ar.io

Network + Token White Paper

v1.0.0 – February 01, 2024

TABLE OF CONTENTS

CONTRIBUTORS

The Foundation is grateful for the contributions of the following individuals, who generously provided their time, expertise, and insights to the development of this paper and design of the network. Their contributions helped to shape the content and ideas presented here. Each contributor brought unique skills and perspectives to the project, and their efforts were instrumental in creating a comprehensive and informative resource for the AR.IO Network.

- **Philip Mataras**, CEO at Permanent Data Solutions, Inc.
- **Jonathan Policke**, COO at Permanent Data Solutions, Inc.
- **Ariel Melendez**, CTO at Permanent Data Solutions, Inc.
- **David Whittington**, Senior Architect at Permanent Data Solutions, Inc.
- **Dylan Fiedler,** Senior Engineer at Permanent Data Solutions, Inc
- **Tim Bukher**, Strategic Advisor and CLO at Permanent Data Solutions, Inc.

In addition, the Foundation warmly thanks everyone who provided invaluable input and "fresh eyes" during the draft phases of this paper's development. Their contributions have significantly enhanced its quality and the network's design; their ongoing support and collaboration are highly appreciated.

1 INTRODUCTION

1.1 WHAT IS AR.IO?

AR.IO is a global network, protocol, and currency that enables the permaweb: the files, apps, webpages, and data permanently stored on the Arweave decentralized storage network.

The AR.IO Network is an open, ownerless, distributed network, with participation from operators, developers, and end users from around the world. The various nodes on the network, known as AR.IO Gateways, form a decentralized interface between users and the permaweb. Each gateway acts like a "Permaweb Service Provider" and supports the critical web services that apps need like reading, writing, querying, and indexing of Arweave data.

These gateways follow various network protocols to provide a uniform experience for developers and end users. In addition, each gateway is tasked with assessing the performance of their peers by participating in an observation and reporting protocol aimed to keep the network healthy and keep gateway operators accountable.

The AR.IO Network leverages a utility token to further proliferate access to and services for the permaweb. This token, referred to as IO or ϕ, is leveraged for various network utilities and incentives. This includes being a currency for services like the Arweave Name System (ArNS), stake used to join a gateway to the network and distributed as an incentive for performing duties as a gateway.

1.2 WHY AR.IO?

Arweave (a Layer 1 blockchain network) offers scalable and permanent on-chain data storage in a sustainable manner. It does this by incentivizing miner nodes through a tokenomic endowment model which ensures data is globally stored and replicated for hundreds of years without the need for continual payment or maintenance by its uploader.

However, the Arweave protocol does not incorporate all the needs of permaweb applications like data indexing, querying, retrieval, and other vital services. Consequently, over the past few years, infrastructure services have been independently developed and deployed to meet the demands of the permaweb at scale. Users and apps have come to rely on these gateway utilities, but they are closed source, have complex codebases, and are expensive to operate.

Arweave does not offer any tokenomic incentives to offset the expenses associated with operating a gateway, which has led to the community's reliance on a single centrally controlled gateway subsidized for the betterment of the network: arweave.net. While arweave.net currently caches and indexes the entire weave with a high quality of service, it is a single bottleneck and point of failure for the whole ecosystem.

AR.IO seeks to reduce the barriers of entry and attract more gateway operators to the permaweb with the goal of further enhancing its overall health, resiliency, and functionality through decentralized mechanisms that are as trustless as possible.

The solution will be applied in two directions:

1. By reducing gateway overhead costs with open source, efficient, modular networked architecture.
2. By creating an economic incentive layer with the IO Token.

The overall goal of this white paper is to present the framework for a healthy and sustainable decentralized gateway network.

1.3 TL;DR

This white paper details a decentralized and incentivized gateway network aimed at attracting more gateways to the Arweave network and at making the permaweb more accessible to all. At the core of AR.IO's incentivization mechanism is the IO Token, a utility token used for joining the network, payments, gateway accountability, and protocol incentives. The network features modular and composable gateway infrastructure in addition to the Arweave Name System (ArNS) – a system for assigning friendly domain names to permaweb data.

