GUMMIWORM

PI LANNINGHAM

ABSTRACT. Gummiworm is a layer 2 for Cardano being built by Sundae Labs.

Cardano optimizes for robust, global, permissionless decentralized consensus with very lower power requirements, with practical settlement times measured in tens of minutes, and total peak throughput (currently) measured in the low 10s of transactions per second.

Hydra, on the other hand, optimizes for high throughput and instant settlement transactions among a small group of unanimously consenting parties, with a robust and trustless conflict resolution mechanism.

Gummiworm is a protocol that tries to recognize that there are many different dimensions on which decentralization can matter, and separating those can provide a robust, practical, **and** decentralized Layer 2 for Cardano.

Additionally, it is built as a low-divergence fork of the Hydra node. This lets it reuse and benefit from the continued development of the Hydra protocol, and avoids "throwing the baby out with the bath water" in a need to reinvent everything from scratch.

CONTENTS

1. Overview

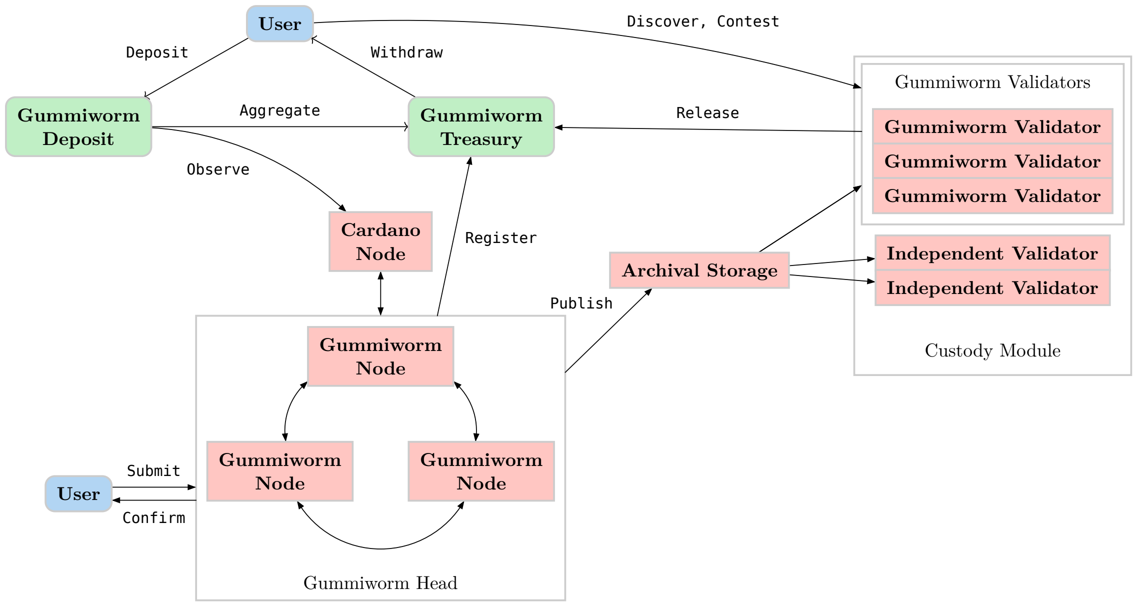
This specification will provide a high fidelity specification for the major concerns of the Gummiworm protocol.

Note

This document is very much a work in progress, and several sections have been left blank pending further details.

Additionally, many details may change or be made more precise in response to reviewer feedback.

At 50,000 feet, the protocol can be described by the diagram below:



A set of modified Hydra nodes run as a cluster, known as a Gummiworm Head, register by depositing collateral into the Gummiworm Treasury. Rather than using the native Hydra capabilities for committing funds, users deposit funds through a time-lock contract, which is then aggregated into the treasury. As the Gummiworm Head observes these deposits, it fabricates these tokens in its ledger.

Users submit transactions to the Gummiworm Head, which returns a signed receipt for the transaction. At regular intervals, the Gummiworm Head publishes its transactions to an archival storage mechanism.

A set of nodes, known as Gummiworm Validators, download these archives and independently validate them. Exactly how this set is selected is configurable, and allows for a number of different deployments based on the real world requirements of the work.

If a Gummiworm Head is found to have accepted an invalid transaction, or if a user presents a transaction receipt that doesn't appear in the transaction stream in a reasonable time, the Gummiworm Validators can seize some or all of the collateral of a Gummiworm Head to pay the impacted users for the inconvenience.

