



Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

*Distributed
Computing*



Polymarkets 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 9, 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

todo

Contents

Acknowledgements	i
Abstract	ii
1 Introduction	1
1.1 Speculation	1
1.1.1 Market Integrity	1
1.1.2 Modern Regulation	1
1.2 Prediction Market Exchanges	2
1.2.1 Growth	3
1.2.2 US Market	4
1.2.3 Accessibility	5
2 Objective and Methodology	6
2.1 Methodology	6
3 Endpoints and Clients	8
3.1 Polymarket Event and Market Endpoints	8
3.1.1 Event and Market Objects	8
3.2 Polymarket Orderbook and Trade Endpoints	9
3.2.1 Book Endpoint	9
3.2.2 Falsely Missing Orderbook State Objects	10
3.2.3 Market Feed	10
3.2.4 Ambiguous Event Ordering	11
3.3 Final Polymarket Client	11
3.4 Kalshi Endpoints	12
3.5 Kalshi Client	13

4 Smart Contracts	14
4.1 Conditional Tokens	14
4.2 UMA framework	15
4.3 UmaCtfAdapter	16
4.4 CTFExchange	16
4.5 Negrisk	17
4.6 Polygon Collection	17
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	24
6 Orderbook and Trading	25
6.1 State, Ordering and Matching	25
6.2 Slippage	26
6.2.1 Slippage Rate on Polymarket	26
6.3 Arbitrage	28
7 Discussion	29
Bibliography	30
A Endpoint Documented JSON responses	A-1
B Smart Contracts	B-1
C Polymarket Event Object JSON Tree Analysis	C-1

CHAPTER 1

Introduction

1.1 Speculation

Classifying and regulating speculation on uncertain events has historically been challenging. A common classification hinges on the purpose of the speculation, whether it serves to hedge risk and provide insurance, to enable price discovery for information aggregation and forecasting, or to function as entertainment akin to gambling.

1.1.1 Market Integrity

This regulatory ambiguity can undermine market integrity through two primary risks: direct manipulation and exploitation.

A moral hazard arises when market participants can directly influence an event's outcome. Match-fixing in sports provides an illustrative example: a boxer might have an economic incentive to lose deliberately if betting returns on their defeat exceed the prize for winning.

A related issue is asymmetric information. Not all informational advantages are malicious; some are an unavoidable consequence of information diffusion. However, the asymmetry becomes exploitative, potentially amounting to insider trading, when it arises from privileged access that allows a few actors to profit from unfairly obtained information without altering the outcome.

Both risks extend to more severe scenarios, from corporate managers taking positions that benefit from a project failure they can influence to insiders profiting from foreknowledge of military actions.

1.1.2 Modern Regulation

To mitigate these risks, regulators can prohibit speculation on events that present particularly high moral hazard or exploitation risks. For example, the U.S. Commodity Futures Trading Commission (CFTC) implements 17 C.F.R. § 40.11,

which states that a registered entity "shall not list for trading or accept for clearing" contracts that, among other things, "involve, relate to, or reference terrorism, assassination, war, gaming, or an activity that is unlawful under any State or Federal law" and permits the Commission to identify similar activities it deems contrary to the public interest [1]. This rule was the primary battleground that Kalshi challenged in order to list specific political prediction markets [2] and is a primary reason for Polymarket's restricted access for U.S. users [3].

1.2 Prediction Market Exchanges

Polymarket and Kalshi are two exchanges popularizing prediction markets and making them accessible globally. 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" [4]. Their legally binding rulebook states that the final outcome is decided at their discretion and that contract modifications are possible [5]. 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.

1.2.1 Growth

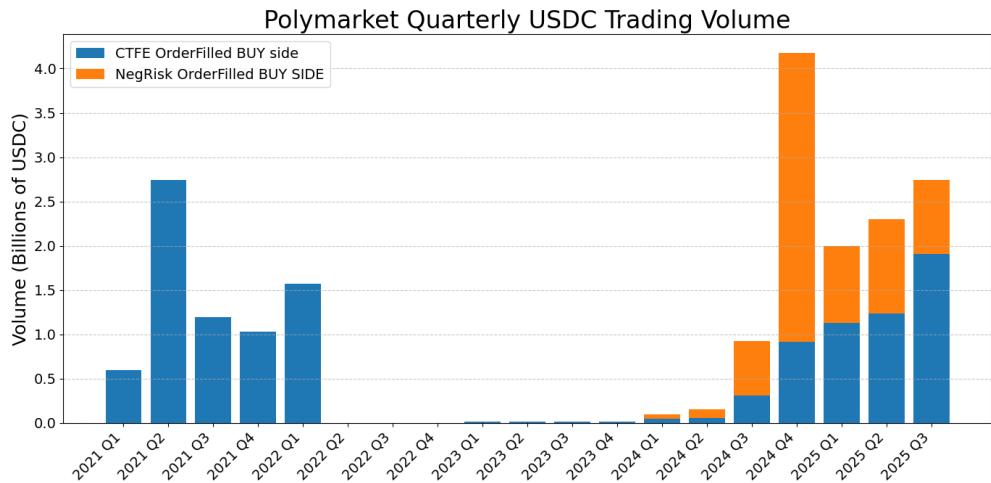


Figure 1.1: Quarterly trade volume on Polymarket. Source: Author’s own compilation using Dune Analytics (<https://dune.com/queries/6298071>).



Figure 1.2: Quarterly trade volume on Kalshi. (<https://tokenterminal.com/explorer/projects/kalshi/metrics/trading-volume?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 Basic CTFE 4.4.

1.2.2 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 [6, 7]. 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 [8]. On Polymarket, the main market for the presidential election accumulated a total volume of 3.5 billion USDC [9].

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 [10]. This occurred the month after the U.S. election markets concluded and coincided with a decline

of over 50% in total trading volume [11].

1.2.3 Accessibility

Kalshi expanded to over 140 countries in October 2025 [12], 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 [13]. This entity was subsequently rebranded as “Polymarket US” [14]. 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 [15].

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 [16].

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, hexadeciml, 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 order-book 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 additionally request the orderbook state, as the identifiers allow for unambiguous maintenance of the orderbook state. The Kalshi client therefore functions similarly to the Polymarket client, but without the need for periodic orderbook state requests.

CHAPTER 4

Smart Contracts

Polymarket has a dedicated github account with over 90 repositories. There are many individual repositories containing the solidity source for smart contracts and their deployment status. They also have a dedicated repository which lists any deployment address that seemingly can be found elsewhere, the associated contract and a security audit, if present [17]. It is not explicitly mentioned if it is an exhaustive list, but at least looking through the entire documentation on the website and all repositories on github, does not reveal any undeclared deployment, since there is no other authoritative source overriding this, it will be used as the reference of the disclosed deployments.