Details pertaining to gateway configuration, the token contract, and specific settings or values can be found in the supporting technical documentation, codebase repositories, and smart contract state. *Note that any value settings presented in this paper are initial values only, subject to change through network testing and development.*

1.4 PAPER OUTLINE

The remaining sections of this document have been organized as follows:

- **Section 2:** Outlines **'The Foundation'** and its role in fostering the development of The AR.IO Network and ecosystem.
- **Section 3:** Provides background on the **Arweave** network, the **Permaweb**, and the technical components that form the base layers of AR.IO.
- **Section 4:** Describes the AR.IO **smart contract**.
- **Section 5:** Introduces the **IO Token**, its functions, and its tokenomics.
- **Section 6:** Describes the role of **staking** and the various **user interactions** participants have with the network.
- **Section 7:** Details the components and functions of an AR.IO **gateway**.
- **Section 8:** Details the structure and function of the gateway **network**.
- **Section 9:** Describes the **Arweave Name System (ArNS)** and how it bridges Arweave to a friendly user experience.
- **Section 10:** Overviews the network's **observation and incentive** structure and how it promotes gateways to adhere to network standards and maintain a high level of quality and reliability.
- **Section 11: Concludes** the paper and outlays the **future vision** of the network.
- **Section 12:** Contains a **glossary** and other **appendix** information.

2 THE FOUNDATION

2.1 OVERVIEW

This section is intended as a brief overview of The AR.IO Foundation, formally incorporated as Stichting AR.IO, a Dutch non-profit foundation. The creation and content of this white paper are attributed to the Foundation. The Foundation's complete constitution and mandate will be released separately from this paper.

The AR.IO Foundation is dedicated to the stewardship and prosperity of The AR.IO Network and its associated token ecosystem. It holds a non-revocable, exclusive license to promote the development of the network, prioritizing the ecosystem's wellbeing, particularly the users.

Key strategies employed by the Foundation (with the assistance of third-party teams) in support of the network include:

- Providing grants and incentive programs
- Making strategic investments
- Engaging in direct software development
- Producing educational content
- Conducting publicity and marketing initiatives
- Forming partnerships

The Foundation is distinct from Permanent Data Solutions, Inc. (PDS), a for-profit software developer based in the United States and the grantor of the network license. PDS is responsible for developing The AR.IO Network and continues to independently create software solutions that interact with both the Arweave and AR.IO networks. While PDS contributes to the network's development, the Foundation is exclusively tasked with advocating for the network's health.

2.2 GUIDING PHILOSOPHY

The Foundation is meant to serve as a rallying point for the ecosystem to communicate, innovate, and further progress. It avoids exerting undue influence over the network's economic mechanisms. The Foundation adheres to the following principles:

1. **Transparency:** The Foundation commits to regular reporting, detailing activities such as IO token sales, employee remuneration, expenditure, investments, and grants.

2. **Right to dissolve:** The Foundation has the right to dissolve itself if desired. This is achieved by enshrining in the bylaws of the Foundation's legal structure a stipulation that it must be disbanded if a majority vote is reached.

In summary, the AR.IO Foundation functions as the steward of the network, balancing its growth and development with a commitment to the community's needs and the network's foundational ethos. Its approach allows for adaptability while ensuring the ecosystem's stability and integrity.

It should be emphasized that the AR.IO Smart Contract contains immutable protocols without governance or special write access for its developers or the Foundation. This means that upgrades to the protocol can only be made through a fork which, in order to be successful, requires that the fork be advocated for, and consensus be achieved within the AR.IO ecosystem and amongst gateway operators. Details of this will be elaborated on throughout the paper.

3 ARWEAVE AND THE PERMAWEB

3.1 THE PERMANENCE PIE

The permanent data storage ecosystem can be thought of as a three-tiered arrangement of protocols, services, and applications – dubbed here as "The Permanence Pie".

