notional_value": 123,
        "notional_value_dollars": "0.5600",
        "previous_yes_bid": 123,
        "previous_yes_bid_dollars": "0.5600",
        "previous_yes_ask": 123,
        "previous_yes_ask_dollars": "0.5600",
        "previous_price": 123,
        "previous_price_dollars": "0.5600",
        "liquidity": 123,
        "liquidity_dollars": "0.5600",
        "expiration_value": "<string>",
        "tick_size": 123,
        "rules_primary": "<string>",
        "rules_secondary": "<string>",
        "price_level_structure": "<string>",
        "price_ranges": [
          {
            "start": "<string>",
            "end": "<string>",
            "step": "<string>"
          }
        ],
        "expected_expiration_time": "2023-11-07T05:31:56Z",
        "settlement_value": 123,
        "settlement_value_dollars": "0.5600",
        "settlement_ts": "2023-11-07T05:31:56Z",
        "fee_waiver_expiration_time": "2023-11-07T05:31:56Z",
        "early_close_condition": "<string>",
        "strike_type": "greater",
        "floor_strike": 123,
        "cap_strike": 123,
        "functional_strike": "<string>",
        "custom_strike": {},
        "mve_collection_ticker": "<string>",
        "mve_selected_legs": [
          {
            "event_ticker": "<string>",
            "market_ticker": "<string>",
            "side": "<string>",
            "yes_settlement_value_dollars": "0.5600"
          }
        ],
        "primary_participant_key": "<string>",
        "is_provisional": true
      }
    ]
  },
  "markets": [
```

```
    {
      "ticker": "<string>",
      "event_ticker": "<string>",
      "market_type": "binary",
      "title": "<string>",
      "subtitle": "<string>",
      "yes_sub_title": "<string>",
      "no_sub_title": "<string>",
      "created_time": "2023-11-07T05:31:56Z",
      "updated_time": "2023-11-07T05:31:56Z",
      "open_time": "2023-11-07T05:31:56Z",
      "close_time": "2023-11-07T05:31:56Z",
      "expiration_time": "2023-11-07T05:31:56Z",
      "latest_expiration_time": "2023-11-07T05:31:56Z",
      "settlement_timer_seconds": 123,
      "status": "initialized",
      "response_price_units": "usd_cent",
      "yes_bid": 123,
      "yes_bid_dollars": "0.5600",
      "yes_ask": 123,
      "yes_ask_dollars": "0.5600",
      "no_bid": 123,
      "no_bid_dollars": "0.5600",
      "no_ask": 123,
      "no_ask_dollars": "0.5600",
      "last_price": 123,
      "last_price_dollars": "0.5600",
      "volume": 123,
      "volume_fp": "10.00",
      "volume_24h": 123,
      "volume_24h_fp": "10.00",
      "result": "yes",
      "can_close_early": true,
      "open_interest": 123,
      "open_interest_fp": "10.00",
      "notional_value": 123,
      "notional_value_dollars": "0.5600",
      "previous_yes_bid": 123,
      "previous_yes_bid_dollars": "0.5600",
      "previous_yes_ask": 123,
      "previous_yes_ask_dollars": "0.5600",
      "previous_price": 123,
      "previous_price_dollars": "0.5600",
      "liquidity": 123,
      "liquidity_dollars": "0.5600",
      "expiration_value": "<string>",
      "tick_size": 123,
      "rules_primary": "<string>",
      "rules_secondary": "<string>",
      "price_level_structure": "<string>",
      "price_ranges": [
        {
          "start": "<string>",
          "end": "<string>",
```

```
          "step": "<string>"
        }
      ],
      "expected_expiration_time": "2023-11-07T05:31:56Z",
      "settlement_value": 123,
      "settlement_value_dollars": "0.5600",
      "settlement_ts": "2023-11-07T05:31:56Z",
      "fee_waiver_expiration_time": "2023-11-07T05:31:56Z",
      "early_close_condition": "<string>",
      "strike_type": "greater",
      "floor_strike": 123,
      "cap_strike": 123,
      "functional_strike": "<string>",
      "custom_strike": {},
      "mve_collection_ticker": "<string>",
      "mve_selected_legs": [
        {
          "event_ticker": "<string>",
          "market_ticker": "<string>",
          "side": "<string>",
          "yes_settlement_value_dollars": "0.5600"
        }
      ],
      "primary_participant_key": "<string>",
      "is_provisional": true
    }
  ]
}
```

Appendix B

Smart Contracts

Table B.1: Contract Event Signatures

Contract / Event Signatures
CTFExchange
FeeCharged(receiver(address), tokenId(uint256), fee(uint256))
NewAdmin(admin(address), newAdmin(address))
NewOperator(operator(address), newOperator(address))
OrderCancelled(orderHash(bytes32))
OrderFilled(orderHash(bytes32), maker(address), taker(address), makerAssetId(uint256), takerAssetId(uint256), making(uint256), taking(uint256), fee(uint256))
OrdersMatched(orderHash(bytes32), maker(address), makerAssetId(uint256), takerAssetId(uint256), making(uint256), taking(uint256))
ProxyFactoryUpdated(oldProxyFactory(address), newProxyFactory(address))
RemovedAdmin(admin(address), oldAdmin(address))
RemovedOperator(operator(address), oldOperator(address))
SafeFactoryUpdated(oldSafeFactory(address), newSafeFactory(address))
TokenRegistered(token0(uint256), token1(uint256), conditionId(bytes32))
TradingPaused(trader(address))
TradingUnpaused(trader(address))
Conditional Tokens
ConditionPreparation(conditionId(bytes32), oracle(address), questionId(bytes32), outcomeSlotCount(uint256))

Table B.1 – continued from previous page

Contract / Event Signatures
ConditionResolution(conditionId(bytes32), oracle(address), questionId(bytes32), outcomeSlotCount(uint256), payoutNumerators(uint256[]))
PositionSplit(stakeholder(address), collateralToken(address), parentCollectionId(bytes32), conditionId(bytes32), partition(uint256[]), amount(uint256))
PositionsMerge(stakeholder(address), collateralToken(address), parentCollectionId(bytes32), conditionId(bytes32), partition(uint256[]), amount(uint256))
PayoutRedemption(redeemer(address), collateralToken(address), parentCollectionId(bytes32), conditionId(bytes32), indexSets(uint256[]), payout(uint256))
URI(value(string), id(uint256))
Neg Risk Operator
MarketPrepared(marketId(bytes32), feeBips(uint256), data(bytes))
NewAdmin(admin(address), newAdmin(address))
QuestionEmergencyResolved(questionId(bytes32), result(bool))
QuestionFlagged(questionId(bytes32))
QuestionPrepared(marketId(bytes32), questionId(bytes32), requestId(bytes32), questionIndex(uint256), data(bytes))
QuestionReported(questionId(bytes32), requestId(bytes32), result(bool))
QuestionResolved(questionId(bytes32), result(bool))
QuestionUnflagged(questionId(bytes32))
RemovedAdmin(admin(address), oldAdmin(address))
Neg Risk Adapter