4.1 Conditional Tokens

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

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

$$Q = \sum_{i=1}^n O_i \quad (4.1)$$

The *splitPositions* method atomically swaps an amount n of collateral Q for n units 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 utilized, the condition must be prepared using the *prepareConditions* 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:

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

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

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

After this assignment we still have that (4.1) holds.

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

4.2 UMA framework

UMA designs protocols for oracles and deploys ethereum based contracts on various block chains like polygon. These deployments are connected to UMAs dedicated web application, which allows users actively to participate in solving price requests.

Polymarket exclusively used UMAs OptimisticOracleV2 (OOV2) deployment up until August 2025, where the UMIP-189 governance proposal passed and UMA deployed a ManagedOptimisticOracleV2 (MOOV2) dedicated only for Polymarket usage. The ManagedOptimisticOracleV2 is based on the same protocol, but with additional access control configurations [18].

Polymarket atomically sets the requested prices to be event based in its *UmaCtfAdapter* contract. Which starts with a price request. This emits the *RequestPrice* event. For a price request to be settled, the effective price must first be determined, this requires an initial price proposal to be made by suggesting a price for the request. This emits the *ProposePrice* event. After a price proposal is made, the price can be disputed within the liveness period which by default is two hours. If the proposal remains undisputed during this period, then it becomes the effective price for the request and the request can be settled. Both the proposals and disputes have no access control in OOV2. If the proposal is disputed, the *DisputePrice* event is emitted and then a stake based voting process starts. The contract for this is deployed on the Ethereum blockchain and therefore involves an off chain coordinated message relayer. After the price is determined the request can be settled, to reward or slash the proposer and disputer if existent, depending on the correctness of their proposals compared to the final outcome. This emits the *Settle* event.

The OOV2 does not specify what the prices exactly mean. So in the context of Polymarket the ancillary data has to formally specify semantic meaning of the integer price values. Polymarket does not explicitly state this, but analyzing the emitted events shows that virtually all ancillary data extends the UMIP-107 standard, such that the requested price identifier is *YES_OR_NO_QUERY* and the ancillary data is a utf-8 encoded dictionary, which specifies the question using the *q*:; the associated market by the *market id* key and the 4 possible prices through the *p1*:; *p2*:; *p3*: and *p4*: keys. Which specify which price is mapped to the outcomes *YES*, *NO*, *UNKNOWN* and *EARLY PROPOSAL* [19].

4.3 UmaCtfAdapter

This is Polymarkets custom contract, built on top of the CTF and OOV2, which it stores as immutable references by address. It provides an *initialize* which atomically prepares the condition on the CTF, using itself as an oracle and a hard coded outcome slot count of two, and also makes the price request to the OOV2. The *initialize* method is permissionless and emits the *QuestionInitialized* event. The other permissionless method *resolve*, emitting the *QuestionResolved* event, can only be invoked if the markets requested price has been determined and it is not paused. It calls the OOV2 to settle and get the *int256* price and if the price is not that of the *EARLY PROPOSAL* outcome then uses this price to report the payouts in the CTF. The *EARLY PROPOSAL* outcome is hard coded as the least *int256* value for comparison, a price of 0 is considered as *NO*, a price of 0.5 Ether is considered as *UNKNOWN*, any other price is considered *YES*. If the outcome is *NO* then the payout array passed to the CTF is [0,1], if the outcome is *UNKNOWN* then the it is [1,1] and otherwise it is [1,0] for the outcome *YES*.

The rest of the public methods in this contract all have access restricted to either *onlyAdmin* or *onlyOracle*. This also includes the *resolveManually* method which directly reports the payouts array passed without settling any price request.

4.4 CTFExchange

The CTFExchange is another custom contract by Polymarket it provides the *matchOrders* method used for settling a match on the CLOB between a taker order and *n* maker orders and emits one *OrdersMatched* event, and *n + 1* *OrderFilled* events. One OrderFilled event is for the taker facing the exchange and the other *n* are for the makers facing the taker. There are three types of matches; split, merge and complementary. A split match is when the taker and maker are both buyers, one for YES shares and the other for NO shares. Analogously, a merge match is when the taker and maker are both sellers, one for YES shares

and the other for NO shares. When the taker is selling the shares the maker is buying, or vice versa, then it is a complementary match. The match type and also the amount of the instruments exchanged can be deduced from the order filled events.

4.5 Negrisk

The Negrisk framework builds on top of the UmaCtfAdapter to create multiple markets with mutually exclusive outcomes and thereby provide the *convertPositions* method, which enables atomically swapping a set YES shares to the complementary set of NO shares or vice versa. This is implemented in the Negrisk Adapter, but the framework also entails a Negrisk Operator, which is used by the Negrisk UmaCtfAdapter as immutable reference for the conditional tokens contract interface, but the Negrisk Operators conditions prepared method does literally nothing and its reportPayouts method does not call any other contract, but it has other public methods which relay to the adapter, which actually references the CTF. Additionally a dedicated Negrisk CTFExchange deployment exists, which is simply another instance of the CTFExchange contract.

figure for immutable deployment references

4.6 Polygon Collection

The polygon collection is tailored for 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 the documentation. However, testing showed that the actual rate limit between requests without getting rate limited is more around 1s. The rtt for a single eth_getLogs request can be over 30s long, which is why its suited for concurrent requests. Besides the request frequency, there also is limit of 10'000 events that can requested. But the request is specified using a block range. For this purpose we can use alpha estimation with a margin 500 to converge to the block density for requesting the block expected to return 9'500 events. 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 amount of events returned have been compared to the amount of events returned to that found in dune over the same block range. The full list of collectected events can be found int the appendix B.2

CHAPTER 5

Market Lifecycle

To analyze the ma block numbers 79172086 inclusive and 81225971 inclusive, which is November 18th 2025 7:00:03 UTC to January 5th 2026 00:00:01 UTC, from polygon through the Infura client specified in [4.6](#) and historically fetchable market objects on Polymarkets endpoints we can analyze the market lifecycle.

5.1 Market Creation

5.1.1 Reporting Oracles

In section [4.1](#) we saw that the reporting oracle for a condition is passed as an argument for condition preparation on the CTF, and is the only address which can report payouts for the final YES and NO share value after resolution. Which is why it is essential to analyze which the reporting oracles being used by Polymarket.

Out of the 128'003 ConditionPreparation events, 124'340 have a matching off chain market object with the same condition id. These matched ConditionPreparation events have three distinct reporting oracles, two of which are not documented. Namely the *MOOV2 Adapter* and *Centralized Adapter* in Appendix [B.1](#).