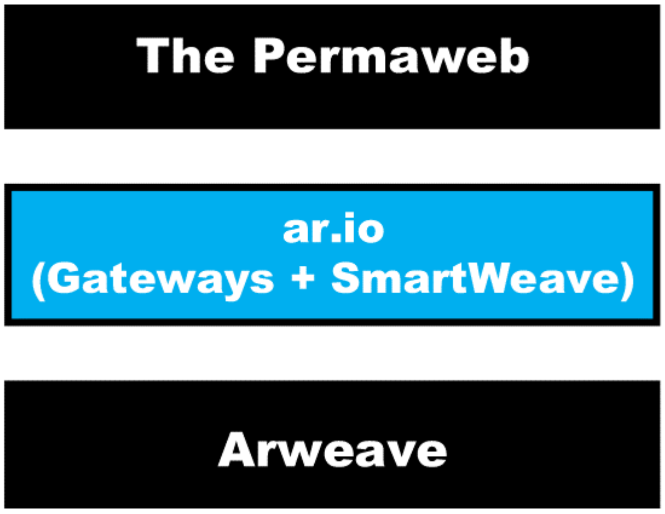

Diagram 3.1: *The Permanence Pie*

The base layer of that pie is the Arweave protocol and network, which is the backbone of the permanent data storage ecosystem. It provides the infrastructure for data to be stored on the network in a decentralized manner and incentivizes nodes to keep the data stored for long periods of time.

The second layer is made up of services that sit on top of the Arweave protocol and network. These services include gateways, data retrieval services, and smart contracts that help to provide a seamless and functional experience for users, creators, and developers.

Finally, the top layer of the pie consists of applications that utilize the data stored on the Arweave network. This includes everything from simple applications that allow users to access and view their data to complex, decentralized applications that use the Arweave network as their backbone.

Each layer of the Permanence Pie is crucial to the overall success and growth of the permanent data storage ecosystem. The Arweave protocol and network provide the foundation for data storage, the services layer helps to facilitate data retrieval and usage, and the application layer brings the benefits of the ecosystem to users and developers alike.

3.2 WHAT IS ARWEAVE?

Arweave is a decentralized Layer 1 data storage protocol optimized for long-term permanent storage through its unique proof of access mechanism and tokenomic endowment model.

The information stored on Arweave is immutable and globally replicated by miner nodes. Instead of a traditional blockchain ledger which links blocks of transactions together in linear sequence, Arweave arranges blocks in a web known as the blockweave. These miner nodes secure the blockweave by operating the Succinct Proof of Random Access (SPoRA) algorithm. SPoRA requires miners to prove that they have access to recall randomly selected bits of weave data in order to produce and share a block. If successful, miners are rewarded in Arweave's native AR token. These token rewards are derived from transaction fees as well as the network's storage endowment. The endowment is a protocol-controlled pool of tokens designed to ensure adequately replicated and fund the projected cost of storage for 200+ years.

Arweave is file type agnostic – any type of file ranging from simple text files to family photos to complex web applications and archival databases can be stored on the network. To upload data, users must pay an amount of AR proportional to the size of the files being uploaded. Arweave is unique when compared with other decentralized storage solutions in that users only pay once per data / file upload, then that is it – the data will be stored in perpetuity without any additional upkeep or subscription fees paid by the user.

The Arweave protocol is designed to handle 1,000 base layer transactions per block with new blocks being mined roughly every two minutes. Each transaction may also store an unbounded number of signed, non-AR-transacting data items assembled into a bundle (i.e., a bundled data item). Since its launch in 2018, this scalable architecture has allowed the network's weave size (total data stored on the network) to grow to **151.0 TB** with approximately **2.6 billion** base layer transactions and bundled data items submitted from over **193k** unique wallets. The Arweave protocol endowment has received **68.9k AR** to cover the projected storage costs with a cost of storage **0.858 AR/GiB.** *

* data as of January 31, 2024

3.3 GATEWAYS

Gateways act as the front door to the permaweb. They are infrastructure utilities that sit above the base storage layer and allow users to access and query the information stored on Arweave. Gateways are specialized nodes responsible for data retrieval, caching, and serving as well as indexing transactions into a database that can be easily queried at scale. These functions are not performed by the Arweave mining nodes which are optimized for securing the blockweave and replicating information throughout the network through a mechanism known as Wildfire.

By taking on these responsibilities, gateways allow low cost and maintenance free hosting of static and dynamic content for users, creators, and developers. But there are costs associated with operating a gateway and Arweave does not offer any tokenomic incentives to offset these expenses. As the permaweb grows, these costs can become very significant.