MarketPrepared(marketId(bytes32), oracle(address), feeBips(uint256), data(bytes))
NewAdmin(admin(address), newAdmin(address))
OutcomeReported(marketId(bytes32), questionId(bytes32), outcome(bool))
PayoutRedemption(redeemer(address), conditionId(bytes32), amounts(uint256[]), payout(uint256))
PositionSplit(stakeholder(address), conditionId(bytes32), amount(uint256))
PositionsConverted(stakeholder(address), marketId(bytes32), indexSet(uint256), amount(uint256))

Table B.1 – continued from previous page

Contract / Event Signatures
PositionsMerge(stakeholder(address), conditionId(bytes32), amount(uint256))
QuestionPrepared(marketId(bytes32), questionId(bytes32), index(uint256), data(bytes))
RemovedAdmin(admin(address), oldAdmin(address))
UmaCtfadapter
AncillaryDataUpdated(questionId(bytes32), creator(address), ancillaryData(bytes))
NewAdmin(admin(address), newAdmin(address))
QuestionFlagged(questionId(bytes32))
QuestionInitialized(questionId(bytes32), timestamp(uint256), creator(address), ancillaryData(bytes), rewardToken(address), reward(uint256), proposalBond(uint256))
QuestionManuallyResolved(questionId(bytes32), payouts(uint256[]))
QuestionPaused(questionId(bytes32))
QuestionReset(questionId(bytes32))
QuestionResolved(questionId(bytes32), price(int256), payouts(uint256[]))
QuestionUnflagged(questionId(bytes32))
QuestionUnpaused(questionId(bytes32))
RemovedAdmin(admin(address), oldAdmin(address))
Managed Optimistic Oracle V2
MessageSent(data(bytes))
PriceRequestAdded(identifier(bytes32), timestamp(uint256), ancillaryData(bytes), childRequestId(bytes32))
PriceRequestBridged(requester(address), identifier(bytes32), time(uint256), ancillaryData(bytes), childRequestId(bytes32), parentRequestId(bytes32))
PushedPrice(identifier(bytes32), timestamp(uint256), ancillaryData(bytes), price(int256), childRequestId(bytes32))
ResolvedLegacyRequest(identifier(bytes32), timestamp(uint256), ancillaryData(bytes), price(int256), childRequestId(bytes32), parentRequestId(bytes32))
AllowedBondRangeUpdated(proposer(address), minBond(uint256), maxBond(uint256))
CustomBondSet(identifier(bytes32), proposer(address), ancillaryData(bytes32), data(bytes), currency(address), bond(uint256))

Table B.1 – continued from previous page

Contract / Event Signatures
CustomLivenessSet(identifier(bytes32), proposer(address), ancillaryData(bytes32), data(bytes), liveness(uint256))
CustomProposerWhitelistSet(identifier(bytes32), proposer(address), ancillaryData(bytes32), data(bytes), newProposer(address))
DefaultAdminDelayChangeCanceled()
DefaultAdminDelayChangeScheduled(delay(uint48), newDelay(uint48))
DefaultAdminTransferCanceled()
DefaultAdminTransferScheduled(newAdmin(address), time(uint48))
DefaultProposerWhitelistUpdated(newWhitelist(address))
DisputePrice(requester(address), proposer(address), disputer(address), identifier(bytes32), timestamp(uint256), ancillaryData(bytes), proposedPrice(int256))
Initialized(version(uint64))
MinimumLivenessUpdated(minimumLiveness(uint256))
ProposePrice(requester(address), proposer(address), identifier(bytes32), timestamp(uint256), ancillaryData(bytes), proposedPrice(int256), expirationTime(uint256), currency(address))
RequestManagerAdded(requestManager(address))
RequestManagerRemoved(requestManager(address))
RequestPrice(requester(address), identifier(bytes32), timestamp(uint256), ancillaryData(bytes), currency(address), reward(uint256), finalFee(uint256))
RequesterWhitelistUpdated(newWhitelist(address))
RoleAdminChanged(role(bytes32), previousAdminRole(bytes32), newAdminRole(bytes32))
RoleGranted(role(bytes32), account(address), adminRole(bytes32))
RoleRevoked(role(bytes32), account(address), adminRole(bytes32))
Settle(requester(address), proposer(address), disputer(address), identifier(bytes32), timestamp(uint256), ancillaryData(bytes), resolvedPrice(int256), payout(uint256))
Upgraded(implementation(address))
Oracle Child Tunnel
PriceRequestBridged(requester(address), identifier(bytes32), time(uint256), ancillaryData(bytes), childRequestId(bytes32), parentRequestId(bytes32))
ResolvedLegacyRequest(identifier(bytes32), timestamp(uint256), ancillaryData(bytes), price(int256), childRequestId(bytes32), parentRequestId(bytes32))

Table B.1 – continued from previous page

Contract / Event Signatures
`PriceRequestAdded(identifier(bytes32), timestamp(uint256), ancillaryData(bytes), childRequestId(bytes32))`
`PushedPrice(identifier(bytes32), timestamp(uint256), ancillaryData(bytes), price(int256), childRequestId(bytes32))`
`MessageSent(data(bytes))`
FxTunnel
`NewFxMessage(rootMessageSender(address), receiver(address), data(bytes))`
Optimistic Oracle V2
`RequestPrice(requester(address), identifier(bytes32), timestamp(uint256), ancillaryData(bytes), currency(address), reward(uint256), finalFee(uint256))`
`ProposePrice(requester(address), proposer(address), identifier(bytes32), timestamp(uint256), ancillaryData(bytes), proposedPrice(int256), expirationTime(uint256), currency(address))`
`DisputePrice(requester(address), proposer(address), disputer(address), identifier(bytes32), timestamp(uint256), ancillaryData(bytes), proposedPrice(int256))`
`Settle(requester(address), proposer(address), disputer(address), identifier(bytes32), timestamp(uint256), ancillaryData(bytes), resolvedPrice(int256), payout(uint256))`

Table B.2: Contract Aliases, Deployment Addresses, and Event Tables

Alias	**Deployment Address**	**Contract Table Name**
CTFE	0x4bfb41d5b3570 defd03c39a9a4d8 de6bd8b8982e	CTFExchange
Negrisk CTFE	0xC5d563A36AE 78145C45a50134d 48A1215220f80a	CTFExchange
CTF	0x4d97dcd97ec94 5f40cf65f87097ace 5ea0476045	(Gnosis) Conditional Tokens
Negrisk Operator	0x71523d0f655b4 1e805cec45b17163 f528b59b820	Negrisk Operator
Negrisk Adapter	0xd91e80cf2e7be2 e162c6513ced06f1 dd0da35296	Negrisk Adapter
Negrisk UmaCtfAdapter	0x2F5e3684cb1F3 18ec51b00Edba38 d79Ac2c0aA9d	UmaCtfAdapter
OOV2 UmaCtfAdapter	0x6A9D222616C9 0FcA5754cd1333c FD9b7fb6a4F74	UmaCtfAdapter
MOOV2 Adapter	0x65070BE91477 460D8A7AeEb94 ef92fe056C2f2A7	UmaCtfAdapter
MOOV2	0x2C0367a9DB23 1dDeBd88a94b4f 6461a6e47C58B1	(UMA) Managed Optimistic Oracle V2
Oracle Child Tunnel	0xac60353a54873 c446101216829a6 A98cDbbC3f3D	Oracle Child Tunnel
FxTunnel	0x8397259c98375 1DAf40400790063 935a11afa28a	FxTunnel
OOV2	0xeE3Afe347D5C 74317041E2618C 49534dAf887c24	(UMA) Optimistic Oracle V2

Appendix C

Polymarket Event Object JSON Tree Analysis

Table C.1: JSON Tree Structure Analysis of `/events` Endpoint Responses. For each entry, the Path and Total Count are on the first line, with Presence and Type Makeup on the indented second line.