The MOOV2 Adapter is an UmaCtfAdapter instance described in section [4.3](#). The MOOV2 Adapter uses the MOOV2 instead of the OOV2. Interestingly, no condition has been prepared with the Basic Adapter as reporting oracle. This aligns with the umip-189 proposal and suggests that the MOOV2 Adapter replaced the Basic Adapter, even though Polymarket has not disclosed anything on this matter.

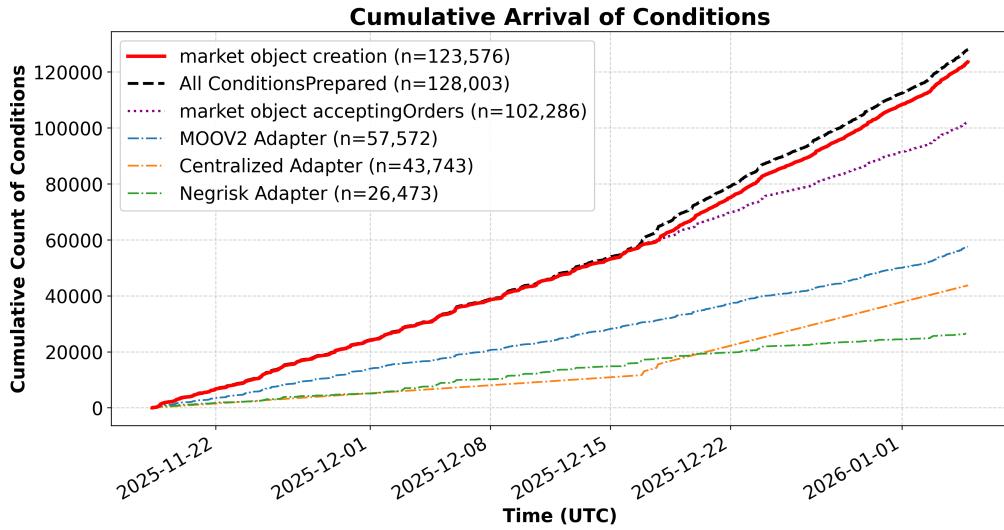


Figure 5.1: Cumulative count of conditions found over time

In figure 5.1 we can see that around December 17th 2025 some market objects start to appear without an acceptingOrders timestamp, as can also be seen in the appendix C.1. Additionally, this is where the 3'663 conditions were prepared on the CTF with no matching off chain market object, out of which virtually all use the Centralized Adapter as reporting oracle, which also had a change in the rate at which new conditions were prepared with it as reporting oracle.

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 more in depth.

Table 5.1: Label frequency in 41'016 market objects matched with 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 frequency of all labels in market objects matched with the conditions prepared with the Centralized Adapter as reporting oracle. Clearly, all labels are tied to crypto asset price related events. If we look up individual matched markets on polymarkets web application, we find that none of them have a link to the UMA resolution status on the UMA webapplication, which for other markets exists. Instead they link to a source which tracks the underlying crypto asset price, which is directly used as the ground truth for resolution. An example for this are the cyclical 15 minute “Bitcoin up or down?” markets.

5.1.3 Resolution Source

Whilst the MOOV2 Adapter makes a price request atomically with the condition preparation through the initialize method, a condition prepared for a Negrisk does not atomically make a price request and condition preparation atomically unless some hidden method is used for this, as explained in section 4.5. The ConditionPreparation events and RequestPrice events have nothing that can be used to match them directly. However, virtually every Ancillary data contains a market id key, which can be used to match price requests to market objects and then to ConditionPreparation events. Out of the 84,684 RequestPrice events, 83,743 have a matching off-chain market object and 83,652 have a matching ConditionPreparation.

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 out of the price requests made to the OOV2 matched with conditions prepared using the MOOV2 Adapter as reporting oracle, and 50 out of the requests made to the OOV2 matched with the MOOV2 Adapter as reporting oracle do not share the same transaction hash and are therefore not atomically executed using the initialize method.

The conditions prepared with the Negrisk Adapter as reporting oracle matched exclusively with price requests made to the OOV2. Out of these matches non share the same transaction hash, which means they are not executed atomically, but 21,672 (83.10%) share the same block number, which demonstrates that they mostly are executed in a timely manner, but coordinated through separate transactions.

5.2 Exchange Registration

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

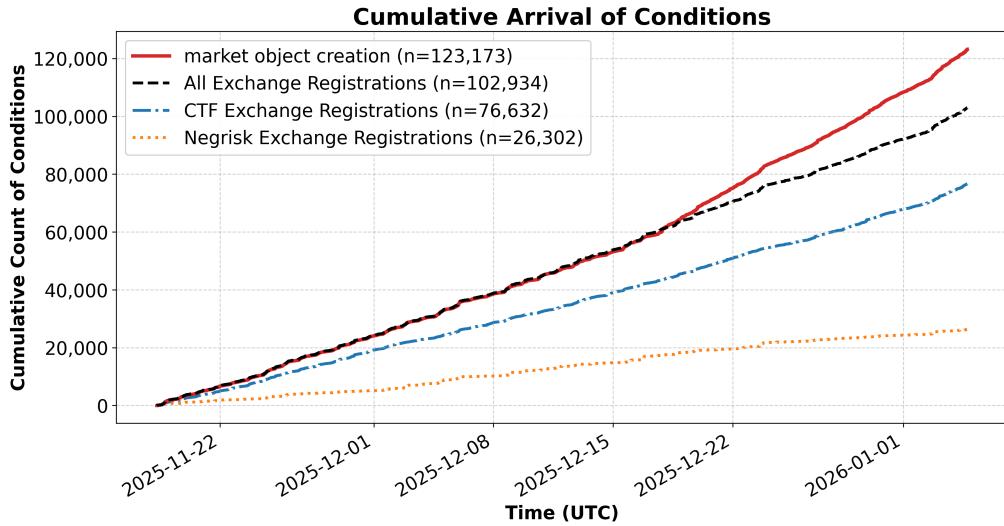


Figure 5.2: CLOB markets and registered tokens unique condito

In figure 5.2 we can see after some point in time, markets are consistently being created without being registered with the on chain exchange. This happens around the same time at which market objects start being created without acceptingOrder timestamps.

5.2.1 Unregistered Traded Outcomes

If we match the ConditionsPrepared events to the RegisteredToken events, we find that virtually all unregistered markets were prepared using the Centralized Adapter as reporting oracle. Additionally, less than half of the conditions are registered.