To ensure a Gummiworm Head is in good standing, users monitor messages published by the Gummiworm validator set before making a deposit or a transaction. The Gummiworm Validators can also apply back-pressure if the Gummiworm Head is too far ahead of the validators.

Note

This document describes Gummiworm v0, an initial version of the protocol that makes certain compromises. In particular, it uses a simplified custody module that accepts more centralization than future versions will. This will allow us to build and harden a production deployment of the protocol, and find real world clients that can benefit from the increased throughput and tolerate the tradeoffs.

2. Motivations

Cardano is optimized for permissionless and decentralized consensus. For a few hundred dollars a month in infrastructure, anyone can run a node and potentially participate in contributing blocks to the network. So long as 51% of the network is honest, by weight of delegation, the network remains secure against a number of adversarial attacks.

It pays for this with a lower throughput and slower finality than chains that make different tradeoffs. It relies on relatively “quiet” periods between blocks, to minimize the chance of network forks.

Conversely, Hydra is an isomorphic state channel protocol. A small group of people lock their funds into a smart contract, which can be released by presenting a signed snapshot of the UTxO state. Off-chain, they can use a relatively simple unanimous consensus to share transactions, and then sign new snapshots. If there is a dispute, they can present one of these snapshots to the layer 1 contract. After a short timeout (in which another participant can present a newer snapshot), the contract releases and fans out the funds according to that final snapshot.

This achieves throughputs in the hundreds of transactions per second, with instant finality. However, its security relies on every participant signing each snapshot. While this holds, the protocol enjoys a unanimous consent security, where a single honest party disallows a falsified transaction.

However, as soon as there are participants that do not sign the transaction (such as transactions spending funds held at third party addresses), this security property becomes much more dangerous. Hydra cannot be used for a decentralized exchange, for example, because funds belonging to potentially tens of thousands of users would be at the mercy of a relatively small number of head participants. They could, for example, sign a snapshot that asserted that all funds had been paid directly to them, and then unlock the layer 1.

The key insight behind Gummiworm is recognizing that there are multiple dimensions on which you can measure the decentralization. In particular:

- Execution: The ability to select **which** transactions get included, the ability to censor transactions, the ability to reorder transactions, and the ability to validate (or not validate) transactions.
- Custody: The long-term ability to **spend** funds and extract real world value from them.

Gummiworm is a protocol that recognizes that, for the sake of throughput, users are often willing to compromise on the first dimension, but are much less willing to compromise on the second.

It leverages the Hydra peer to peer consensus for execution, meaning a small set of participants decide which transactions to include, what order to serialize them in, and assert to their short-term validity.

However, they have **no** direct custody of user funds. Instead, they must demonstrate to a separate custody module that there exists a sequence of valid transactions that leads to the requested withdrawals. This allows the Gummiworm Head to race ahead in bursts, with the tradeoff that withdrawals may take longer while the custody module catches up.

3. Actors

In this section, we enumerate the various actors, software components, and smart contracts involved, and the actions they can perform.

3.1. **Human Actors.**

1. **User** - A standard, unprivileged user that wishes to use various dApps on the Gummiworm Protocol.
    - Honest Actions
        1. Discover - Retrieve a list of Gummiworm Heads and their status
        2. Deposit - Initiate a deposit into a specific Gummiworm head
        3. Cancel Deposit - Cancel a deposit that never got claimed
        4. Submit Transaction - Submit a valid transaction to a Gummiworm Head in good stead
        5. Withdraw - Initiate a withdrawal from a Gummiworm Head
        6. Hard Withdraw - Request a withdrawal directly from the validators, in cases where the Gummiworm Head ignores the request
        7. Contest - Present a transaction receipt to the validators that never made it into the transaction stream
    - Malicious Actions
        1. Submit Invalid Transaction - Attempt to submit a malformed transaction to the Gummiworm Head
        2. False Contestation - Present a fake transaction receipt to the validator set
        3. Supurfluous Hard Withdrawal - Request a hard withdrawal, even when the Gummiworm Head is behaving honestly
2. **Gummiworm Head Operator** - Someone who, along with a small cohort of other operators, runs a Gummiworm Head and receives transactions from users.
    - Honest Actions
        1. Open Head - Open a new Gummiworm Head
        2. Increase Collateral - Add new collateral into the head
        3. Withdraw Collateral - Withdraw some collateral from the head
        4. Close Head - Close a Gummiworm Head completely
        5. Receive Transaction - Receive a transaction from a user, validate it, and return a receipt
        6. Publish Archive - Publish an archive containing many transactions to shared storage
    - Malicious Actions
        1. Censor Transaction - Refuse to respond to a user submitting transactions

