Tellor Layer

Draft Notice: *Tellor Layer is currently in production. This is an overview of the intended workings of the system. Feedback and contributions are very welcome and we hope to engage in a dialogue with the community throughout the design and build process.*

Last Edited: *8 Oct 2024*

How the Chain Works

Tellor Layer is a stand alone L1 built using the cosmos sdk for the purpose of coming to consensus on any subjective data. It works by using a network of staked parties who are crypto-economically incentivized to honestly report requested data.

Submitting Values

Any staked reporter can submit a value for a given query (i.e. data request, e.g. BTC/USD price). All queries have a report time frame. For that given time period, reporters can add their value to the array of submissions. At the end of the time period, the official value is determined by a weighted aggregation of all the values submitted.

All queries have an aggregation type associated with it (e.g. median, mode, average, etc.). Once the time frame ends, the reports are subject to weighted aggregation. For this reason, larger reporters contribute more to the official value than smaller ones. Each official value takes a minimum of two blocks (target time 2s per block). The first block is the tip (e.g. “I request BTC/USD”). The second block is the reporting and aggregation phase, where all values are submitted and then aggregated, at which point an official value is determined for the query. Tips are distributed to all reporters for a given query using their weighted contribution.

The reporting time frame begins once a queryId is tipped. All reporters can add their value to the submission array for inclusion in the aggregation and for distribution of the rewards.[1] Tips received within the time frame are added to the initial tip. If no reports are submitted during the time frame, the time frame is restarted upon the next tip and the original tip is added to the new tip for that queryId.

Cycle List

In order to maintain freshness, the system will maintain a list of enshrined queries. Reporters will continuously report for the next query in the list to ensure that tips do not need to be submitted in previous blocks to have a base level of reporting (thus avoiding wasting gas by validators who just want inflationary rewards). This will signal reporters what to report so they act in unison and contribute as much aggregate power to a single report as is available. Reporters are never required to report for any queries including those in this list, but as long as inflationary rewards are enough to cover the gas costs of reporting, some query from the cycle list should be reported to consensus every block. Cycle list changes are voted on by governance.

[1] The reason for this is that some queries are not automatic, e.g. “lets type in an answer manually”, so we want to give room for non-time sensitive queries to get more reports. Spot prices and automated queries will have a report time frame of 1 or 2 blocks.

Disputing

Disputes work differently than with previous versions of Tellor. Reporter stakes are not tied to specific values, but rather just to a given reporter. For instance, a reporter can submit ETH/USD once every second with their same stake. If they submitted a bad value 2 days ago, they can still be subject to a slashing event. Similar to the old Tellor system, however, is the idea that since you can censor a party by disputing, disputes are a bet or wager on who is correct (the reporter or the accuser).

Any party can raise a dispute with free floating TRB, but unlike the old system, reporters and validators can use their stake (or part of it) to begin a dispute. Once initiated, the dispute fee and the potential slash amount (from the accused reporter) are put in escrow and removed from corresponding staking powers.

To initiate a dispute, the disputing party will submit a dispute against a given validator for one of three categories:

- Warning (dispute fee is 1% of stake) - jail[2] with no minimum time lock, can call a function to be released from jail and begin reporting again
- Minor Infraction (dispute fee is 5% of stake) - jailed for 10 minutes and out when they call the release from jail function
- Major Infraction (dispute fee is 100% of stake) - jail until dispute over (since 100% of stake).

A release function has to be called after a warning or minor infraction to ensure the staker has looked at the dispute and implemented a fix as necessary. Infractions in these lower two tiers can generally be assumed to not be malicious.

After specifying the dispute category, the disputer will submit an amount of TRB up to the minimum slashing amount before the dispute can initiate. If they don't have enough funds themselves, for up to one day, others can add to the pot until they hit the slashing amount(1, 5, or 100 percent depending on the slashing category). Once the amount is hit (could be hit instantly upon proposing the dispute, or could take up to a day), the potential slashing amount will be taken from the disputed validator and placed into a locked stake.

For up to two days, stakeholders will vote on which side of the dispute they support. Tellor Layer will use the same voting distribution as traditional Tellor governance:

25% users (tips)
25% reporters
25% token holders
25% team

[2] "Jail" is a concept in the tendermint system where the validator (or reporter in this case) is locked out of participating for a certain period of time

Once the two days are over there is a one day period where the dispute can be reopened and the same two day voting round is repeated. However, if at any point a quorum of >50% total voting power votes in favor of one side of the dispute, the dispute is considered finalized and no new rounds can be opened.