Path	**Total Count**
`ROOT` *Presence: 100.0%* *Type: Object (100.0)*	126558
`ROOT.tags` *Presence: 100.0%* *Type: Object (100.0)*	126555
`ROOT.tags.[]` *Presence: 100.0%* *Type: Array (100.0)*	126555
`ROOT.markets` *Presence: 100.0%* *Type: Object (100.0)*	126523
`ROOT.markets.[]` *Presence: 100.0%* *Type: Array (100.0)*	126523
`ROOT.series` *Presence: 83.6%* *Type: Object (100.0)*	105820
`ROOT.series.[]` *Presence: 83.6%* *Type: Array (100.0)*	105820
`ROOT.markets.[].clobRewards` *Presence: 7.7%* *Type: Object (100.0)*	9805
`ROOT.markets.[].clobRewards.[]` *Presence: 7.7%* *Type: Array (100.0)*	9805

Table C.1 – continued from previous page

Path	Total Count
`ROOT.eventCreators` *Presence: 0.1% Type: Object (100.0)*	87
`ROOT.eventCreators.[]` *Presence: 0.1% Type: Array (100.0)*	87
`ROOT.tags.[].slug` *Presence: 100.0% Type: str:general (100.0), str:integer (0.0)*	740323
`ROOT.tags.[].requiresTranslation` *Presence: 100.0% Type: bool (100.0)*	740323
`ROOT.tags.[].label` *Presence: 100.0% Type: str:general (100.0), str:integer (0.0)*	740323
`ROOT.tags.[].id` *Presence: 100.0% Type: str:integer (100.0)*	740323
`ROOT.tags.[].updatedAt` *Presence: 100.0% Type: str:timestamp (100.0)*	740162
`ROOT.tags.[].createdAt` *Presence: 98.8% Type: str:timestamp (100.0)*	731578
`ROOT.tags.[].forceShow` *Presence: 88.6% Type: bool (100.0)*	655610
`ROOT.tags.[].publishedAt` *Presence: 40.7% Type: str:general (100.0)*	301036
`ROOT.markets.[].approved` *Presence: 100.0% Type: bool (100.0)*	294240
`ROOT.markets.[].active` *Presence: 100.0% Type: bool (100.0)*	294240
`ROOT.markets.[].clearBookOnStart` *Presence: 100.0% Type: bool (100.0)*	294240
`ROOT.markets.[].createdAt` *Presence: 100.0% Type: str:timestamp (100.0)*	294240
`ROOT.markets.[].cyom` *Presence: 100.0% Type: bool (100.0)*	294240
`ROOT.markets.[].feesEnabled` *Presence: 100.0% Type: bool (100.0)*	294240
`ROOT.markets.[].umaResolutionStatuses` *Presence: 100.0% Type: str:json_encoded (100.0)*	294240

Table C.1 – continued from previous page

Path	Total Count
`ROOT.markets.[].deploying` *Presence: 100.0% Type: bool (100.0)*	294240
`ROOT.markets.[].description` *Presence: 100.0% Type: str:general (100.0)*	294240
`ROOT.markets.[].funded` *Presence: 100.0% Type: bool (100.0)*	294240
`ROOT.markets.[].archived` *Presence: 100.0% Type: bool (100.0)*	294240
`ROOT.markets.[].bestAsk` *Presence: 100.0% Type: float (59.7), int (40.3)*	294240
`ROOT.markets.[].manualActivation` *Presence: 100.0% Type: bool (100.0)*	294240
`ROOT.markets.[].id` *Presence: 100.0% Type: str:integer (100.0)*	294240
`ROOT.markets.[].holdingRewardsEnabled` *Presence: 100.0% Type: bool (100.0)*	294240
`ROOT.markets.[].pagerDutyNotificationEnabled` *Presence: 100.0% Type: bool (100.0)*	294240
`ROOT.markets.[].pendingDeployment` *Presence: 100.0% Type: bool (100.0)*	294240
`ROOT.markets.[].question` *Presence: 100.0% Type: str:general (100.0)*	294240
`ROOT.markets.[].ready` *Presence: 100.0% Type: bool (100.0)*	294240
`ROOT.markets.[].rewardsMaxSpread` *Presence: 100.0% Type: int (82.9), float (17.1)*	294240
`ROOT.markets.[].outcomes` *Presence: 100.0% Type: str:json_ encoded (100.0)*	294240
`ROOT.markets.[].rfqEnabled` *Presence: 100.0% Type: bool (100.0)*	294240
`ROOT.markets.[].rewardsMinSize` *Presence: 100.0% Type: int (100.0), float (0.0)*	294240
`ROOT.markets.[].spread` *Presence: 100.0% Type: float (83.4), int (16.6)*	294240

Table C.1 – continued from previous page

Path	Total Count
`ROOT.markets.[].requiresTranslation` *Presence: 100.0% Type: bool (100.0)*	294240
`ROOT.markets.[].restricted` *Presence: 100.0% Type: bool (100.0)*	294240
`ROOT.markets.[].slug` *Presence: 100.0% Type: str:general (100.0)*	294240
`ROOT.markets.[].conditionId` *Presence: 100.0% Type: str:hexadecimal (99.97), str:empty (0.03)*	294240
`ROOT.markets.[].closed` *Presence: 100.0% Type: bool (100.0)*	294240
`ROOT.markets.[].negRiskOther` *Presence: 100.0% Type: bool (100.0)*	294240
`ROOT.markets.[].marketMakerAddress` *Presence: 100.0% Type: str:empty (91.1), str:hexadecimal (8.9)*	294240
`ROOT.markets.[].new` *Presence: 100.0% Type: bool (100.0)*	294209
`ROOT.markets.[].clobTokenIds` *Presence: 99.9% Type: str:json_ encoded (100.0)*	294080
`ROOT.markets.[].image` *Presence: 99.8% Type: str:general (99.6), str:empty (0.4)*	293596
`ROOT.markets.[].icon` *Presence: 99.7% Type: str:general (99.6), str:empty (0.4)*	293371
`ROOT.markets.[].updatedAt` *Presence: 99.6% Type: str:timestamp (100.0)*	293018
`ROOT.markets.[].hasReviewedDates` *Presence: 99.3% Type: bool (100.0)*	292264