2. Refuse Receipt - Refuse to return a signed transaction receipt to a user, but include the transaction
3. Invalid Receipt - Return an invalid receipt for a transaction
4. Drop Transaction - Don't include a transaction in the archive for which we've issued a receipt
5. Ignore Deposit - Don't honor a deposit requested by a user
6. Publish Invalid Archive - Publish an archive that is invalid, or contains an invalid transaction
7. Stop Publishing Archives - Fail to publish the next archive in a timely manner
8. Split History - Maintain a separate history between the validators and the users

3. **Gummiworm Validator** - One of many participants who download and validate the archives published by the Gummiworm Heads, and control custody of the Gummiworm Treasury.
   - Honest Actions
     1. Aggregate Deposits - Aggregate many deposited tokens into the Gummiworm Treasury
     2. Disperse Withdrawals - Disperse requested withdrawals to the users who requested them
     3. Respond to Contestation - Check a users contest, and if valid, disburse a small amount of collateral
     4. Report Status - Report which heads are in good status, and apply backpressure on transactions if needed
   - Malicious Actions
     1. Steal Treasury - Steal funds from the treasury
     2. Go Offline - Stop honoring Withdrawals or Contestation
     3. Hide Gummiworm Heads - Hide open gummiworm heads to deny them users
     4. False Backpressure - Report false backpressure, causing users to face delays
     5. Skip Validation - Rubber-stamp transaction archives without validating them

3.2. Software Components.

1. **Gummiworm Node** - Receive and Validate Transactions, issue confirmation receipts, monitor for deposit requests, and publish transaction archives. Run by the Gummiworm Head Operator.
2. **Gummiworm Validator** - Download and Validate transaction archives, maintain consensus with other Gummiworm Validators, sign transactions from the treasury.
3. **Wallet** - Provide an interface for users to query Gummiworm Heads and submit transactions.

4. **DApp (Decentralized Application)** - Build and submit transactions, query Gummiworm Head state through the wallet, Query Gummiworm Head status from the validators, honor backpressure.
5. **Gummiworm DApp** - One such DApp that focuses on deposits, withdrawals, and status reporting.

3.3. Smart Contracts.

- **Gummiworm Deposit** - Hold a pending deposit. Allow a user to cancel the deposit after a timeout. Allow the Gummiworm Validators to aggregate it into the treasury once its been confirmed in the head.
- **Gummiworm Treasury** - Hold the aggregated deposits, and the Gummiworm Head collateral. Allow spending with a signature from the Gummiworm Validators.

4. User Flows

4.1. Deposits.

A user can initiate a deposit by paying funds into the `Gummiworm Deposit` contract.

To do so, the user first looks up a Gummiworm Protocol UTxO holding a Gummiworm Protocol Instance tracking token. In the datum of this UTxO will be an API endpoint to contact the current Gummiworm Validator Cohort.

The user then queries the Gummiworm Validator Cohort for the current status of the Gummiworm Head they wish to deposit to. This will return whether the head is in good standing, the current collateral multiple, and the current lag time from processing this head. This payload will contain a signature from the current cohort.