Arweave.net, the primary community gateway, has scaled to meet the needs of the entire Arweave ecosystem and stored the entire weave. Over the last 6 months, this gateway indexed and cached approximately **9.4 million** base layer transactions and bundled data items per day, served **166 million** requests for data and node information per day, and responded to **5.3 million** GQL queries per day. *

Gateway use cases, and the types of administrators who operate them, can range from at-home projects hosted by hobbyists to larger decentralized platforms and dApps run by small teams, all the way up to scaled out environments capable of supporting enterprise offerings.

* data as of January 31, 2024

3.4 SMARTWEAVE

SmartWeave is the smart contract protocol built for Arweave. This standard utilizes 'lazy evaluation' which performs computation on the client side as opposed to other smart contract protocols that compute at the node level. This strategy eliminates associated computation fees as well as unburdens developers from file size limitations and programming language requirements.

With SmartWeave, developers can create SmartWeave Tokens (SWT) and incorporate incentive structures into their applications and infrastructure. The AR.IO Network leverages the **Warp SDK** implementation of this technology for development of its network protocol and IO token.

3.5 THE PERMAWEB

The permaweb is the third and final layer of the permanence pie. The permaweb stands for the permanent web, a collection of all the webpages, apps, and files stored on top of the Arweave network and enlivened with the functionality of the AR.IO Network. For users and builders, the permaweb offers low-cost, zero maintenance, permanent hosting of their web apps, files, and web pages.

AR.IO is a global network, protocol, and currency built on top of Arweave that enables the permaweb.

4 THE AR.IO SMARTWEAVE CONTRACT

4.1 OVERVIEW

The AR.IO SmartWeave contract contains all the functionality needed to support the network's currency, utilities, and management. Its codebase is written in Typescript, compiled to JavaScript. It is then immutably and publicly stored on the Arweave network and easily auditable by outsiders.

The contract architecture is grouped into several logical components:

1. Writing state changes through contract interactions.
2. Business logic to support state evaluation.
3. Contract state data model.

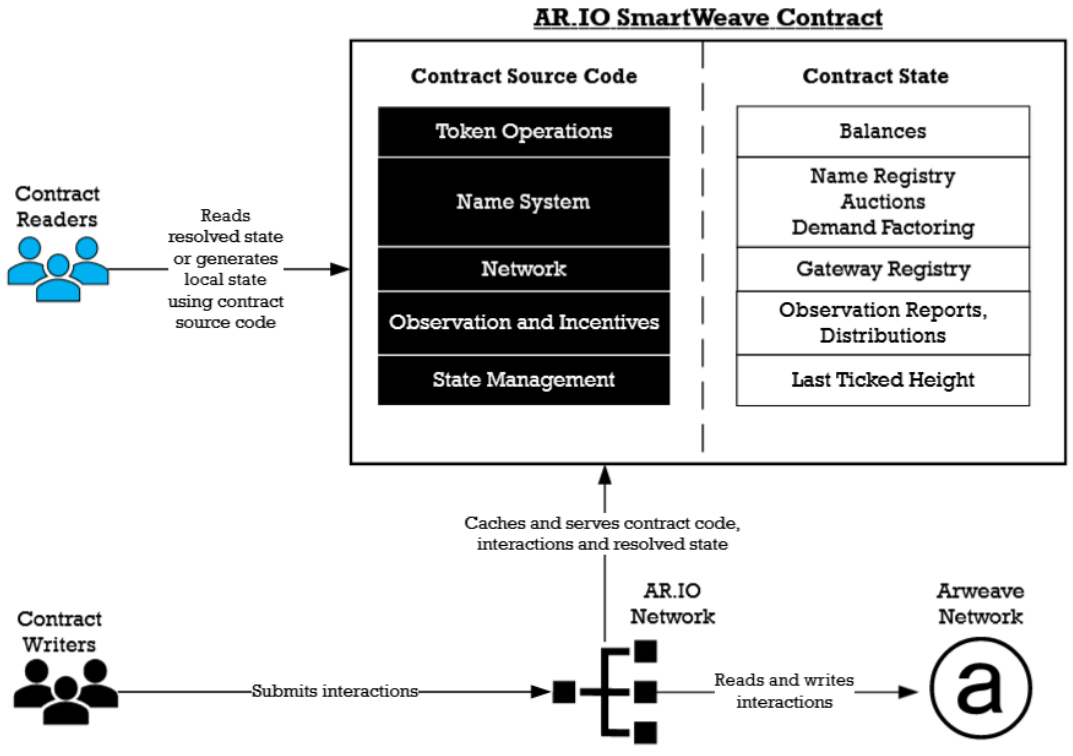
AR.IO SmartWeave Contract
Contract Source Code
Token Operations
Name System
Network
Observation and Incentives
State Management
Contract State
Balances
Name Registry Auctions Demand Factoring
Gateway Registry
Observation Reports, Distributions
Last Ticked Height
Contract Readers
Reads resolved state or generates local state using contract source code
Caches and serves contract code, interactions and resolved state
Contract Writers
Submits interactions
AR.IO Network
Reads and writes interactions
Arweave Network