`ROOT.markets.[].questionID` *Presence: 99.3% Type: str:hexadecimal (100.0)*	292192
`ROOT.markets.[].enableOrderBook` *Presence: 99.1% Type: bool (100.0)*	291620
`ROOT.markets.[].endDate` *Presence: 99.0% Type: str:timestamp (100.0)*	291388
`ROOT.markets.[].startDate` *Presence: 98.7% Type: str:timestamp (100.0)*	290272

Table C.1 – continued from previous page

Path	Total Count
`ROOT.markets.[].orderMinSize` *Presence: 98.4%* *Type: int (100.0), float (0.0)*	289522
`ROOT.markets.[].orderPriceMinTickSize` *Presence: 98.4%* *Type: float (100.0), int (0.0)*	289520
`ROOT.markets.[].endDateIso` *Presence: 98.2%* *Type: str:timestamp (100.0)*	288839
`ROOT.markets.[].acceptingOrders` *Presence: 97.0%* *Type: bool (100.0)*	285530
`ROOT.markets.[].negRisk` *Presence: 96.8%* *Type: bool (100.0)*	284707
`ROOT.markets.[].outcomePrices` *Presence: 95.5%* *Type: str:json_encoded (100.0)*	280947
`ROOT.markets.[].automaticallyActive` *Presence: 95.4%* *Type: bool (100.0)*	280826
`ROOT.markets.[].featured` *Presence: 94.7%* *Type: bool (100.0)*	278588
`ROOT.markets.[].startDateIso` *Presence: 91.5%* *Type: str:timestamp (100.0)*	269084
`ROOT.markets.[].volume` *Presence: 91.4%* *Type: str:float (77.1), str:integer (22.9)*	268901
`ROOT.markets.[].volumeNum` *Presence: 91.4%* *Type: float (77.0), int (23.0)*	268901
`ROOT.markets.[].acceptingOrdersTimestamp` *Presence: 90.4%* *Type: str:timestamp (100.0)*	266051
`ROOT.markets.[].volumeClob` *Presence: 89.8%* *Type: float (76.6), int (23.4)*	264197
`ROOT.markets.[].groupItemThreshold` *Presence: 89.6%* *Type: str:integer (100.0)*	263602
`ROOT.markets.[].closedTime` *Presence: 88.3%* *Type: str:general (100.0)*	259848
`ROOT.markets.[].lastTradePrice` *Presence: 88.0%* *Type: int (78.0), float (22.0)*	258812
`ROOT.markets.[].umaResolutionStatus` *Presence: 87.6%* *Type: str:general (100.0)*	257686

Table C.1 – continued from previous page

Path	Total Count
`ROOT.markets.[].umaEndDate` *Presence: 87.4%* *Type: str:timestamp (96.9), str:general (3.1)*	257309
`ROOT.markets.[].automaticallyResolved` *Presence: 84.7%* *Type: bool (100.0)*	249301
`ROOT.tags.[].updatedBy` *Presence: 32.7%* *Type: int (100.0)*	241730
`ROOT.markets.[].resolvedBy` *Presence: 81.8%* *Type: str:hexadecimal (100.0)*	240654
`ROOT.markets.[].resolutionSource` *Presence: 80.4%* *Type: str:general (74.5), str:empty (25.5)*	236453
`ROOT.tags.[].isCarousel` *Presence: 31.2%* *Type: bool (100.0)*	231335
`ROOT.markets.[].volume1yrClob` *Presence: 77.1%* *Type: float (61.6), int (38.4)*	226919
`ROOT.markets.[].volume1yr` *Presence: 77.1%* *Type: float (61.6), int (38.4)*	226919
`ROOT.markets.[].volume1moClob` *Presence: 77.0%* *Type: float (61.5), int (38.5)*	226441
`ROOT.markets.[].volume1mo` *Presence: 77.0%* *Type: float (61.5), int (38.5)*	226441
`ROOT.markets.[].submitted_by` *Presence: 76.7%* *Type: str:hexadecimal (99.7), others (0.3)*	225800
`ROOT.markets.[].volume1wkClob` *Presence: 76.0%* *Type: float (60.6), int (39.4)*	223602
`ROOT.markets.[].volume1wk` *Presence: 76.0%* *Type: float (60.6), int (39.4)*	223602
`ROOT.markets.[].umaBond` *Presence: 74.5%* *Type: str:integer (98.7), str:float (1.3)*	219274
`ROOT.markets.[].umaReward` *Presence: 74.5%* *Type: str:integer (98.6), str:float (1.4)*	219274
`ROOT.markets.[].negRiskRequestID` *Presence: 70.9%* *Type: str:empty (54.9), str:hexadecimal (45.1)*	208566
`ROOT.markets.[].groupItemTitle` *Presence: 69.9%* *Type: str:general (91.6), others (8.4)*	205672

Table C.1 – continued from previous page

Path	Total Count
`ROOT.markets.[].oneDayPriceChange` *Presence: 68.4%* *Type: float (75.7), int (24.3)*	201115
`ROOT.markets.[].deployingTimestamp` *Presence: 64.6%* *Type: str:timestamp (100.0)*	189939
`ROOT.markets.[].customLiveness` *Presence: 61.3%* *Type: int (100.0)*	180461
`ROOT.markets.[].bestBid` *Presence: 58.7%* *Type: float (71.6), int (28.4)*	172631
`ROOT.markets.[].oneHourPriceChange` *Presence: 53.9%* *Type: float (52.6), int (47.4)*	158604
`ROOT.markets.[].competitive` *Presence: 47.1%* *Type: int (85.3), float (14.7)*	138520
`ROOT.markets.[].liquidity` *Presence: 47.1%* *Type: str:integer (80.6), str:float (19.4)*	138485
`ROOT.markets.[].liquidityNum` *Presence: 47.1%* *Type: int (82.3), float (17.7)*	138485
`ROOT.markets.[].showGmpOutcome` *Presence: 46.8%* *Type: bool (100.0)*	137594
`ROOT.markets.[].showGmpSeries` *Presence: 46.8%* *Type: bool (100.0)*	137593
`ROOT.markets.[].liquidityClob` *Presence: 45.4%* *Type: int (84.3), float (15.7)*	133631
`ROOT.markets.[].oneWeekPriceChange` *Presence: 42.4%* *Type: int (60.2), float (39.8)*	124819