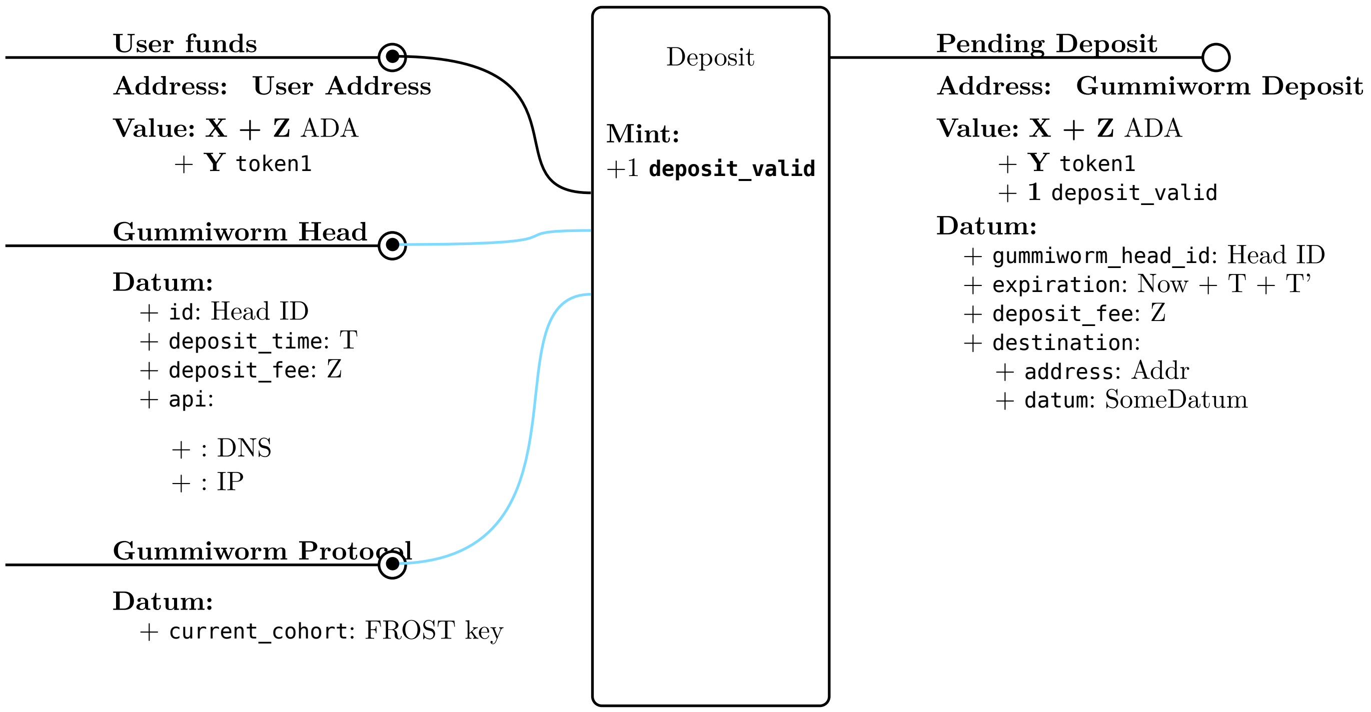
The user should check the following conditions before initiating a deposit:

- The Head is in good standing; A head that has been penalized should not receive additional deposits.
- The Head has a sufficient collateral multiple; The amount of funds that are safe to deposit into the head is a function of the collateral they have locked.
- The Lag time is a comfortable threshold for the user. A lag time too large can indicate that there is trouble with a head, and should be a source of backpressure.

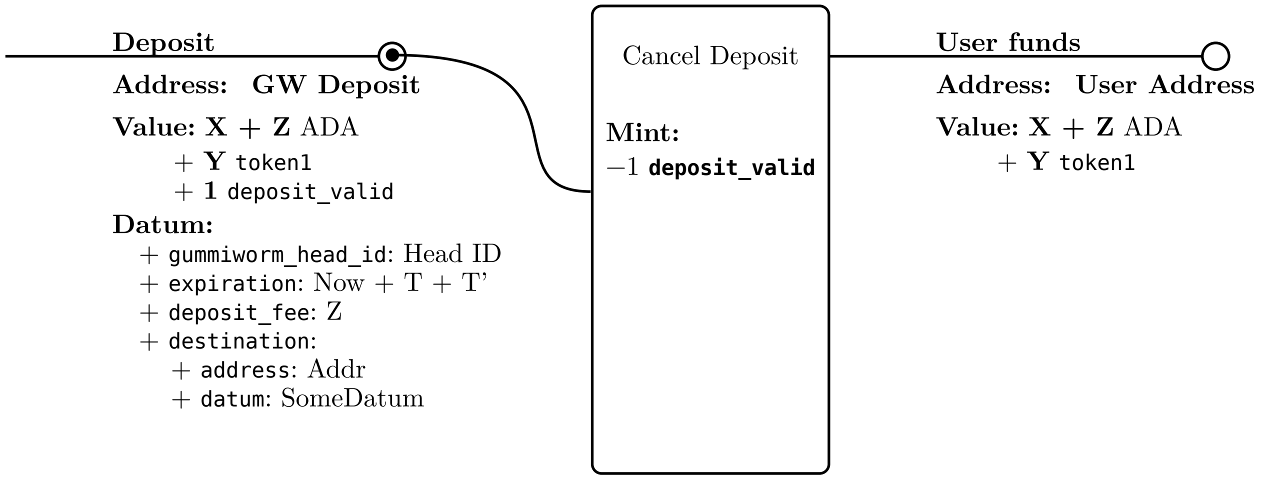
> **Note**
>
> Some of these conditions may be enforced by the deposit_valid minting script. Exactly how to determine these thresholds is still an open question, and one we'd love feedback on.