Once the dispute is resolved, the stake from the losing party is transferred to the winning party(ies) as undelegated, staked TRB. Tokens disputed or used as fees in disputes are not released as free floating tokens so as disputes cannot be used as a way to exit staking faster, however if free floating tokens are used to pay the fee, they are returned as free floating tokens in the case of a successful dispute.

Note that each dispute round (even the initial one) takes 5% of the dispute fee. Of that 5%, half is burned and the other half is divided amongst voters in the system. The dispute fee then doubles each round up to the slash amount to further incentivize voting and to prevent spamming. Once quorum is reached on a dispute, further dispute rounds cannot be raised.

For usage purposes, values are not attested to if the disputed reporter was the official aggregated value (median contribution). Values that need to be flagged are also added to evidence when the dispute is initiated (an array of values). For this reason, users who rely on optimistic finality should be aware of dispute censoring attacks and the potential for values to take time to get through.

Tipping

All tips will be in TRB. Each query can be tipped directly and its tip will increase as more users tip it. Once a report is submitted and aggregated, 98% of the tip will be split amongst reporters for that query in that given block. There will be a 2% fee on the tips (to prevent vote farming/ spamming). This 2% fee will be burned. Tips are distributed as locked TRB. Once locked, parties can run a function to claim the tip, which acts as identical to “depositStake” in which the tips are unlocked and added to their stake. This is to ensure that parties cannot bypass validator deposit limits through tipping, as well as to prevent farming vote power via the tipping mechanism.

Note that tipping in Layer will consist of only one time tips. The idea of built in heart beats or price thresholds is nice, but the complexity added is unnecessary and better handled off-chain (e.g. tip bots like the autoTipper we currently have handles this functionality already).

Validator Set Dynamics

Chains using the tendermint consensus mechanism have a limited number of validators. By setting the number of validators, this allows for efficient interconnectedness between chains as well as faster throughput. The tradeoff that any chain must consider is that more validators leads to slower blocktimes. Tellor will start with a limit of 100 validators (more than current

reporting set), but will move to a larger set as technological advances or market conditions allow (e.g. if no one needs sub-2 second blocks). The current target blocktime is 2 seconds.

The validator set can only change by a maximum of 5% per 12 hours including tip claims. This is for purposes of maintaining a stable validator set for bridging efficiency. Once the 5% change is hit, new validators will need to wait until rolling percent change per 12 hours is under the cap. This 5% is a parameter that can be changed by governance.[3]

Dual Delegation - Reporting vs Validating

Tendermint uses a delegated proof-of-stake(dPoS) model where there is a set number of validators, but all token holders can delegate to the top validators to share in rewards. Tellor uses this delegation but adds a second delegation for reporting duties. Each token can be used as a stake for reporting and for validating. Parties can delegate both the reporting and validating to the same party, to different parties, or even to themselves. The same token is subject to slashing by either method (reporting data or failing to honestly validate the chain) and the stake balance for both delegations is reduced immediately upon either consequence.

The reason for this dual delegation is that validator sets are capped in tendermint based systems, however we need to remove that cap to enable smaller and more reporters to help decentralize the data provider set. Additionally, the cost of bridging is directly tied to the validator set size (verifying signatures for the light client bridges), so a large validator set such as Ethereum is unfeasible for our intended uses (the need for fast, cheap bridging of data).

Tokenomics

TRB will be the native currency of the chain and used for staking, tipping, and voting. All tokenomics will remain the same. Four thousand tokens will continue to be the dev share to the team each month and an equivalent amount of tokens will be distributed to reporters and validators as inflationary rewards. Seventy-five percent of time based rewards will be given to reporters with the other twenty-five percent given to validators.

Gas fees will be given to the block proposing validator (selected randomly by validator weight), while tips and inflationary rewards to reporters are distributed proportionally to the reports submissions based on weighted validator support. As an example, if a query is submitted for and gets 50% of the stake contributing to the aggregated value, this report will get some inflationary rewards, however if a more supported query gets submitted for with 100% of total stakes reporting for it, this will get 2x the rewards as the first value. There is no benefit being the only reporter for a query, unless of course there are tips that support that query over others. Additionally, to prevent tips on queries that only a handful of reporters support, time based rewards are only available on cycle list queries.