Diagram 4.1: *SmartWeave Contract*

Writing to a contract's state, for example to purchase an ArNS Name, involves submitting a SmartWeave contract interaction. Each interaction is a base layer transaction on the Arweave Network, with an appropriate amount of gas fees paid in AR. All interactions must include the necessary information needed to pass the contract's business logic rules for state evaluation and generation.

Users, apps, and infrastructure reading from the contract can use any SmartWeave-aware client (like Warp Contracts) to download the contract's source code, evaluate every interaction made, and generate the latest state.

As the number of contract interactions increases, so does its state generation time and its overhead placed on downstream applications that are dependent on timely state updates. AR.IO Gateways, or other SmartWeave caches can continually evaluate, resolve, and publish the latest state of the AR.IO SmartWeave contract to provide apps and users with rapid state information in lieu of local state generation. It should be noted that state can always be independently cached and verified at anyone, at any time.

4.2 CONTRACT STATE

The AR.IO Contract state data model is a strict information architecture, defined within a JSON object, and used as the global, immutable database of record for the AR.IO Network.

It consists of, but is not limited to, the following data structures:

	Contract State Structure
Data Structure	**Definition**
Ticker	The short token symbol for the AR.IO Network.
Name	The friendly name of the token, shown in block explorers and marketplaces.
Version	The semantic version number of the contract.
Records	A list of all Arweave Name System records and their corresponding attributes.
Balances	A list of all unlocked token holder balances.
Vaults	A list of nested lists of token vaults, each with an end date at which the vaulted tokens are unlocked for transfer.
Reserved	A list of reserved ArNS names.
Auctions	A list of all active Arweave Name System name auctions.
Gateways	The Gateway Address Registry, which contains all active gateways, their stakes, and their delegates.
Observations	The health reports and failure summaries submitted by observers for an epoch.
Prescribed Observers	The observers selected / prescribed to perform observation duties for a given epoch.
Distributions	All gateways that have received token rewards for each epoch they participated in.
Base Registration Fees	The fundamental price for ArNS names, varying by character length, adjusted by step pricing. These are the Genesis Fees at network launch.
Demand Factoring	Used to track the demand for ArNS and dynamically update the prices of names.

Table 4.2: *Contract State Data Structures*

4.3 STATE TICKING

A few contract features rely on the passage of time, and thus chain height, to drive changes to the state of the contract. Examples include, but are not limited to:

- availability of reserved names
- purchase and auction prices of names
- lease expiration statuses
- protocol reward distributions
- vaulted token unlocks

Since SmartWeave contracts are evaluated lazily, it is necessary for most read and write interactions on the contract to advance the state of the system to the present block height before conducting the interaction. This process is referred to as "ticking".

During read interactions, the results of the tick are ephemeral and the computation is disposed of once the read interaction result is returned. In the case of write interactions, however, the result of the state tick is carried forward to all downstream computations of the state.

The contract can be ticked at any time via a deliberate "tickState" interaction, but in the absence of those, any other write interaction will advance the contract's state prior to evaluating the interaction.