The user then builds and submits a transaction to the Cardano L1 containing the funds they wish to deposit, and signalling to the Gummiworm Head Operator that a deposit should be performed within the deposit time `T`.

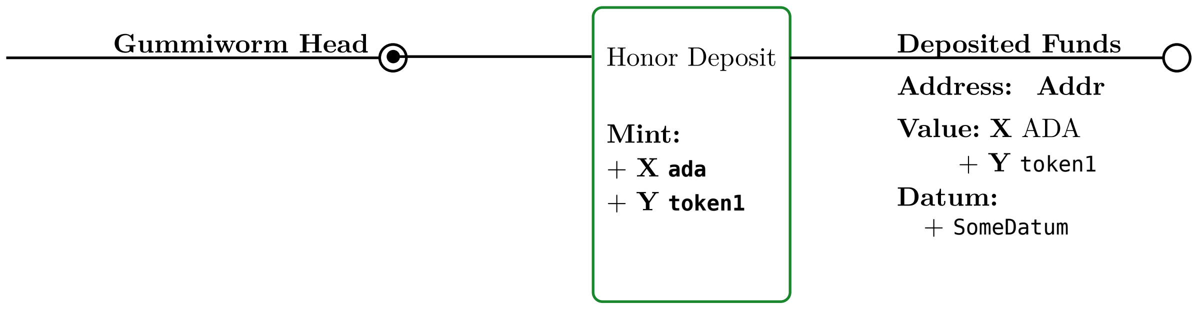
The user mints a `deposit_valid` token to ensure that the expiration times are set correctly, with reference to the Gummiworm Head's commitment time. The minting policy also checks that the redeemer contains `T'` with a signature from the current cohort.



If no action is performed, the user can reclaim these funds after `Now + T + T'` expiration time, where `T` is the Gummiworm Head's commitment time, and `T'` is a protocol parameter to capture the current Gummiworm Head's lag time.



The Gummiworm Head determines its own policy for finality. After seeing the deposit transaction, depending on the quantity of funds and its own confidence in the transaction finality, it can mint the appropriate tokens on the L2, paid to the `Destination` from the deposit datum. The metadatum of this transaction signals the Layer 1 UTxO it is honoring. One transaction can honor many deposits as separate UTxOs.