[3] There is also a delay (21 days) to exit (the unbonding period).

The goal is to get broad support among as many queries as possible without either incentivizing parties to report things that no one needs (e.g. report an obscure query that only I support) or disincentivizing reports of new queries. Tips will still, and should, be used as the primary incentive mechanism used to garner support for a given query.

Bridging

Updating the Validator Set

The bridge will be initialized with a starting validator set. Whenever the Layer validator set changes by 5%, all validators sign this new validator set checkpoint. This 5% threshold helps limit bridging costs as updating the validator set in the bridge contract costs gas. These signatures are used to update the bridge's record of the validator set, and any new data proofs will not pass the bridge's checks until the new validator set is relayed. Like data proofs, the bridge will accept a validator set update if at least ⅔ of the last known validator set signed off on it.

Tellor Token Bridge

The Tellor token will be interchangeable between the current ERC20 TRB contract on Ethereum and the base token TRB on Layer. It will be secured by a two way bridge, operated by Tellor layer itself via the trustless light client bridge. We will initially use a time-locked admin key to handle forks, but it will quickly move to just using the base layer bridge.

Bridging TRB from Ethereum to Layer will take 13 hours. Once the deposit is made, Tellor reporters will be able to report this information to Layer for an hour (to allow for a high level of finality on Ethereum). Once that one hour window is closed, the deposit can be claimed 12 hours later on Layer. A party bridging TRB to Layer for the first time will not have TRB available on Layer to pay the required gas fee to claim the deposit. However, they will be able to include a 'claim deposit' tip when they initiate a deposit on Ethereum to incentivize Layer participants to call the claim deposit function for them.

In order to prevent attacks, withdrawals from Layer can only be done if unstaked from being a validator on Layer (which requires 28 days). Once a withdrawal is initiated and attested to by validators, it can be retrieved on Ethereum 12 hours later. As an additional security measure, the bridge contract will not allow more than 20% of the total supply on Layer to be bridged within a 12 hour period (the function will be locked).

We have prioritized security over speed on the token bridge.

Data Bridge

To use Layer data, a light client bridge will be present on each chain and parties will relay the requested data and signatures as well as information related to validator set updates and chain upgrades. Relaying is a trustless role, as anyone can push validator signatures from layer to validate data on the bridge contract, however access controls can be put in place at the chain or user contract level.

Signing Prices

Once a value is aggregated (finalized in a given block), all validators sign the information. The resulting signatures will allow users to access the value, timestamp, aggregate power (amount of reporter stake used in aggregation), previous report timestamp, and next report timestamp of a given query. The previous report timestamp and next report timestamp by queryID are included for proving various properties of a given report.

Where: *data = value, timestamp, aggregatePower, previousReportTimestamp, nextReportTimestamp*

Validators will: *sign(queryId, data, validatorCheckpoint, attestationTimestamp)*

Parties using Layer can then grab the signatures and relay it to their chain.

Users can also request new signatures for data from previous blocks, (e.g. request(queryId, timestamp)) and the chain will return signatures on the older data, but with a newer attestationTimestamp. This is so parties can use values optimistically, by checking that the data has stayed on Layer for a certain amount of time without disputes, as data that has been disputed will not be signed again.

Relayer

The relayer (pushing signed data from layer to user chains) can be anyone, however we will provide software for relaying values and tipper scripts for setting up recurring feeds. Many keeper services also could fill this role (Keepr, Gelato, etc.).

Relay fallback

If there are no updates to a given light client bridge for >21 days (the unbonding period on Layer), then the bridge is considered stale, a situation where it needs a manual update of a validator set. In this case, we will set the team's address as a fallback that can update this list. If parties do not want it to fallback to the team, they can simply update the contract once every 21 days or deploy a light client bridge without this fallback (or where it falls back to a different address).

Other cross-chain methods

We know that some parties already have existing bridge solutions that they prefer. Layer data can be used via any of these solutions and we look forward to building and working with the teams to deliver the fastest and most secure solution for users. Some potential usage solutions include IBC, Hyperlane, Succinct, LayerZero, Chainlink, and many more.

In the future, it is likely that native or zero-knowledge bridges will be used to verify signatures, consensus, as well as inclusion of values. Tellor will be leaning on other teams currently specializing in cryptography research, but we fully expect that all bridges will be cheaper and faster using this method and should be operational within the next cycle.