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)

Strangely, there are provably markets, which are visible on the web application, listed as active in the market object and have trade events being sent out on the websocket market feed, but do not have any registered token events nor orders matched or order filled events emitted from either the CTFE nor Negrisk CTFE. An example for this, are some of the 15 minute cyclical “bitcoin up or down?” markets.

5.3 Resolution

5.3.1 Disputes

As described in section 4.2 the UMA OOV2 and MOOV2 protocol allows for price proposals to be disputed within a liveness period.

Table 5.4: 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%)

There are a total of 74,833 matched proposals and 340 matched disputes. Table 5.4 shows the ten most frequent labels assigned to the markets objects matched with the disputes. Out of the 340 matched disputes, 145 are on the OOV2 and 195 on the MOOV2. This is ironic, because according to the umip-189 proposal, the MOOV2 was explicitly designed with a propososer whitelist, which Polymarket manages. Yet, the MOOV2 has more disputes than the OOV2. However, its important to note that the MOOV2 also has 53,818 proposals, compared to the OOV2 which has 21,015 proposals.

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 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, we would expect them to be reported equally, but most negrisk events are designed to have one prediction market resolve to YES and the rest to NO. In fact out of the 19'541 ConditionResolution events with a Negrisk Adapter as 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 of the UNKNOWN outcomes were reported by the MOOV2 Adapter.

Table 5.5: Ten Most Frequent Tags For Markets resolved to UNKNOWN

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%)

Table 5.5 shows what kind of labels are associated with markets reported to have an UNKNOWN outcome.

5.3.3 Manual Resolutions

In sections ?? it was mentioned that the UmaCtfAdapter and Negrisk Operator have administrative methods, which can be used to manually resolve markets without UMA price requests being settled. All of the relevant methods emit events, including the *resolveMarketManually* method in the UmaCtfAdapter and the *emergencyResolveQuestion* method in the Negrisk Operator. However, there no events for either of these methods was emitted in any of the UmaCtfAdapter instances or the single Negrisk Operator instance. This does not fully exclude the possibility of manual resolution, but if this is the case, then it must be done through a delegate call from a hidden proxy contract, such that the events are emitted from the proxy contract.

CHAPTER 6

Orderbook and Trading

In section 1.2 we explained how a prediction market consists of YES and NO shares whose price is denoted in USD. The CLOB used by Polymarket and Kalshi captures the intention of traders through market and limit orders, which are placed through HTTPS requests to *order* endpoints and are matched based on price-time priority [?]. A limit order allows the trader to specify price at which they want to buy or sell shares, but this price can not cross other resting limit orders, otherwise it becomes marketable. A marketable order is either executed immediately against the best possible resting limit orders by price-time priority, or is aborted [?].

6.1 State, Ordering and Matching

A trader can either buy or sell YES or NO shares, which creates four possible orders; Sell Yes, Sell No, Buy Yes, Buy No. However, among these four orders there are only two intentions, which are either to speculate on the YES outcome by buying YES shares or selling NO shares, or to speculate on the NO outcome by buying NO shares or selling YES shares.

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

The state of the CLOB can be defined through 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 for an outcome share, $v \in \mathbb{R}^+$ is the quantity of the shares, $s \in \{\text{YES}, \text{NO}\}$ is the outcome and $i \in \mathbb{N}$ is the priority it has to be selected for matching with market orders at price p and side s , a lower i has a higher priority. This guarantees a unique ordering for both buy and sell limit orders which is used for matching.

Formally, for limit orders we have that:

$$((p_1, v_1, s_1, i_1) > (p_2, v_1, s_2, i_2)) \iff p_1 < p_2 \vee (p_1 = p_2 \wedge i_1 < i_2) \quad (6.1)$$

Where the best price for the taker is:

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

If the quantity for a marketable order can not be fully filled at market price, then the taker would suffer from **price slippage**, if the order is executed. If the marketable order can't be filled with limit orders at any price, then 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

In general, Slippage occurs when the trade is executed at a different price than what the market suggested at the time where the intent to trade was submitted. This was already covered for fully Automated Market Makers in section ??.

Slippage in the CLOB can occur when a taker order for x shares is not filled entirely at the market price when the order was submitted, resulting in a less favorable execution cost for the taker. The CLOB is designed to allow this, with the core principle being the taker is guaranteed timely order execution but not price, whilst 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 , we have:

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

The least limit order $(p, v, s, i) = \min\{M\}$ might only be part of a limit order such that another limit order (p, v', s, i) now exists where the previous limit order was (p, v'', s, i) for $v'' = v' + v$ and:

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

6.2.1 Slippage Rate on Polymarket

Using the Orderfilled events emitted by the CTFExchange, we can measure the slippage for all 1'443'953 settlements, since we have the USDC amount that

the taker paid or received in the fill where taker is facing the exchange, and the amount of shares and their price for every fill where the maker is facing the taker. We can now construct a set A of tuples (x, y, t) for every settlement, where x is the USDC amount the taker paid or received, y is the slippage and t is the timestamp of the settlement.

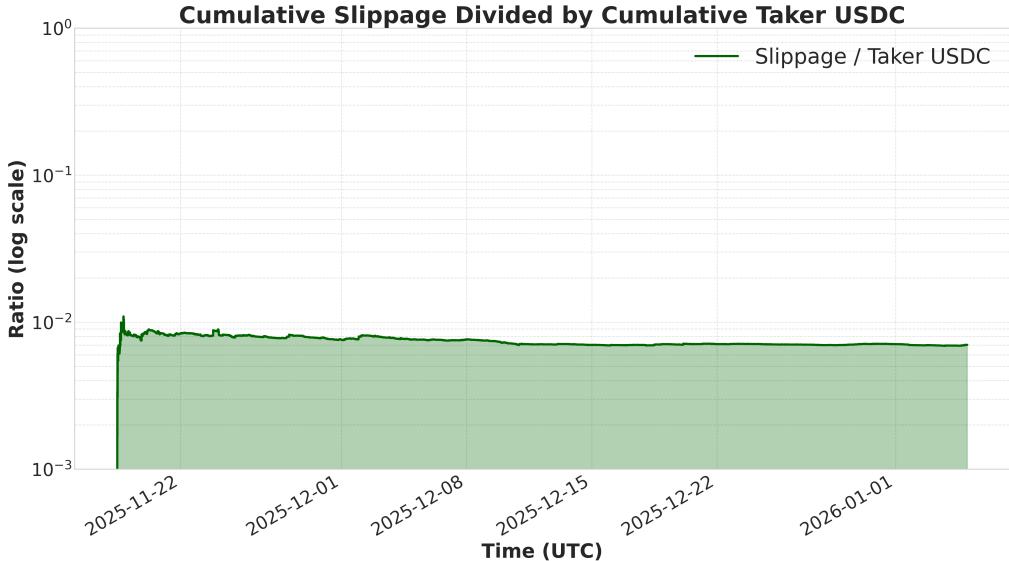


Figure 6.1: Cumulative Taker USDC Divided by Cumulative Slippage

In figure 6.1 we can see that the cumulative taker USDC amount divided by the cumulative slippage up until time t for all settlements, converges to some value around 100. This can be explained by the law of large numbers, if we assume that the x and y values in the tuples in A are i.i.d. and that the expected value of x is finite and non zero, then we have that:

$$\frac{\sum_{(x,y,t) \in A} x}{\sum_{(x,y,t) \in A} y} \xrightarrow{a.s.} \frac{E[x]}{E[y]} \quad (6.5)$$