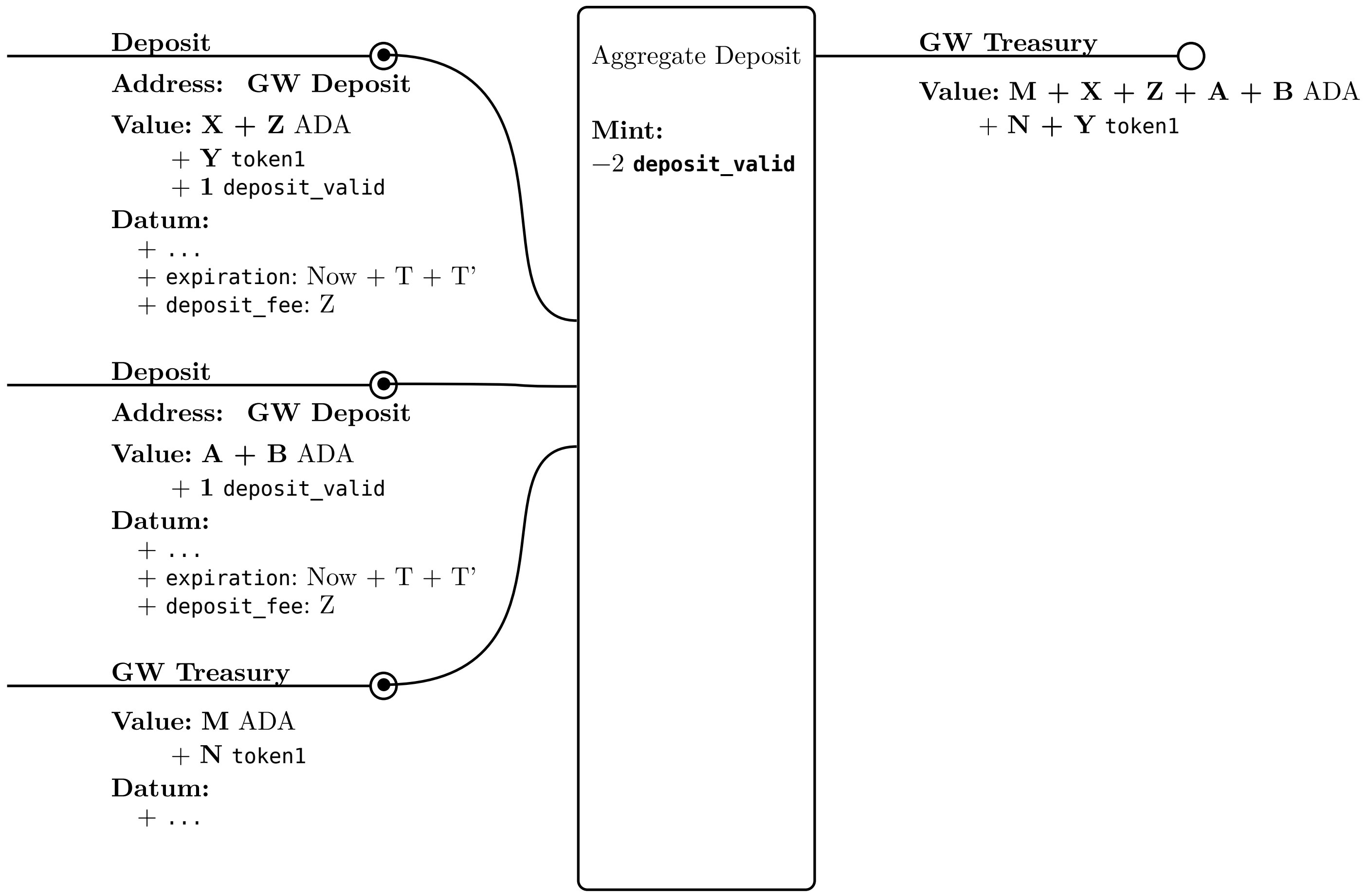


Note: Deposits: [TxHash#Idx]

Note

A future version of the protocol may extend the Gummiworm ledger model to add a `fabricate` and `destroy` field to the transaction body. This would allow a permissioned set of nodes to create or destroy tokens without running the minting policy, to honor deposits and withdrawals without creating a "wrapped" token.

Once this transaction is confirmed, and the transaction appears in the Gummiworm Archive (paid to the correct destination), the Gummiworm Validators can aggregate a batch of deposits directly into the Gummiworm Treasury, to prevent the user from cancelling the deposit after extracting value from those tokens within the Gummiworm Head.



4.2. **Transactions.**

This section outlines the lifecycle of a transaction within the Gummiworm protocol.

4.2.1. *Submission.*

After initiating a deposit, a user can begin interacting with the Gummiworm Head.

There will be a UTxO for each Gummiworm Head, containing a `gummiworm_head` authenticating token. The datum on this UTxO will contain one or more API endpoints that can be used to interact with the Gummiworm Head.

> ⓘ **Note**
>
> We would like to standardize the query protocol for these formats across all layer two projects. We are closely watching proposals like CPS-0012 to see if a proposal emerges.

> ⓘ **Note**
>
> We would like querying the layer 2 to be integrated with the wallets as well, as a CIP-30 extension. A CIP proposing how to signal to the wallet the extra layer-2′s that the user wishes to connect to on connect will be forthcoming.

Using those API endpoints, the user (or the dApp / wallet on their behalf) can query the status of their deposit, and any UTxOs available for building transactions.

The user should periodically check the status of the Gummiworm Head, to ensure it stays in good standing, and that the backlog isn't growing. If a head is flagged in bad standing, or the lag time grows outside of a tolerable window, the user should cease transacting with the head immediately. Exactly how often to check, or what those thresholds are, is left to the users discretion.

If the user is satisfied with the status of the head, they can build and submit transactions directly to the Gummiworm Head as if it were a standard Cardano submit API.

4.2.2. *Receipts.*

On receipt of a transaction, the Gummiworm Head should follow the underlying Hydra protocol to validate and circulate the transaction among its peers. Once all peers have seen the transaction, and signed the next snapshot, this signed snapshot, along with a proof that the transaction is present in the snapshot, should be returned to the user in the form of a "transaction receipt".

This receipt should largely follow the format of the signed Hydra snapshots, and should at least include:

- The gummiworm head ID
- The users transaction ID (or a merkle root that can be used to prove the transaction is included)
- A monotonically increasing "snapshot number"

This receipt gives the user confidence that the transaction will eventually be seen by the Gummiworm Validators. If the user presents this receipt to the Gummiworm Validators and proves that the transaction was never included in the final transaction stream, a portion of the Gummiworm Validators collateral will be slashed for the inconvenience.