4.4 CONTRACT READING AND WRITING

The AR.IO SmartWeave contract includes the following non-exhaustive list of read and write interactions that relate to the IO token, network, incentives, and name system:

Contract Read and Write Interactions	
Action	**Description**
balance	Gets the amount of IO tokens held by a specific wallet.
priceForInteraction	Gets the price of a given contract interaction, like purchasing an ArNS name.
record	Gets information about a registered ArNS name.
auction	Gets information about an active ArNS name auction.
gateway	Gets registered gateway information for a specific gateway wallet and returns its performance weight.
gateways	Returns all network registered gateways and their performance weights.
observer	Checks if a specific observer wallet is a prescribed gateway observer for a network epoch and returns its performance weights.
observers	Returns all network registered observers and their performance weights.
prescribedObservers	Returns all prescribed observers for a network epoch.
transfer	Transfers IO Tokens from one wallet to another, unlocked.
buyRecord	Registers a new ArNS name instantly or through an auction.
submitAuctionBid	Places a bid for a name under an auction.
extendRecord	Extends the lease time for an active ArNS name.
increaseUndernameCount	Increases the amount of undernames usable for an ArNS name.
joinNetwork	Joins a gateway into the AR.IO Network and adds the gateway operator's wallet and gateway settings into the Gateway Address Registry.
leaveNetwork	Withdraws a registered gateway from the AR.IO network and returns all stake back to the operator and any delegates.
increaseOperatorStake	Increase the IO token amount staked for an existing registered gateway.
decreaseOperatorStake	Decrease the IO token amount staked for an existing registered gateway.
updateGatewaySettings	Modifies an existing registered gateway settings in the Gateway Address Registry.
delegateStake	Delegates IO tokens to a registered gateway.
decreaseDelegateStake	Withdraws a delegated gateway stake in portion or in full.
saveObservations	Saves the observation report and gateway failure summary from a prescribed observer.
tick	Performs cleanup and other utility functions to manage the state.
createVault	Creates a vault of locked tokens with a defined end block height.
vaultedTransfer	Transfers tokens to a recipient directly to a locked vault with an end time (in block height).
extendVault	Adds additional lock time (in blocks) to an existing vault of locked tokens.
increaseVault	Adds additional IO tokens to an existing vault of locked tokens.
epoch	Returns the current epoch's start and end block heights.

Table 4.4: *Contract Read and Write Interactions*

4.5 CONSTANTS

The AR.IO SmartWeave contract has fixed constants within the source code. These are enshrined at the genesis of the network and can only be changed through a contract fork. This ensures that these important values and parameters cannot be tampered with or changed without serious consideration and intent from the community.

The table below shows representative examples of constants which would require a contract evolution to modify:

Example Protocol Constants		
Example	**Category**	**Description**
Number of Decimals	Token	The number of decimal places to translate token units to sub-units.
Max years	ArNS	The maximum amount of time that a name can be leased.
Max name length	ArNS	The maximum character length of a name.
Minimum Stake	Network	The minimum amount of IO staked to join a gateway to the network.
Max Observers per Epoch	Observation and Incentive	The maximum number of observers that are prescribed each epoch.
Epoch Block Length	Observation and Incentive	The number of blocks that have to elapse for an epoch to complete and incentives distributed.

Table 4.5: *Example Protocol Constants*

4.6 CONTRACT MANIFEST

SmartWeave write interactions to the AR.IO Network SmartWeave Contract must be unbundled, base layer Arweave transactions. To enforce this, the AR.IO Network SmartWeave Contract State must only be generated from valid base layer transactions, as bundled SmartWeave transactions will not initially be supported by the AR.IO Network.

To enforce this rule, a SmartWeave Contract manifest is specified during deployment. It contains evaluation options that a client uses to properly evaluate the contract’s state. These evaluation options also extend to foreign contract calls, to ensure the full state is not processed with less secure evaluation options than those set for the one being read by the user.

4.7 PROTOCOL BALANCE

The Protocol Balance is the primary sink and source of IO tokens circulating through the AR.IO Network. This balance is akin to a central vault or wallet programmatically encoded into the network’s smart contract from which ArNS revenue is accumulated and incentive rewards are distributed.

This balance is stored like any other token balance in the SmartWeave contract, using the contract’s ID as the balance owner. Should a user or organization desire, tokens can even be sent directly into this balance to support the reward protocol and ecosystem.