Fitting $y = \alpha x + \beta$ for y as the cumulative Slippage amount and x as the cumulative Taker USDC amount, for the 1'412'812 settlements after one day in, to avoid the initial artifact, we find that $\alpha = 0.00126$ and $\beta = 2.04$ with a standard error $SE = 5.65 \cdot 10^{-6}$ and $R^2 = 0.01765$. This means that for the aggregate of all settlements we have an extremely good fit, but for individual settlements this is a poor predictor of slippage given the total USDC amount of the taker.

6.3 Arbitrage

In the prior section we saw that the event “Which party will win the house 2026?” had low expected price discrepancies. However, it is important to also consider that the price for the event itself did not move much, as illustrated by the range of the prices throughout the entire time period in table ??, and also was not close to being resolved yet. With the cyclical 15 minute “Bitcoin up or down?” event, we can also see how price discrepancies relate to the timeline in the market life cycle.

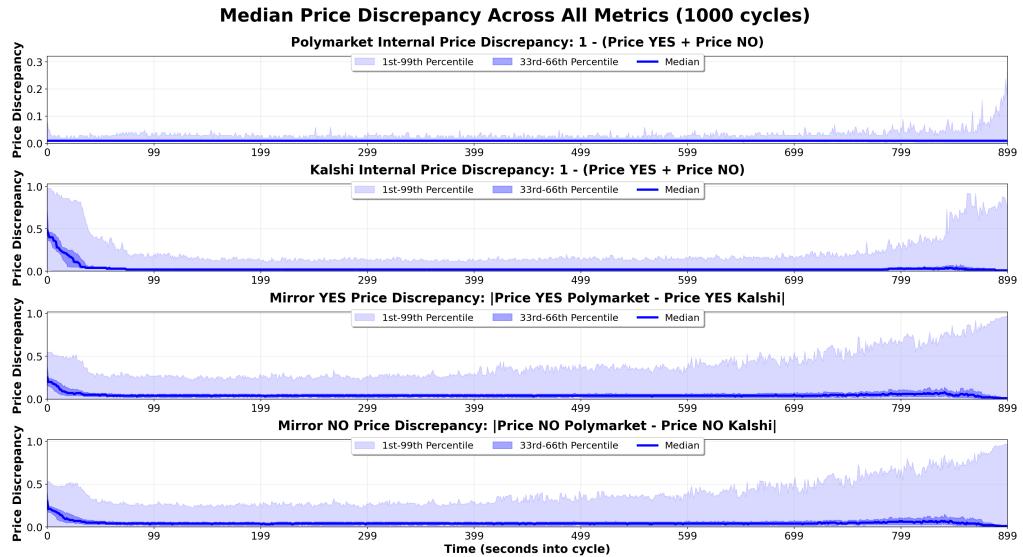


Figure 6.2: Price Discrepancy Distribution for 15 minute cyclical “Bitcoin up or down?” markets

CHAPTER 7

Discussion

Love is the answer.

Bibliography

- [1] C. L. School, “17 c.f.r. § 40.11,” 2026, prohibition on contracts involving terrorism, assassination, war, and gaming. [Online]. Available: <https://www.law.cornell.edu/cfr/text/17/40.11>
- [2] “KalshiEx LLC v. Commodity Futures Trading Commission,” D.D.C., September 2024, memorandum Opinion granting summary judgment to Plaintiff.
- [3] U.S. Commodity Futures Trading Commission, “In re Blockratize, Inc. d/b/a Polymarket.com,” CFTC Docket No. 22-09, January 2022, order Instituting Proceedings Pursuant to Section 6(c) and 6(d) of the Commodity Exchange Act.
- [4] Kalshi, “Payout on yes and no outcomes,” 2026. [Online]. Available: <https://help.kalshi.com/kalshi-101/what-are-prediction-markets>
- [5] Kalshi, “Kalshi rulebook,” 2026. [Online]. Available: <https://kalshi-public-docs.s3.amazonaws.com/regulatory/rulebook/Kalshi%20Rulebook%20v1.21.pdf>
- [6] 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>
- [7] 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>
- [8] G. Gottsegen, “In 2024, prediction markets called the presidential election before the polls could. now, they’re mostly betting on sports,” Morningstar, December 2024. [Online]. Available: <https://www.morningstar.com/news/marketwatch/20241220282/in-2024-prediction-markets-called-the-presidential-election-before-the-polls-could-now-theyre-m>
- [9] 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/>

- [10] 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>
- [11] 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
- [12] 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>
- [13] Markets Wiki. Qcx llc. [Online]. Available: https://www.marketswiki.com/wiki/QCX,_LLC
- [14] CFTC. Qcx llc dcm exchange registry. [Online]. Available: <https://www.cftc.gov/IndustryOversight/IndustryFilings/TradingOrganizations/49571>
- [15] 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>
- [16] D. Stabile, “Kalshi sues illinois regulators over cease-and-desist order,” Gaming Today, November 2025. [Online]. Available: <https://www.gamingtoda.com/news/kalshi-sues-illinois-regulators-over-cease-and-desist-order/>
- [17] Polymarket, “Polymarket contract security,” Github, November 2025. [Online]. Available: <https://github.com/Polymarket/contract-security>
- [18] UMAprotocol, “Umip-189,” Github, November 2025. [Online]. Available: <https://github.com/UMAProtocol/UMIPs/blob/master/UMIPs/umip-189.md>
- [19] UMAprotocol, “Umip-107,” Github, November 2025. [Online]. Available: <https://github.com/UMAProtocol/UMIPs/blob/master/UMIPs/umip-107.md>