> ⓘ **Note**
>
> The person submitting a transaction is ultimately just one party who cares about the finality receipt; but the user **receiving** the funds also cares, as they're about to make decisions based on having those funds. We should consider mechanisms where that finality receipt can be served to the recipient when they learn about the transaction as well.

For exactly how this process works, see the section on Contesting with Receipts.

> ⓘ **Note**
>
> The exact amount of punitive pressure to apply to Gummiworm Heads is an open question, and one we invite feedback on. Perhaps there is a way to express this as the settings of the Gummiworm head, or perhaps this is sensitive enough to flag the Gummiworm head in bad standing and slash their entire collateral.

4.2.3. *Publishing.*

At regular intervals, the Gummiworm Head should package a large batch of transactions into an archive format described below, and publish them to a shared storage mechanism.

> ⓘ **Note**
>
> The exact storage mechanism is flexible, and may even vary from head to head depending on the use case, but should at a minimum be accessible to the Gummiworm Head and the Gummiworm Validators. Some examples might include a private S3 bucket, a data availability layer, a public IPFS pinned file, or Iagon storage.

The exact frequency of these archives is a parameter of the Gummiworm Head, and consists of 3 limits. When the prospective archive hits any of these three limits, it should be closed off, published, and a new archive begun:

- A time interval; for example, every 30 minutes
- A transaction count; for example every 1,000,000 transactions
- A byte count; for example, every 10 gigabytes

This ensures that each archive is timely, and reasonably sized.

The Gummiworm Validator set will then download this archive and validate each transaction, collecting up a list of withdrawals.

If an archive is all valid, and contains no withdrawals, there is nothing else for the Gummiworm Validator set to do.

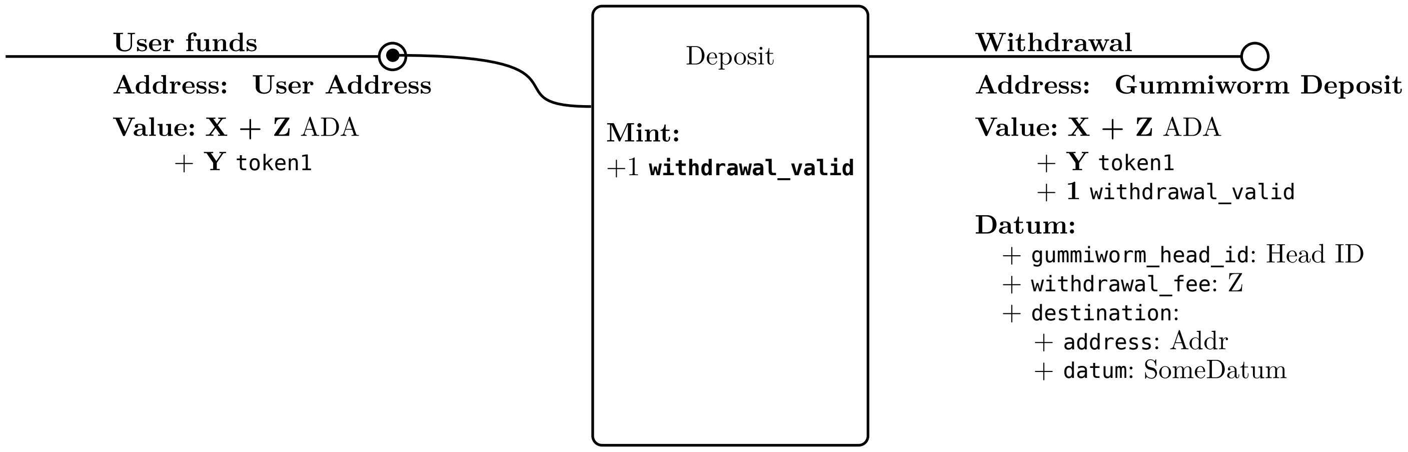
If there is an invalid transaction, or at least one withdrawal, this will initiate the appropriate process for the Gummiworm Validator set. They will engage in a small local consensus to produce an aggregate FROST signature on the withdrawals or dispute, and publish a transaction on the Cardano Layer 1. For exactly how these processes work, see the later sections describing these flows.

4.3. **Withdrawals.**

In order to withdraw funds from a Gummiworm Head, there are two processes that a user can follow. One is the optimistic path, when a Gummiworm Head is operating correctly, and the other is a pessemistic path, for when a Gummiworm Head is misbehaving or offline.

In this section, we will cover the first, and leave the second for the section dealing with all manner of protocol Abuse.