Data Usage

Reporters are not required to provide data for all queries. This creates two types of data on-chain, *robust data* and *edge data*. Robust data is data that reaches consensus or support from ⅔ of the reporters and validators, while edge data can be any data that did not. While confidence on both types of data increases as time goes by (no disputes or forks in the case of robust data), they have to be used differently.

Robust Data Usage

If ⅔ of Tellor reporters sign off on a value, it can be consumed faster. A relayer will grab the desired data and signatures and push it to the consumer chain. Parties can then use the information in their protocol. There is no need to validate it any further, however waiting could be helpful, if only in extreme cases (e.g. large price moves), to check for forks, or widespread dispute conditions (e.g. a protocol or exchange failure/hack where api information may not properly reflect desired data specifications). In most cases however, this method will allow for updates as fast as the chain itself.

Robust data can be consumed instantly only if it is assumed that the Tellor Layer validator set is not compromised (signed off on bad data to the bridge or in the chain). However unlikely it is that the validator set would be compromised we advise implementing some precautions when using the data immediately. Users can run or whitelist relayers on their contracts and/or run monitoring tools, not allow immediate withdrawals, system freezes, etc. as necessary. See the **Additional Self-Driven Security and Fallbacks** section for more information.

Edge Data Usage

Similar to robust data usage, parties will request signatures and relay data from Layer. However, edge data should be handled optimistically, meaning that the user should allow time for a dispute to be raised and before using the data validate that the timestamp returned is within the

needed period of relevance (old enough to allow for disputes but fresh enough to use). They can either use an older value (similar to current Tellor), and/or verify that X% of validators signed off on it.

Additional data security

Dispute Monitoring

Dispute monitoring is beneficial for both robust and edge data. However, this is especially important when parties are using edge data; they must be cognizant that the system is only as secure as the monitoring for disputes. If, for instance, a party requests an obscure piece of data that only one validator reports for, and their stake is low, there will likely be few parties checking this query for disputes. This is why if using an edge value, adding extra security is essential. Running your own dispute monitor, educating more reporters to support it, or even more secure measures such as using only if the median is within x% of a given (e.g. their own team's) reporter.

Optimistic as Fallback

If consensus fails on certain values that are typically robust, parties have two choices: wait for consensus to return or handle the edge value, optimistically as specified above. This could be a great option for some parties using data where quality can quickly change. A price feed for example might not come to consensus in times of api failures or exchange manipulations (e.g. feeds go down). In this case, it might be best to pause the system, but they could also go with the optimistic approach if their protocol needs a faster (albeit still slower than consensus) price. Note that in most cases, an uncertain value would NOT be pushed to your protocol. Tellor is unique in that rather than forcing reporters/validators to sign off on an api or price feed, the addition of their value is optional. This means that if a reporter is not certain it's a valid value, they will likely just sit out that round(not risk their stake or part of it) and the value will not reach consensus; something that should be seen as a good thing in cases where ambiguity still exists (e.g. two exchanges differ wildly on price).

Additional Self-Driven Security and Fallbacks

Although security is at the center of Tellor's design, consumers of Tellor data can add custom additional security. One way is to simply limit who can push prices/ state updates on the consumer chain. By adding validation at this level, chains could use either their own validators or stakers to push over the prices after validating them. This would be an excellent option for users with this level of ownership over their protocol.

An option for non-chains via the low latency model would be to add validation before a trusted party (e.g. the app's dao) pushes the data. It gives the trusted party an option to censor, but

they would be unable to change what the price is, something that would work similarly to a multisig having pause authority/control over a protocol. Users can also do OEV limits this way (relayer is a known party or even auction off the right for OEV[4] each day/month).

Another option to increase security is consumer side pauses and delays. Pausing the system could be similar to the Maker design, where token holders of their own system can freeze the system in the case of a bad value.[5] For delays, parties could just design a system where it costs X dollars (a large amount) to delay the use of a reported value. While delayed, they could initiate a dispute on the Tellor system to remove a reporter or evaluate the situation. This would work well for systems that can handle delays (e.g. a prediction market delaying payouts). Just note that this would be on a per-user basis and custom as the cost to dispute/pause is also the cost to censor if it freezes the system for certain protocols.

Users can also add disputes directly in their protocol (similar to Tellor's disputing system, but can be handled by their governance). This would work well if coupled with custom staking requirements for reporters and could also be added as the "trusted" party that the Tellor value must match.