APPENDIX A

Endpoint Documented JSON responses

Listing A.1: Polymarkets Documented Orderbook State Object Sample

```
1  {
2      "market": "0x1b6f76e5b8587ee896c35847e12d
3      11e75290a8c3934c5952e8a9d6e4c6f03cfa",
4      "asset_id": "1234567890",
5      "timestamp": "2023-10-01T12:00:00Z",
6      "hash": "0xabc123def456...",
7      "bids": [
8          {
9              "price": "1800.50",
10             "size": "10.5"
11         }
12     ],
13     "asks": [
14         {
15             "price": "1800.50",
16             "size": "10.5"
17         }
18     ],
19     "min_order_size": "0.001",
20     "tick_size": "0.01",
21     "neg_risk": false
22 }
```

Listing A.2: Polymarkets Documented Book Event

```
1  {
2      "event_type": "book",
3      "asset_id": "65818619657568813474341868652308942079804919
4      287380422192892211131408793125422",
5      "market": "0xbd31dc8a20211944f6b70f31557f1001557
6      b59905b7738480ca09bd4532f84af",
7      "bids": [
8          { "price": ".48", "size": "30" },
9          { "price": ".49", "size": "20" },
10         { "price": ".50", "size": "15" }]
```

```

11   ],
12   "asks": [
13     { "price": ".52", "size": "25" },
14     { "price": ".53", "size": "60" },
15     { "price": ".54", "size": "10" }
16   ],
17   "timestamp": "123456789000",
18   "hash": "0x0...."
19 }
```

Listing A.3: Polymarkets Documented Price Change Event

```

1 {
2   "market": "0x5f65177b394277fd294cd75650044e32
3   ba09a95022d88a0c1d565897d72f8f1",
4   "price_changes": [
5     {
6       "asset_id": "71321045679252212594626385532706912
7       750332728571942532289631379312455583992563",
8       "price": "0.5",
9       "size": "200",
10      "side": "BUY",
11      "hash": "56621a121a47ed9333273e21c83b660cff37ae50",
12      "best_bid": "0.5",
13      "best_ask": "1"
14    },
15    {
16      "asset_id": "521143195012459155160551060468842
17      09969926127482827954674443846427813813222426",
18      "price": "0.5",
19      "size": "200",
20      "side": "SELL",
21      "hash": "1895759e4df7a796bf4f1c5a5950b748306923e2",
22      "best_bid": "0",
23      "best_ask": "0.5"
24    }
25  ],
26  "timestamp": "1757908892351",
27  "event_type": "price_change"
28 }
```

Listing A.4: Polymarkets Documented Last Trade Price Event

```

1 {
2   "asset_id": "114122071509644379678018727908709560
3   226618148003371446110114509806601493071694",
4   "event_type": "last_trade_price",
5   "fee_rate_bps": "0",
6   "market": "0x6a67b9d828d53862160e470329ffea5246f3
7   38ecffffdf2cab45211ec578b0347",
8   "price": "0.456",
9   "side": "BUY",
10  "size": "219.217767",
11  "timestamp": "1750428146322"
```

12 }

Listing A.5: Polymarkets Documented New Market Event

```

1 {
2   "id": "1031769",
3   "question": "Will NVIDIA (NVDA) close above
4   $240 end of January?",
5   "market": "0x311d0c4b6671ab54af4970c06fcf
6   58662516f5168997bdda209ec3db5aa6b0c1",
7   "slug": "nvda-above-240-on-january-30-2026",
8   "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\n
     The 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.",
9   "assets_ids": [
10    "76043073756653678226373981964075571318
11    267289248134717369284518995922789326425",
12    "316909342633857276642020992785456880077
13    99199447969475608906331829650099442770"
14  ],
15  "outcomes": [
16    "Yes",
17    "No"
18  ],
19  "event_message": {
20    "id": "125819",
21    "ticker": "nvda-above-in-january-2026",
22    "slug": "nvda-above-in-january-2026",
23    "title": "Will NVIDIA (NVDA) close above ___ end of January
     ?",
24    "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\n
     If 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."
25 },
26 "timestamp": "1766790415550",
27 "event_type": "new_market"
28 }

```

Listing A.6: Polymarkets Documented Market Resolved Event

```

1 {
2   "id": "1031769",
3   "question": "Will NVIDIA (NVDA) close above $240
4   end of January?",
5   "market": "0x311d0c4b6671ab54af4970c06fc
6 f58662516f5168997bdda209ec3db5aa6b0c1",
7   "slug": "nvda-above-240-on-january-30-2026",
8   "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\\n
      The 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.",
9   "assets_ids": [
10     "760430737566536782263739819640755713182
11     67289248134717369284518995922789326425",
12     "316909342633857276642020992785456880077
13     99199447969475608906331829650099442770"

```

```

14 ],
15 "winning_asset_id": "7604307375665367822637398196407
16 5571318267289248134717369284518995922789326425",
17 "winning_outcome": "Yes",
18 "event_message": {
19     "id": "125819",
20     "ticker": "nvda-above-in-january-2026",
21     "slug": "nvda-above-in-january-2026",
22     "title": "Will NVIDIA (NVDA) close above ___ end of January
23         ?",
24     "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."
25 },
26 "timestamp": "1766790415550",
27 "event_type": "new_market"
27 }

```

Listing A.7: Kalshi Documented Orderbook Delta

```

1 {
2     "type": "orderbook_delta",
3     "sid": 2,
4     "seq": 3,
5     "msg": {
6         "market_ticker": "FED-23DEC-T3.00",
7         "market_id": "9b0f6b43-5b68-4f9f-9f02-9a2d1b8ac1a1",
8         "price": 96,
9         "price_dollars": "0.960",
10        "delta": -54,
11        "delta_fp": "-54.00",
12        "side": "yes",
13        "ts": "2022-11-22T20:44:01Z"
14    }
15 }

```

Listing A.8: Kalshi Documented Orderbook Snapshot

```

1 {
2   "type": "orderbook_snapshot",
3   "sid": 2,
4   "seq": 2,
5   "msg": {
6     "market_ticker": "FED-23DEC-T3.00",
7     "market_id": "9b0f6b43-5b68-4f9f-9f02-9a2d1b8ac1a1",
8     "yes": [
9       [
10         8,
11         300
12       ],
13       [
14         22,
15         333
16       ]
17     ],
18     "yes_dollars": [
19       [
20         "0.080",
21         300
22     ],
23     [
24       "0.220",
25       333
26     ]
27   ],
28   "yes_dollars_fp": [
29     [
30       "0.0800",
31       "300.00"
32     ],
33     [
34       "0.2200",
35       "333.00"
36     ]
37   ],
38   "no": [
39     [
40       54,
41       20
42     ],
43     [
44       56,
45       146
46     ]
47   ],
48   "no_dollars": [
49     [
50       "0.540",
51       20
52     ],