A user can request an optimistic withdrawal on the Layer 2 by submitting a transaction that pays the funds they wish to withdraw to a new UTxO locked by the Gummiworm Deposit contract, with a `null` Gummiworm Head ID. This contract will allow the Gummiworm operator to spend and burn the wrapped assets.



Because the Gummiworm Head is instant finality, and there is no hard custody of user funds, there is no timeout for the withdrawal.

While processing an archive from the transaction stream, Gummiworm Validators should track all withdrawals they observe.

Once the end of that archive is reached, a Gummiworm Validator should:

- If no messages have been received from peer Validators
    - Build a transaction that spends the Gummiworm Treasury, and pays out the requested withdrawals to the requested destinations
    - Broadcast this transaction, along with a round number, to the peer Validators requesting a commitment for a FROST signature
- If a messages has been received from a peer with a proposed transaction that matches the observed withdrawals, respond with a commitment to sign this transaction and mark yourself as having voted for this (archiveID, roundNumber)
    - If you've already voted in a round, refuse to vote again
    - If you've already seen a transaction and the archive is currently marked as complete, refuse to vote again
- If a quorum of votes is reached, aggregate the signatures and broadcast the transaction to the Cardano Layer 1

- If such a transaction is seen, mark that archive as completed; if the transaction is rolled back, mark it as incomplete
- If a timeout occurs and the archive is not still marked as completed, build a new withdrawal transaction and broadcast it to your peers with an incremented round number.

Note

The above is a simple RAFT like protocol for aggregating FROST signatures.

The exact details may need to be refined as it is implemented and tested heavily.

Additionally, we may wish to replace this with a more adversarially safe consensus, such as a proof of stake weighted protocol.

5. Abuse

5.1. Contesting with Receipts.

When submitting each transaction, the submit API for the Gummiworm Head should hold the request open until it can respond with a signed finality receipt. This receipt has a signature from each of the Gummiworm Head operators, and is a commitment to include the transaction at a specific height in the transaction stream.

If the archive delivered to the Gummiworm Validators ultimately does not include this transaction, the receipt can be used as evidence to slash a portion of the head operators collateral for censoring the transaction.

In principle, a user can do this themselves by downloading the appropriate archive to check that the transaction was included. In practice, most users will not do this.

For this reason, the Independent Gummiworm Validators consist of anyone willing to run a Gummiworm Validator node. They don't participate in the custody of the funds, but can receive **all** receipts, and then check them for the user. If they find a discrepancy, they submit them to the Gummiworm Validators on behalf of the user for a small portion of the collateral.

> ⓘ **Note**
>
> The exact incentives of how much collateral is at risk, and whether to halt the Gummiworm head in this case, are still under consideration. We would gladly welcome feedback on this.

5.2. Withdrawal Request.

In cases where the Gummiworm Head is not honoring withdrawal requests, users can submit the signed withdrawal transaction to the Gummiworm Validators directly.

One of the Gummiworm Validators will submit this transaction to the Gummiworm Head directly, to confirm that the Gummiworm Head is rejecting it. If so, it will initiate the dispute resolution process among the Gummiworm Validators to flag the head as in bad standing, and begin unwinding the Gummiworm state as of the last known good state.

6. Security

The security argument for this protocol rests on measuring the nakamoto coefficient of the custody of the user funds.

The Gummiworm Validator Set relies on a M of N FROST key. Therefore, so long as M nodes remain available, and $(N - M) + 1$ nodes remain honest, the following properties hold:

- A user who deposits their funds, but whose deposit is not honored by the Gummiworm Head within the time limit, will be able to reclaim their deposit before it is aggregated by the Gummiworm Validators

- A user who deposits their funds, and whose deposit **is** honored by the Gummiworm Head, will not have their funds spent by the Gummiworm Head without a signature from their wallet
- A user who receives funds on the Gummiworm Layer 2 in a transaction that makes it into the transaction archives, and doesn't otherwise spend them in a later transaction, will eventually be able to withdraw those funds back to their address on the Cardano Layer 1
- A user who submits a transaction to the Gummiworm Head and receives a receipt, will either see that transaction in the eventual transaction stream and (assuming they don't spend the funds first), be able to withdraw them; or receive a portion of the Gummiworm Head's collateral as a punitive measure
- The Gummiworm Head operator will be unable to forge transaction signatures, or smart contract outcomes, without forfeiting their collateral
- The Gummiworm Head operator will never be able to spend user funds on the Cardano Layer 1

Sundae Labs
Email address: pi@sundae.fi