Polymarket

How to verify Polymarket Proposals

What is the YES_OR_NO_QUERY?

The YES_OR_NO_QUERY (UMIP can be readhere) is a price identifier intended for plaintext binary questions. Requesters encode their questions in ancillary data. For a full overview of how ancillary data is formatted, referhere .

The YES_OR_NO_QUERY is most typically used by Polymarket today to resolve the outcome of information markets. This can come in the form of asking for the outcome of sports games, a crypto or nft price at a specific point in time or world events among many other things.

Note that this price identifier is often in the form of YES or NO, but could also be in the form of THIS or THAT - ie “did the Lakers or Clippers win the game last night” can work even though it is not really a YES or NO question. The example used in the rest of this document will illustrate how this can be done.

Components of Ancillary Data

From the UMIP:

When converted from bytes to UTF-8, the ancillary data should be a dictionary object containing q (question), p1, p2, p3 and p4 keys and values. p4 is optional and will only apply in certain situations.

Example:

q: title: French Open Final: Djokovic vs. Ruud, description: This market will resolve to "Djokovic" if Novak Djokovic wins the final, or to "Ruud" if Casper Ruud wins the final. res_data: p1: 0, p2: 1, p3: 0.5. Where p1 corresponds to Ruud, p2 to Djokovic, p3 to unknown/50-50,initializer:91430cad2d3975766499717fa0d66a78d814e5c5

The UMIP goes on to note default values for p1, p2, p3 and p4.

- p1 is usually used for “NO” values and defaults to`0` if not explicitly assigned
- p2 is used for “YES” values and defaults to `1` if not explicitly assigned
- p3 is for “UNKNOWN” or “CANNOT BE DETERMINED” and defaults to`0.5` if not explicitly assigned.
- p4 is for situations where the question is expected to eventually be able to be evaluated, but cannot be at this time. An example would be if the outcome to a sports game was asked for, but the game has not yet happened or finished. This will be referred to as the “magic number” and defaults to the minimum int256 value or before e18 scaling: -57896044618658097711785492504343953926634992332820282019728.792003956564819968
-

Important Note: Polymarket currently uses OptimisticOracleV2. Therefore, the early request should only be used if a proposed value is proposed earlier than the expected event resolution time noted in ancillary data.

For additional context on when the magic number or p3 are expected to be returned, from the UMIP:

p4 is intended to be used for situations where it is not a given that the price request (or contract settlement) should even occur yet. An example of this would be the UMA event-based expiry LSP. A request to settle an event-based expiry LSP can be submitted at any time but if the question can not be resolved yet it should be ignored.

The default p4 value is the minimum int256 value and is used as a "magic number" to indicate that an event-based expiry request is invalid and the contract should continue as normal. For example, if the question is related to a basketball game on January 6th and a settlement request comes in on January 5th, the question can not be resolved yet, and voters should return the p4 value with the magic number to reject the settlement request. This value also moves the decimal place 18 spaces to the left, due to the default behavior of the UMA voting interface to scale input values to 18 decimals. After scaling by the interface, the value will be -57896044618658097711785492504343953926634992332820282019728792003956564819968

Notice that a p3 value would never be returned earlier than the final price request time noted in the ancillary data or the requesting contract's expiration timestamp and a p4 value would never be returned after that point. Consider an unresolvable question like, "Was the weather nice on January 6th, 2022?" If the question was asked on January 7th, 2022, you would return the p3 value. If the same question was asked on January 5th, 2022, you would return the p4 value.

Scaling values

Also noted in the UMIP is the fact that return values are not scaled when referred to in the UMIP or present in ancillary data.

This simply means that if the ancillary data contains “p2:1”, this should really be returned on-chain as 1e18 (not 1 wei). So 1000000000000000000 instead of 1.

All price identifier values are treated this way in the UMA system. The UMA voter dapp and Optimistic Oracle proposer interface both perform this scaling automatically for UMA voters or proposers.

Understanding Question Data

When verifying a proposal, it is important to assess the question data and arrive at their own conclusion for what the return value should be. Here is the example above in the oracle UI:

Key data in the proposal is the identifier, timestamp, and the ancillaryData.

The YES_OR_NO_QUERY identifier tells us which “pricing methodology” we should be referring to. This is best understood by reading the corresponding UMIP. For all price identifiers, the UMIP can be looked up[here](#) by price identifier name.

The proposed time tells us the timestamp a value was proposed and therefore should be evaluated.

The Additional Text Data above represents the ancillary data and contains the plaintext version of the binary question that a proposer should be evaluating. It is important to read the full text data as it could also contain pricing specifications or data sources that should be used.

In the example, I can see that the question is asking if Novak Djokovic or Casper Ruud wins the French Open Final. It does not specify a data source so I can use a wide array of publicly available information to determine what I think the correct answer should be. Referring to ESPN scores:

?

I see that Djokovic won and I should continue to read the ancillary data to determine which return value should be used in this scenario.

Ancillary data question:

This market will resolve to "Djokovic" if Novak Djokovic wins the final, or to "Ruud" if Casper Ruud wins the final.

Resolution key:

res_data: p1: 0, p2: 1, p3: 0.5. Where p1 corresponds to Ruud, p2 to Djokovic, p3 to unknown/50-50

Since Djokovic won and p2 corresponds to 1, the proposed value should be p2:1.

Voters/proposers should typically evaluate all price requests independent of the party that has requested the data. But as a validation tool, the requestor can sometimes provide information that will help us verify the values that we are about to propose. As an example, this question was fromhttps://polymarket.com/ .

This is one simple example of evaluating a YES_OR_NO_QUERY, but almost all questions follow the same format.