```

```

53     [
54         "0.560",
55         146
56     ]
57 ],
58 "no_dollars_fp": [
59     [
60         "0.5400",
61         "20.00"
62     ],
63     [
64         "0.5600",
65         "146.00"
66     ]
67 ]
68 }
69 }
```

Listing A.9: Kalshi Documented Trade Update

```

1 {
2   "type": "trade",
3   "sid": 11,
4   "msg": {
5     "trade_id": "d91bc706-ee49-470d-82d8-11418bda6fed",
6     "market_ticker": "HIGHNY-22DEC23-B53.5",
7     "yes_price": 36,
8     "yes_price_dollars": "0.360",
9     "no_price": 64,
10    "no_price_dollars": "0.640",
11    "count": 136,
12    "count_fp": "136.00",
13    "taker_side": "no",
14    "ts": 1669149841
15  }
16 }
```

Listing A.10: Kalshi Documented Market Lifecycle V2

```

1 {
2   "type": "market_lifecycle_v2",
3   "sid": 13,
4   "msg": {
5     "market_ticker": "INXD-23SEP14-B4487",
6     "event_type": "created",
7     "open_ts": 1694635200,
8     "close_ts": 1694721600,
9     "additional_metadata": {
10       "name": "S&P 500 daily return on Sep 14",
11       "title": "S&P 500 closes up by 0.02% or more",
12       "yes_sub_title": "S&P 500 closes up 0.02%+",
13       "no_sub_title": "S&P 500 closes up <0.02%",
14       "rules_primary": "The S&P 500 index level at 4:00 PM ET...",
15       "rules_secondary": ""
16     }
17   }
18 }
```

```

16     "can_close_early": true,
17     "event_ticker": "INXD-23SEP14",
18     "expected_expiration_ts": 1694721600,
19     "strike_type": "greater",
20     "floor_strike": 4487
21   }
22 }
23 }
```

Listing A.11: Kalshi Documented Event Object

```

1 {
2   "event": {
3     "event_ticker": "<string>",
4     "series_ticker": "<string>",
5     "sub_title": "<string>",
6     "title": "<string>",
7     "collateral_return_type": "<string>",
8     "mutually_exclusive": true,
9     "category": "<string>",
10    "available_on_brokers": true,
11    "product_metadata": {},
12    "strike_date": "2023-11-07T05:31:56Z",
13    "strike_period": "<string>",
14    "markets": [
15      {
16        "ticker": "<string>",
17        "event_ticker": "<string>",
18        "market_type": "binary",
19        "title": "<string>",
20        "subtitle": "<string>",
21        "yes_sub_title": "<string>",
22        "no_sub_title": "<string>",
23        "created_time": "2023-11-07T05:31:56Z",
24        "updated_time": "2023-11-07T05:31:56Z",
25        "open_time": "2023-11-07T05:31:56Z",
26        "close_time": "2023-11-07T05:31:56Z",
27        "expiration_time": "2023-11-07T05:31:56Z",
28        "latest_expiration_time": "2023-11-07T05:31:56Z",
29        "settlement_timer_seconds": 123,
30        "status": "initialized",
31        "response_price_units": "usd_cent",
32        "yes_bid": 123,
33        "yes_bid_dollars": "0.5600",
34        "yes_ask": 123,
35        "yes_ask_dollars": "0.5600",
36        "no_bid": 123,
37        "no_bid_dollars": "0.5600",
38        "no_ask": 123,
39        "no_ask_dollars": "0.5600",
40        "last_price": 123,
41        "last_price_dollars": "0.5600",
42        "volume": 123,
43        "volume_fp": "10.00",
44      }
45    ]
46  }
47 }
```

```
44     "volume_24h": 123,
45     "volume_24h_fp": "10.00",
46     "result": "yes",
47     "can_close_early": true,
48     "open_interest": 123,
49     "open_interest_fp": "10.00",
50     "notional_value": 123,
51     "notional_value_dollars": "0.5600",
52     "previous_yes_bid": 123,
53     "previous_yes_bid_dollars": "0.5600",
54     "previous_yes_ask": 123,
55     "previous_yes_ask_dollars": "0.5600",
56     "previous_price": 123,
57     "previous_price_dollars": "0.5600",
58     "liquidity": 123,
59     "liquidity_dollars": "0.5600",
60     "expiration_value": "<string>",
61     "tick_size": 123,
62     "rules_primary": "<string>",
63     "rules_secondary": "<string>",
64     "price_level_structure": "<string>",
65     "price_ranges": [
66         {
67             "start": "<string>",
68             "end": "<string>",
69             "step": "<string>"
70         }
71     ],
72     "expected_expiration_time": "2023-11-07T05:31:56Z",
73     "settlement_value": 123,
74     "settlement_value_dollars": "0.5600",
75     "settlement_ts": "2023-11-07T05:31:56Z",
76     "fee_waiver_expiration_time": "2023-11-07T05:31:56Z",
77     "early_close_condition": "<string>",
78     "strike_type": "greater",
79     "floor_strike": 123,
80     "cap_strike": 123,
81     "functional_strike": "<string>",
82     "custom_strike": {},
83     "mve_collection_ticker": "<string>",
84     "mve_selected_legs": [
85         {
86             "event_ticker": "<string>",
87             "market_ticker": "<string>",
88             "side": "<string>",
89             "yes_settlement_value_dollars": "0.5600"
90         }
91     ],
92     "primary_participant_key": "<string>",
93     "is_provisional": true
94   }
95 ]
96 },
97 "markets": [
```

```
98  {
99      "ticker": "<string>",
100     "event_ticker": "<string>",
101     "market_type": "binary",
102     "title": "<string>",
103     "subtitle": "<string>",
104     "yes_sub_title": "<string>",
105     "no_sub_title": "<string>",
106     "created_time": "2023-11-07T05:31:56Z",
107     "updated_time": "2023-11-07T05:31:56Z",
108     "open_time": "2023-11-07T05:31:56Z",
109     "close_time": "2023-11-07T05:31:56Z",
110     "expiration_time": "2023-11-07T05:31:56Z",
111     "latest_expiration_time": "2023-11-07T05:31:56Z",
112     "settlement_timer_seconds": 123,
113     "status": "initialized",
114     "response_price_units": "usd_cent",
115     "yes_bid": 123,
116     "yes_bid_dollars": "0.5600",
117     "yes_ask": 123,
118     "yes_ask_dollars": "0.5600",
119     "no_bid": 123,
120     "no_bid_dollars": "0.5600",
121     "no_ask": 123,
122     "no_ask_dollars": "0.5600",
123     "last_price": 123,
124     "last_price_dollars": "0.5600",
125     "volume": 123,
126     "volume_fp": "10.00",
127     "volume_24h": 123,
128     "volume_24h_fp": "10.00",
129     "result": "yes",
130     "can_close_early": true,
131     "open_interest": 123,
132     "open_interest_fp": "10.00",
133     "notional_value": 123,
134     "notional_value_dollars": "0.5600",
135     "previous_yes_bid": 123,
136     "previous_yes_bid_dollars": "0.5600",
137     "previous_yes_ask": 123,
138     "previous_yes_ask_dollars": "0.5600",
139     "previous_price": 123,
140     "previous_price_dollars": "0.5600",
141     "liquidity": 123,
142     "liquidity_dollars": "0.5600",
143     "expiration_value": "<string>",
144     "tick_size": 123,
145     "rules_primary": "<string>",
146     "rules_secondary": "<string>",
147     "price_level_structure": "<string>",
148     "price_ranges": [
149         {
150             "start": "<string>",
151             "end": "<string>",