`ROOT.active` *Presence: 100.0%* *Type: bool (100.0)*	126558
`ROOT.title` *Presence: 100.0%* *Type: str:general (100.0)*	126558
`ROOT.pendingDeployment` *Presence: 100.0%* *Type: bool (100.0)*	126558
`ROOT.requiresTranslation` *Presence: 100.0%* *Type: bool (100.0)*	126558
`ROOT.negRiskAugmented` *Presence: 100.0%* *Type: bool (100.0)*	126558

Table C.1 – continued from previous page

Path	Total Count
`ROOT.image` *Presence: 100.0%* *Type: str:general (99.8), str:empty (0.2)*	126558
`ROOT.icon` *Presence: 100.0%* *Type: str:general (99.8), str:empty (0.2)*	126558
`ROOT.id` *Presence: 100.0%* *Type: str:integer (100.0)*	126558
`ROOT.deploying` *Presence: 100.0%* *Type: bool (100.0)*	126558
`ROOT.closed` *Presence: 100.0%* *Type: bool (100.0)*	126558
`ROOT.cyom` *Presence: 100.0%* *Type: bool (100.0)*	126558
`ROOT.createdAt` *Presence: 100.0%* *Type: str:timestamp (100.0)*	126558
`ROOT.enableNegRisk` *Presence: 100.0%* *Type: bool (100.0)*	126558
`ROOT.showAllOutcomes` *Presence: 100.0%* *Type: bool (100.0)*	126558
`ROOT.showMarketImages` *Presence: 100.0%* *Type: bool (100.0)*	126558
`ROOT.slug` *Presence: 100.0%* *Type: str:general (100.0)*	126558
`ROOT.ticker` *Presence: 100.0%* *Type: str:general (100.0)*	126556
`ROOT.archived` *Presence: 100.0%* *Type: bool (100.0)*	126550
`ROOT.restricted` *Presence: 100.0%* *Type: bool (100.0)*	126550
`ROOT.featured` *Presence: 100.0%* *Type: bool (100.0)*	126534
`ROOT.startDate` *Presence: 100.0%* *Type: str:timestamp (100.0)*	126529
`ROOT.new` *Presence: 100.0%* *Type: bool (100.0)*	126520

Table C.1 – continued from previous page

Path	Total Count
`ROOT.updatedAt` *Presence: 100.0%* *Type: str:timestamp (100.0)*	126495
`ROOT.creationDate` *Presence: 99.9%* *Type: str:timestamp (100.0)*	126461
`ROOT.endDate` *Presence: 99.9%* *Type: str:timestamp (100.0)*	126388
`ROOT.description` *Presence: 99.5%* *Type: str:general (100.0), others (0.0)*	125876
`ROOT.commentCount` *Presence: 99.3%* *Type: int (100.0)*	125633
`ROOT.openInterest` *Presence: 98.7%* *Type: int (100.0)*	124935
`ROOT.enableOrderBook` *Presence: 97.5%* *Type: bool (100.0)*	123449
`ROOT.closedTime` *Presence: 94.8%* *Type: str:timestamp (100.0)*	119928
`ROOT.automaticallyResolved` *Presence: 90.5%* *Type: bool (100.0)*	114539
`ROOT.negRisk` *Presence: 89.2%* *Type: bool (100.0)*	112947
`ROOT.resolutionSource` *Presence: 88.3%* *Type: str:general (87.2), str:empty (12.8)*	111781
`ROOT.seriesSlug` *Presence: 83.7%* *Type: str:general (100.0)*	105868
`ROOT.series.[].id` *Presence: 100.0%* *Type: str:integer (100.0)*	105820
`ROOT.series.[].closed` *Presence: 100.0%* *Type: bool (100.0)*	105820
`ROOT.series.[].createdAt` *Presence: 100.0%* *Type: str:timestamp (100.0)*	105820
`ROOT.series.[].requiresTranslation` *Presence: 100.0%* *Type: bool (100.0)*	105820
`ROOT.series.[].commentCount` *Presence: 100.0%* *Type: int (100.0)*	105820

Table C.1 – continued from previous page

Path	Total Count
`ROOT.series.[].active` *Presence: 100.0% Type: bool (100.0)*	105820
`ROOT.series.[].archived` *Presence: 100.0% Type: bool (100.0)*	105820
`ROOT.series.[].ticker` *Presence: 100.0% Type: str:general (100.0)*	105820
`ROOT.series.[].slug` *Presence: 100.0% Type: str:general (100.0)*	105820
`ROOT.series.[].title` *Presence: 100.0% Type: str:general (100.0)*	105820
`ROOT.series.[].updatedAt` *Presence: 100.0% Type: str:timestamp (100.0)*	105820
`ROOT.series.[].seriesType` *Presence: 99.9% Type: str:general (100.0)*	105750
`ROOT.series.[].recurrence` *Presence: 99.9% Type: str:general (99.9), str:empty (0.1)*	105750
`ROOT.volume` *Presence: 82.8% Type: float (95.9), int (4.1)*	104840
`ROOT.automaticallyActive` *Presence: 81.5% Type: bool (100.0)*	103195
`ROOT.markets.[].secondsDelay` *Presence: 34.2% Type: int (100.0)*	100673
`ROOT.markets.[].gameStartTime` *Presence: 33.4% Type: str:general (100.0)*	98324
`ROOT.markets.[].negRiskMarketID` *Presence: 32.0% Type: str:hexadecimal (100.0)*	94024
`ROOT.markets.[].oneMonthPriceChange` *Presence: 30.5% Type: int (83.6), float (16.4)*	89813
`ROOT.startTime` *Presence: 69.7% Type: str:timestamp (100.0)*	88272
`ROOT.series.[].featured` *Presence: 79.2% Type: bool (100.0)*	83843
`ROOT.series.[].restricted` *Presence: 78.7% Type: bool (100.0)*	83242

Table C.1 – continued from previous page

Path	Total Count
`ROOT.series.[].image` *Presence: 76.6%* *Type: str:empty (58.4), str:general (41.6)*	81104
`ROOT.series.[].icon` *Presence: 76.6%* *Type: str:empty (58.4), str:general (41.6)*	81104