Fork Choice

Upgrades and forks to Layer will likely happen. Upgrades in the form of better data aggregation techniques, changes to the consensus protocol(e.g. cosmos sdk updates for faster blocktimes), or even changes to the Tellor system to make it more secure. Hard forks on the other hand are security based and happen if the protocol is attacked, compromised, or broken in some way.

Soft Upgrades

For upgrades, we can have the oracle itself report an upgrade. For each chain using the Tellor protocol, we can have a mapping of chainID to a valid contract address for the verification of the data/protocol. In order to change it, a new light client contract with the updated code will be deployed on the EVM chain. Validators will then update the mapping, and propose a change on the consumer chain. Then after waiting 14 days (allows for users to exit if malicious), the address will be updated to the new validator contract.

Hard Forks

For hard forks, you will also have a way to update this proxy address for verification of the consensus mechanism. The issue here is that time is of the essence. If the validator set is

[4] Oracle extractable value
[5] Depending on the type of data, this is a best practice for any system using an oracle

compromised or a bug is found, parties will want to very quickly switch off the oracle and upgrade. Unfortunately the switch cannot happen quickly, but freezing should be possible.[6]

Plan for legacy Tellor contracts

Tellor currently has users on Ethereum, Polygon, Gnosis, Arbitrum, Mantle, zkEVM, zkSync, BOB, and Optimism.

Tellor contracts are non-upgradeable, so users of the contracts can continue to use the contracts as they do currently. Users will need to continue to incentivize TRB token holders to stay on other chains posting data (tip), and should migrate over time as they upgrade their contracts or launch new systems.

The current TRB token contract will remain the same. But instead of minting the time based rewards to the oracle contract it will send these rewards to the bridge contract. The bridge contract will also have an oracle proxy that allows users to read data from the Layer (the light client contract) and gives the team the ability to vote through it.

The reason the oracle tokens are given to the bridge is to allow a two way bridge since inflationary rewards would be happening on Layer based on the bridged tokens.

Glossary

1. **Tellor Layer:** A standalone Layer 1 (L1) blockchain built using the Cosmos SDK, designed for consensus on subjective data using tendermint and an optimistic approach.
2. **Cosmos SDK**: A software development kit for building blockchains in the Cosmos ecosystem.
3. **Tendermint/ Comet BFT:** A Byzantine Fault Tolerant (BFT) consensus algorithm used by blockchains in the Cosmos network.
4. **Validator:** An entity in the Tellor Layer network that has locked a certain amount of TRB tokens to participate in block validation.
5. **Reporter:** An entity in the Tellor Layer network responsible for submitting values for data requests (queryIDs). Reporters can be staked validators or other participants, depending on the network's rules. They provide data, incentivized through tips, subject to validation and potential disputes.
6. **Query:** A unique identifier for a data request (e.g., BTC/USD price) on the Tellor Layer.
7. **TRB Token:** The native cryptocurrency of Tellor Layer, used for staking, tipping, and governance.
8. **Cycle List:** A governance-controlled list of queryIDs to maintain data freshness and ensure continuous reporting.

---

[6] A library such as: https://github.com/RealityETH/subjectivocracy can be used as a dispute resolution mechanism for freezing and then voting on the results of the fork (all very costly to initiate to prevent censoring). Ultimately, you want this to be decided by the users on the chain itself and it might be a good choice to have a permissioned set (w/ a high cost still) or even a re-staking situation.

9. **Consensus Threshold:** The minimum amount of agreement required among validators for a value to be considered valid.
10. **Disputing**: A process in Tellor Layer where reporters' submissions can be challenged, potentially leading to slashing of their stake.
11. **Slashing Event:** A penalty where a portion of a validator or reporter's stake is removed due to submission of incorrect data or other violations.
12. **Tipping:** A system where users can tip reporters in TRB tokens for submitting data for specific queryIDs.
13. **Relayer:** An entity that transmits data from the Tellor Layer to other blockchains, facilitating cross-chain data accessibility and usage
14. **Robust Data Usage:** A data verification method in Tellor Layer where values agreed upon by a supermajority of reporters are instantly deemed reliable for real-time use.
15. **Edge Data Usage:** A method of using Tellor where submitted values are presumed correct if not disputed over a period of time, and/or X% of validators signed off.
16. **Light Client Bridge:** A mechanism for relaying Tellor Layer data to other blockchains.
17. **Validator Nonce:** A unique number used once by validators to prevent replay attacks.