```

```
152         "step": "<string>"  
153     }  
154 ],  
155 "expected_expiration_time": "2023-11-07T05:31:56Z",  
156 "settlement_value": 123,  
157 "settlement_value_dollars": "0.5600",  
158 "settlement_ts": "2023-11-07T05:31:56Z",  
159 "fee_waiver_expiration_time": "2023-11-07T05:31:56Z",  
160 "early_close_condition": "<string>",  
161 "strike_type": "greater",  
162 "floor_strike": 123,  
163 "cap_strike": 123,  
164 "functional_strike": "<string>",  
165 "custom_strike": {},  
166 "mve_collection_ticker": "<string>",  
167 "mve_selected_legs": [  
168     {  
169         "event_ticker": "<string>",  
170         "market_ticker": "<string>",  
171         "side": "<string>",  
172         "yes_settlement_value_dollars": "0.5600"  
173     }  
174 ],  
175 "primary_participant_key": "<string>",  
176 "is_provisional": true  
177 }  
178 ]  
179 }
```

APPENDIX B

Smart Contracts

Table B.1: Contract Aliases, Deployment Addresses, and Event Tables

Alias	Deployment Address	Contract Table Name
Base CTFExchange	0x4bf841d5b3570de fd03c39a9a4d8de6b d8b8982e	CTFExchange
Negrisk CTFExchange	0xC5d563A36AE78 145C45a50134d48A 1215220f80a	CTFExchange
CTF	0x4d97dc97ec945f 40cf65f87097ace5ea 0476045	(Gnosis) Conditional Tokens
Negrisk Operator	0x71523d0f655b41e 805cec45b17163f52 8b59b820	Negrisk Operator
Negrisk Adapter	0xd91e80cf2e7be2e 162c6513ced06f1dd 0da35296	Negrisk Adapter
Negrisk Adapter	0x2F5e3684cb1F31 8ec51b00Edba38d7 9Ac2c0aA9d	UmaCtfAdapter
Base UmaCtfAdapter	0x6A9D222616C90 FcA5754cd1333cF D9b7fb6a4F74	UmaCtfAdapter
MOOV2 Adapter	0x65070BE9147746 0D8A7AeEb94ef92f e056C2f2A7	UmaCtfAdapter
MOOV2	0x2C0367a9DB231 dDeBd88a94b4f646 1a6e47C58B1	(UMA) Managed Optimistic Oracle V2
Oracle Child Tunnel	0xac60353a54873c4 46101216829a6A98 cDbbC3f3D	Oracle Child Tunnel
FxTunnel	0x8397259c983751 DAf4040079006393 5a11afa28a	FxTunnel
OOV2	0xeE3Afe347D5C74 317041E2618C4953 4dAf887c24	(UMA) Optimistic Oracle V2

Table B.2: Contract Event Signatures

Contract / Event Signatures
CTFExchange

Table B.2 – continued from previous page

Contract / Event Signatures
<pre> 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)) </pre>
Conditional Tokens
<pre> ConditionPreparation(conditionId(bytes32), oracle(address), questionId(bytes32), outcomeSlotCount(uint256)) 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(redememer(address), collateralToken(address), parentCollectionId(bytes32), conditionId(bytes32), indexSets(uint256[]), payout(uint256)) URI(value(string), id(uint256)) </pre>
Neg Risk Operator
<pre> MarketPrepared(marketId(bytes32), feeBips(uint256), data(bytes)) </pre>

Table B.2 – continued from previous page

Contract / Event Signatures
<pre> 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)) </pre>
Neg Risk Adapter <pre> 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)) PositionsMerge(stakeholder(address), conditionId(bytes32), amount(uint256)) QuestionPrepared(marketId(bytes32), questionId(bytes32), index(uint256), data(bytes)) RemovedAdmin(admin(address), oldAdmin(address)) </pre>
UmaCtfadapter <pre> 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)) </pre>

Table B.2 – continued from previous page

Contract / Event Signatures
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))
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))

Table B.2 – continued from previous page

Contract / Event Signatures
<pre>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))</pre>
Oracle Child Tunnel
<pre>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)) PriceRequestAdded(identifier(bytes32), timestamp(uint256), ancillaryData(bytes), childRequestId(bytes32)) PushedPrice(identifier(bytes32), timestamp(uint256), ancillaryData(bytes), price(int256), childRequestId(bytes32)) MessageSent(data(bytes))</pre>
FxTunnel
<pre>NewFxMessage(rootMessageSender(address), receiver(address), data(bytes))</pre>
Optimistic Oracle V2
<pre>RequestPrice(requester(address), identifier(bytes32), timestamp(uint256), ancillaryData(bytes), currency(address), reward(uint256), finalFee(uint256))</pre>

Table B.2 – continued from previous page

Contract / Event Signatures

```
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))
```

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	126558
<i>Presence: 100.0% Type: Object (100.0)</i>	
ROOT.tags	126555
<i>Presence: 100.0% Type: Object (100.0)</i>	
ROOT.tags. []	126555
<i>Presence: 100.0% Type: Array (100.0)</i>	
ROOT.markets	126523
<i>Presence: 100.0% Type: Object (100.0)</i>	
ROOT.markets. []	126523
<i>Presence: 100.0% Type: Array (100.0)</i>	
ROOT.series	105820
<i>Presence: 83.6% Type: Object (100.0)</i>	
ROOT.series. []	105820
<i>Presence: 83.6% Type: Array (100.0)</i>	
ROOT.markets. []. clobRewards	9805
<i>Presence: 7.7% Type: Object (100.0)</i>	
ROOT.markets. []. clobRewards. []	9805
<i>Presence: 7.7% Type: Array (100.0)</i>	

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

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