`ROOT.markets.[].eventStartTime` *Presence: 26.5%* *Type: str:timestamp (100.0)*	78103
`ROOT.tags.[].createdBy` *Presence: 10.4%* *Type: int (100.0)*	76858
`ROOT.series.[].liquidity` *Presence: 71.8%* *Type: float (100.0)*	75989
`ROOT.series.[].volume` *Presence: 71.6%* *Type: float (98.6), int (1.4)*	75775
`ROOT.markets.[].oneYearPriceChange` *Presence: 25.6%* *Type: int (99.9), float (0.1)*	75186
`ROOT.markets.[].volume1wkAmm` *Presence: 25.5%* *Type: int (100.0)*	75118
`ROOT.markets.[].volume1yrAmm` *Presence: 25.5%* *Type: int (100.0)*	75118
`ROOT.markets.[].volume1moAmm` *Presence: 25.5%* *Type: int (100.0)*	75118
`ROOT.volume1yr` *Presence: 58.7%* *Type: float (77.3), int (22.7)*	74284
`ROOT.volume1mo` *Presence: 58.7%* *Type: float (77.3), int (22.7)*	74274
`ROOT.volume1wk` *Presence: 58.5%* *Type: float (77.1), int (22.9)*	74043
`ROOT.markets.[].sportsMarketType` *Presence: 22.9%* *Type: str:general (100.0)*	67474
`ROOT.tags.[].forceHide` *Presence: 8.2%* *Type: bool (100.0)*	60671
`ROOT.competitive` *Presence: 47.9%* *Type: int (92.0), float (8.0)*	60616
`ROOT.liquidity` *Presence: 47.8%* *Type: int (91.0), float (9.0)*	60435

Table C.1 – continued from previous page

Path	Total Count
`ROOT.liquidityClob` *Presence: 47.7% Type: int (91.1), float (8.9)*	60431
`ROOT.markets.[].volume24hr` *Presence: 18.9% Type: int (85.6), float (14.4)*	55570
`ROOT.liquidityAmm` *Presence: 43.5% Type: int (99.9), float (0.1)*	55028
`ROOT.markets.[].volume24hrClob` *Presence: 17.3% Type: int (84.3), float (15.7)*	50866
`ROOT.markets.[].line` *Presence: 14.8% Type: float (98.3), int (1.7)*	43542
`ROOT.markets.[].seriesColor` *Presence: 14.1% Type: str:empty (99.0), str:general (1.0)*	41612
`ROOT.markets.[].volumeAmm` *Presence: 14.1% Type: int (98.7), float (1.3)*	41454
`ROOT.markets.[].volume24hrAmm` *Presence: 14.0% Type: int (100.0)*	41306
`ROOT.markets.[].clobRewards.[].rewardsAmount` *Presence: 100.0% Type: int (100.0)*	32054
`ROOT.markets.[].clobRewards.[].id` *Presence: 100.0% Type: str:integer (100.0)*	32054
`ROOT.markets.[].clobRewards.[].endDate` *Presence: 100.0% Type: str:timestamp (100.0)*	32054
`ROOT.markets.[].clobRewards.[].conditionId` *Presence: 100.0% Type: str:hexadecimal (100.0)*	32054
`ROOT.markets.[].clobRewards.[].assetAddress` *Presence: 100.0% Type: str:hexadecimal (100.0)*	32054
`ROOT.markets.[].clobRewards.[].startDate` *Presence: 100.0% Type: str:timestamp (100.0)*	32054
`ROOT.markets.[].clobRewards.[].rewardsDailyRate` *Presence: 100.0% Type: int (100.0), float (0.0)*	32054
`ROOT.eventDate` *Presence: 21.4% Type: str:timestamp (100.0)*	27116
`ROOT.markets.[].fpmmLive` *Presence: 9.0% Type: bool (100.0)*	26535

Table C.1 – continued from previous page

Path	Total Count
`ROOT.markets.[].wideFormat` *Presence: 8.8%* *Type: bool (100.0)*	25839
`ROOT.markets.[].fee` *Presence: 8.8%* *Type: str:integer (100.0), str:float (0.0)*	25829
`ROOT.eventWeek` *Presence: 18.6%* *Type: int (100.0)*	23516
`ROOT.markets.[].readyForCron` *Presence: 7.8%* *Type: bool (100.0)*	22819
`ROOT.period` *Presence: 16.6%* *Type: str:general (94.1), str:empty (5.9)*	20964
`ROOT.markets.[].sentDiscord` *Presence: 7.0%* *Type: bool (100.0)*	20529
`ROOT.ended` *Presence: 16.1%* *Type: bool (100.0)*	20316
`ROOT.live` *Presence: 16.0%* *Type: bool (100.0)*	20290
`ROOT.score` *Presence: 15.1%* *Type: str:general (99.6), str:empty (0.4)*	19084
`ROOT.markets.[].notificationsEnabled` *Presence: 6.2%* *Type: bool (100.0)*	18372
`ROOT.sortBy` *Presence: 14.4%* *Type: str:general (100.0)*	18208
`ROOT.gameId` *Presence: 13.5%* *Type: int (100.0)*	17121
`ROOT.elapsed` *Presence: 13.4%* *Type: str:empty (96.5), others (3.5)*	16898
`ROOT.finishedTimestamp` *Presence: 10.9%* *Type: str:timestamp (100.0)*	13752
`ROOT.negRiskMarketID` *Presence: 9.2%* *Type: str:hexadecimal (99.9), str:empty (0.1)*	11614
`ROOT.series.[].new` *Presence: 10.1%* *Type: bool (100.0)*	10705
`ROOT.markets.[].gameId` *Presence: 3.5%* *Type: str:hexadecimal (55.1), str:integer (44.9)*	10354

Table C.1 – continued from previous page

Path	Total Count
`ROOT.markets.[].marketType` *Presence: 3.4% Type: str:general (100.0)*	10072
`ROOT.markets.[].creator` *Presence: 3.4% Type: str:empty (100.0)*	10072
`ROOT.deployingTimestamp` *Presence: 6.7% Type: str:timestamp (100.0)*	8511
`ROOT.volume24hr` *Presence: 6.2% Type: int (65.4), float (34.6)*	7886
`ROOT.series.[].layout` *Presence: 7.4% Type: str:general (100.0)*	7868
`ROOT.series.[].volume24hr` *Presence: 7.4% Type: int (100.0)*	7816
`ROOT.series.[].publishedAt` *Presence: 7.4% Type: str:general (100.0)*	7816