4.8 CONTRACT IMMUTABILITY AND UPGRADES

The AR.IO SmartWeave contract code and all interactions made against it are permanently stored on the Arweave network. This immutability ensures that contract does not allow any direct evolution or upgrades, prohibiting any single actor or entity from upgrading the code and forcing network participants to use the new version. While this makes for an unstoppable protocol with a truly censorship resistant, permanent namespace, it poses coordination challenges for beneficial contract upgrades or fixes.

For scenarios that require an upgrade to the AR.IO SmartWeave contract, a proposal for a protocol fork can be initiated by any interested parties to gauge community support and gain consensus. The key concerns are ensuring that:

- the code is upgraded.
- the community is aligned.
- the parent fork's contract state is respected and maintained in its entirety.

In light of these considerations, any proposal to fork the AR.IO SmartWeave contract must be approached with a collaborative mindset, prioritizing the collective interests of the network's community. The goal of such a fork should not only be to enhance the protocol's capabilities but also to ensure continuity and respect for the existing network state. This approach aligns with the ethos of community-driven consensus and development, as emphasized in Arweave's "Draft 17". It ensures that the evolution of the AR.IO Network remains in the hands of its users, maintaining the integrity and resilience that are foundational to its design. Ultimately, this method of "evolutionary forks" allows the AR.IO Network to adapt and improve over time, while upholding its commitment to permanence and censorship resistance. The diagram below was inspired by Draft 17 and is a visual representation of how a "pro-social" upgrade fork should be addressed:

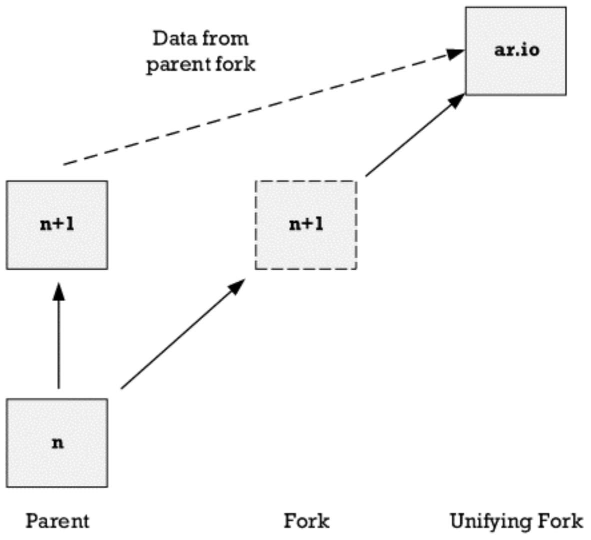

Diagram 4.8: *AR.IO Network Fork Process*

Arweave Network Fork
Should the Arweave base layer experience a fork, the AR.IO contract will simultaneously exist on both the original and forked networks. This situation presents a critical decision point for the AR.IO ecosystem. The community, including the Foundation, gateway operators, and end users, will need to assess the fork's nature and determine its alignment with the network's core principles and direction. The fork's perception, whether seen as collaborative or contentious, will significantly influence the community's response. The AR.IO community's decision on how to proceed must prioritize the network's long-term health and values. This scenario underscores the need for collaborative decision-making and community-driven consensus to maintain the network's integrity and sustainability.

5 THE IO TOKEN

5.1 OVERVIEW

IO is the multifunction SmartWeave Token (SWT) that powers The AR.IO Network and its suite of permaweb applications. The IO Token (ф) has many uses, including protocol incentives, staking by gateways, and payments for services like the Arweave Name System (ArNS). The token acts as a permissionless and censorship resistant medium of common value for the network.

Creating a new, dedicated token for the network was chosen over leveraging an existing token for the following reasons:

- Other digital currencies are not optimized to perform with the network's incentive structure.
- The network should be incentivized by a token whose value is not directly pegged to any outside factors.

The above factors are further elaborated on throughout this paper and are intended to drive an alignment of community incentives to build and participate in the AR.IO ecosystem.