`ROOT.series.[].competitive` *Presence: 7.4% Type: str:integer (100.0)*	7816
`ROOT.series.[].commentsEnabled` *Presence: 7.4% Type: bool (100.0)*	7816
`ROOT.series.[].updatedBy` *Presence: 7.3% Type: str:integer (100.0)*	7710
`ROOT.series.[].createdBy` *Presence: 7.3% Type: str:integer (100.0)*	7710
`ROOT.series.[].startDate` *Presence: 6.6% Type: str:timestamp (100.0)*	7013
`ROOT.gmpChartMode` *Presence: 5.4% Type: str:general (100.0)*	6866
`ROOT.homeTeamName` *Presence: 4.6% Type: str:general (100.0)*	5770
`ROOT.awayTeamName` *Presence: 4.6% Type: str:general (100.0)*	5770
`ROOT.markets.[].commentsEnabled` *Presence: 2.0% Type: bool (100.0)*	5772
`ROOT.markets.[].updatedBy` *Presence: 1.9% Type: int (100.0)*	5544

Table C.1 – continued from previous page

Path	Total Count
`ROOT.markets.[].umaEndDateIso` *Presence: 1.7%* *Type: str:timestamp (100.0)*	5124
`ROOT.published_at` *Presence: 4.0%* *Type: str:general (100.0)*	5089
`ROOT.markets.[].category` *Presence: 1.5%* *Type: str:general (99.9), str:empty (0.1)*	4353
`ROOT.markets.[].takerBaseFee` *Presence: 1.1%* *Type: int (100.0)*	3313
`ROOT.markets.[].makerBaseFee` *Presence: 1.1%* *Type: int (100.0)*	3312
`ROOT.markets.[].twitterCardLocation` *Presence: 1.0%* *Type: str:general (100.0)*	3026
`ROOT.markets.[].twitterCardLastRefreshed` *Presence: 1.0%* *Type: str:integer (100.0)*	3003
`ROOT.category` *Presence: 2.2%* *Type: str:general (100.0)*	2837
`ROOT.markets.[].mailchimpTag` *Presence: 0.9%* *Type: str:integer (90.0), str:null_like (10.0)*	2668
`ROOT.commentsEnabled` *Presence: 1.8%* *Type: bool (100.0)*	2246
`ROOT.markets.[].marketGroup` *Presence: 0.7%* *Type: int (100.0)*	2092
`ROOT.parentEventId` *Presence: 1.3%* *Type: int (100.0)*	1603
`ROOT.markets.[].twitterCardLastValidated` *Presence: 0.5%* *Type: str:float (100.0)*	1435
`ROOT.negRiskFeeBips` *Presence: 0.8%* *Type: int (100.0)*	999
`ROOT.series.[].description` *Presence: 0.6%* *Type: str:general (100.0)*	628
`ROOT.updatedBy` *Presence: 0.4%* *Type: str:integer (100.0)*	530
`ROOT.featuredImage` *Presence: 0.4%* *Type: str:general (66.2), str:empty (33.8)*	518

Table C.1 – continued from previous page

Path	Total Count
`ROOT.series.[].cgAssetName` *Presence: 0.5% Type: str:general (100.0)*	512
`ROOT.series.[].pythTokenID` *Presence: 0.5% Type: str:hexadecimal (100.0)*	512
`ROOT.series.[].subtitle` *Presence: 0.5% Type: str:general (100.0)*	512
`ROOT.featuredOrder` *Presence: 0.3% Type: int (100.0)*	373
`ROOT.markets.[].categoryMailchimpTag` *Presence: 0.1% Type: str:null_like (83.1), str:integer (16.9)*	308
`ROOT.createdBy` *Presence: 0.1% Type: str:integer (100.0)*	182
`ROOT.countryName` *Presence: 0.1% Type: str:general (99.3), str:empty (0.7)*	140
`ROOT.electionType` *Presence: 0.1% Type: str:general (100.0)*	137
`ROOT.tweetCount` *Presence: 0.1% Type: int (100.0)*	134
`ROOT.totalsMainLine` *Presence: 0.1% Type: float (100.0)*	99
`ROOT.spreadsMainLine` *Presence: 0.1% Type: float (100.0)*	99
`ROOT.eventCreators.[].id` *Presence: 100.0% Type: str:integer (100.0)*	87
`ROOT.eventCreators.[].creatorImage` *Presence: 100.0% Type: str:general (100.0)*	87
`ROOT.eventCreators.[].creatorHandle` *Presence: 100.0% Type: str:general (100.0)*	87
`ROOT.eventCreators.[].creatorUrl` *Presence: 100.0% Type: str:general (100.0)*	87
`ROOT.eventCreators.[].creatorName` *Presence: 100.0% Type: str:general (100.0)*	87
`ROOT.eventCreators.[].createdAt` *Presence: 100.0% Type: str:timestamp (100.0)*	87

Table C.1 – continued from previous page

Path	Total Count
`ROOT.markets.[].subcategory`	74
Presence: 0.0% *Type: str:general (89.2), str:empty (10.8)*	
`ROOT.markets.[].lowerBound`	73
Presence: 0.0% *Type: str:integer (78.1), str:float (19.2), others (2.7)*	
`ROOT.markets.[].upperBound`	72
Presence: 0.0% *Type: str:integer (79.2), str:float (19.4), others (1.4)*	
`ROOT.markets.[].disqusThread`	68
Presence: 0.0% *Type: str:general (100.0)*	
`ROOT.subcategory`	68
Presence: 0.1% *Type: str:general (98.5), str:empty (1.5)*	
`ROOT.markets.[].formatType`	66
Presence: 0.0% *Type: str:general (100.0)*	
`ROOT.markets.[].twitterCardImage`	62
Presence: 0.0% *Type: str:general (100.0)*	
`ROOT.sportsradarMatchId`	47
Presence: 0.0% *Type: str:general (100.0)*	
`ROOT.turnProviderId`	35
Presence: 0.0% *Type: str:integer (100.0)*	
`ROOT.disqusThread`	28
Presence: 0.0% *Type: str:general (100.0)*	
`ROOT.gameStatus`	23
Presence: 0.0% *Type: str:general (100.0)*	
`ROOT.markets.[].groupItemRange`	22
Presence: 0.0% *Type: str:json_encoded (100.0)*	
`ROOT.color`	21
Presence: 0.0% *Type: str:general (95.2), str:integer (4.8)*	
`ROOT.estimateValue`	4
Presence: 0.0% *Type: bool (100.0)*	
`ROOT.cantEstimate`	4
Presence: 0.0% *Type: bool (100.0)*	
`ROOT.carouselMap`	4
Presence: 0.0% *Type: str:json_encoded (100.0)*	
`ROOT.markets.[].teamBID`	4
Presence: 0.0% *Type: str:integer (100.0)*	

Table C.1 – continued from previous page

Path	**Total Count**
`ROOT.markets.[].teamAID` *Presence: 0.0%* *Type: str:integer (100.0)*	4
`ROOT.estimatedValue` *Presence: 0.0%* *Type: str:float (100.0)*	3
`ROOT.markets.[].sponsorImage` *Presence: 0.0%* *Type: str:empty (100.0)*	2
`ROOT.markets.[].createdBy` *Presence: 0.0%* *Type: int (100.0)*	1
`ROOT.markets.[].denominationToken` *Presence: 0.0%* *Type: str:empty (100.0)*	1

Table C.2: Raw Pairwise Comparison Statistics for CLOB Market Timestamps

T_1	T_2	Pairs (N)	$P(T_1 < T_2)$	$P(T_1 = T_2)$	$P(T_1 > T_2)$
`acceptingOrdersTimestamp`	`closedTime`	233 087	100.00	0.00	0.00
`acceptingOrdersTimestamp`	`createdAt`	266 051	0.00	0.00	100.00
`acceptingOrdersTimestamp`	`endDate`	263 251	98.36	0.00	1.64
`acceptingOrdersTimestamp`	`startDate`	263 888	99.69	0.00	0.31
`acceptingOrdersTimestamp`	`umaEndDate`	233 022	100.00	0.00	0.00
`acceptingOrdersTimestamp`	`updatedAt`	266 051	100.00	0.00	0.00
`closedTime`	`createdAt`	259 848	0.00	0.00	100.00
`closedTime`	`endDate`	258 742	16.98	0.00	83.02
`closedTime`	`startDate`	256 120	0.02	0.00	99.98
`closedTime`	`umaEndDate`	257 286	1.06	98.07	0.87
`closedTime`	`updatedAt`	259 848	99.99	0.01	0.00
`createdAt`	`endDate`	291 388	97.87	0.00	2.13
`createdAt`	`startDate`	290 272	93.83	0.19	5.98
`createdAt`	`umaEndDate`	257 297	99.90	0.00	0.10
`createdAt`	`updatedAt`	293 018	100.00	0.00	0.00
`endDate`	`startDate`	287 460	2.24	0.72	97.04
`endDate`	`umaEndDate`	256 223	82.24	0.88	16.89
`endDate`	`updatedAt`	290 166	77.45	0.00	22.55
`startDate`	`umaEndDate`	254 373	99.21	0.77	0.02
`startDate`	`updatedAt`	289 086	99.58	0.42	0.00
`umaEndDate`	`updatedAt`	257 297	99.91	0.00	0.09

Appendix D

Miscellaneous

Table D.1: Ten Most Frequent Tags in Matched Disputes

Label	**Appeared In (%)**
Politics	104 (32.00%)
Culture	99 (30.46%)
Sports	74 (22.77%)
Games	72 (22.15%)
Hide From New	49 (15.08%)
Recurring	45 (13.85%)
Weather	44 (13.54%)
Best of 2025	41 (12.62%)
Trump	40 (12.31%)
Geopolitics	39 (12.00%)

Table D.2: Ten Most Frequent Tags For Markets resolved to UNKOWN

Label	**Appeared In (%)**
Games	272 (96.11%)
Sports	259 (91.52%)
Esports	102 (36.04%)
Basketball	51 (18.02%)
NCAA	42 (14.84%)
UFC	39 (13.78%)
Dota 2	35 (12.37%)
NCAA Basketball	31 (10.95%)
Rocket League	30 (10.60%)
Tennis	29 (10.25%)

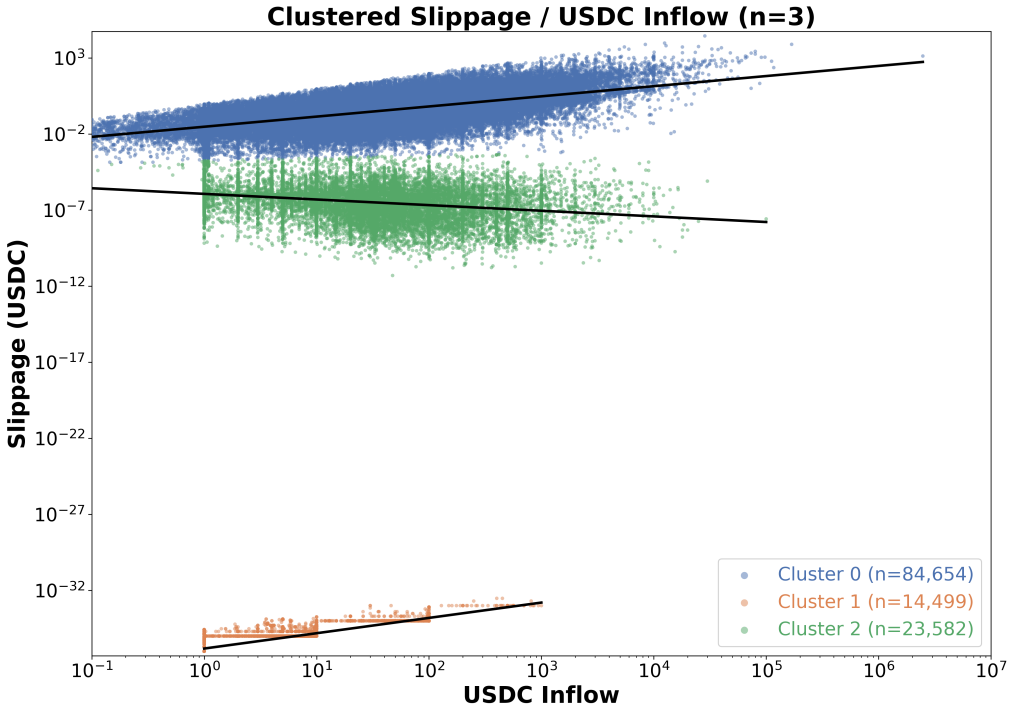

Figure D.1: Slippage to USDC inflow Clusters in Log Log Space

D.1 Slippage Computation Algorithm

The following algorithm extracts matched orders M, share quantity x, and market price p_{market} from the `OrderFilled` events emitted by the CTFExchange for a single settlement.

Algorithm 1 Extract Slippage Parameters from OrderFilled Events

Require: Set of `OrderFilled` events E for a settlement

Phase 1: Identify taker and extract order parameters
for each event $e \in E$ **do**
  if $e.\texttt{taker} = \texttt{EXCHANGE\_ADDRESS}$ **then**
    if $e.\texttt{makerAssetId} = 0$ **then**
      taker_side $\leftarrow$ buy
    else
      taker_side $\leftarrow$ sell
    end if
    $x \leftarrow \max(e.\texttt{making}, e.\texttt{taking})$ ▷ Share quantity, simply is the max between USDC amount and share amount
  end if
end for

Phase 2: Process maker fills to construct matched orders
$M \leftarrow \emptyset$
for each event $e \in E$ where $e.\texttt{taker} \neq \texttt{EXCHANGE\_ADDRESS}$ **do**
  if $e.\texttt{makerAssetId} = 0$ **then**
    maker_side $\leftarrow$ buy
  else
    maker_side $\leftarrow$ sell
  end if
  raw_price $\leftarrow \min(e.\texttt{making}, e.\texttt{taking})$ / $\max(e.\texttt{making}, e.\texttt{taking})$

  if maker_side = taker_side **then** ▷ In case of split or merge match the raw price has to be inverted to for the taker
    $p \leftarrow 1 -$ raw_price
  else
    $p \leftarrow$ raw_price
  end if

  $v \leftarrow \max(e.\texttt{making}, e.\texttt{taking})$ ▷ Share quantity
  $v \leftarrow v \cdot 10^{-6}$ ▷ Convert to correct units
  $M \leftarrow M \cup \{(p, v, NAN, 0)\}$ ▷ The queue priority and side are irrelevant for slippage computation
end for

Phase 3: Compute market price and USDC inflow in correct units
$p_{\text{market}} \leftarrow \min\{p \mid (p, v, s, i) \in M\}$
$x \leftarrow x \cdot 10^{-6}$ ▷ Convert to correct units

return M, p_{market}, x