5.2 SUPPLY AND ALLOCATION

At genesis, the full total fixed supply of 1,000,000,000 ф (one billion IO), with sub-unit micro IO (μIO) where one IO = 1,000,000 μIO), will be created and allocated to the following general categories:

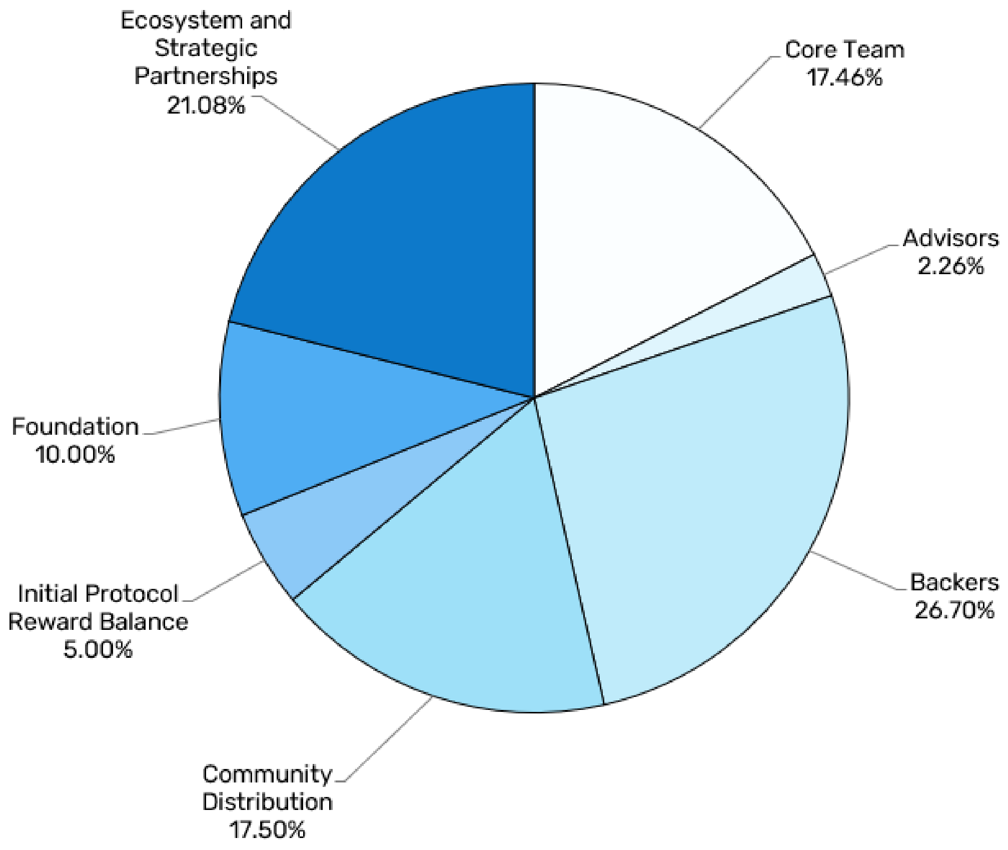

Chart 5.2: *IO Token Allocation*

IO Token Allocation		
Item	**Allocation (ф)**	**% of Total**
Core Team	174,600,000	17.46%
Advisors	22,600,000	2.26%
Backers	267,000,000	26.70%
Community Distribution	175,000,000	17.50%
Initial Protocol Reward Balance	50,000,000	5.00%
Foundation	100,000,000	10.00%
Ecosystem and Strategic Partnerships	210,800,000	21.08%
Total:	**1,000,000,000**	**100.00%**

Table 5.2: *IO Token Allocation*

The protocol shall have a fixed total token supply as such, the smart contract shall not have a "mint new tokens" function. A protocol fork will be required should the AR.IO community determine that an adjustment to the token supply is needed to support the ecosystem's growth and wellbeing.

5.3 TOKEN UNLOCK SCHEDULE

Token allocations for the Core Team, Advisors, Backers, Foundation, and Ecosystem are subject to unlocking schedules. These tokens can be in either of the following states:

1. **Locked:** cannot be transferred, used for ArNS purchases, gateway staking, or delegated staking.
2. **Unlocked:** full functionality available.

The charts below summarize the proposed vesting and unlock schedule of the full token supply:

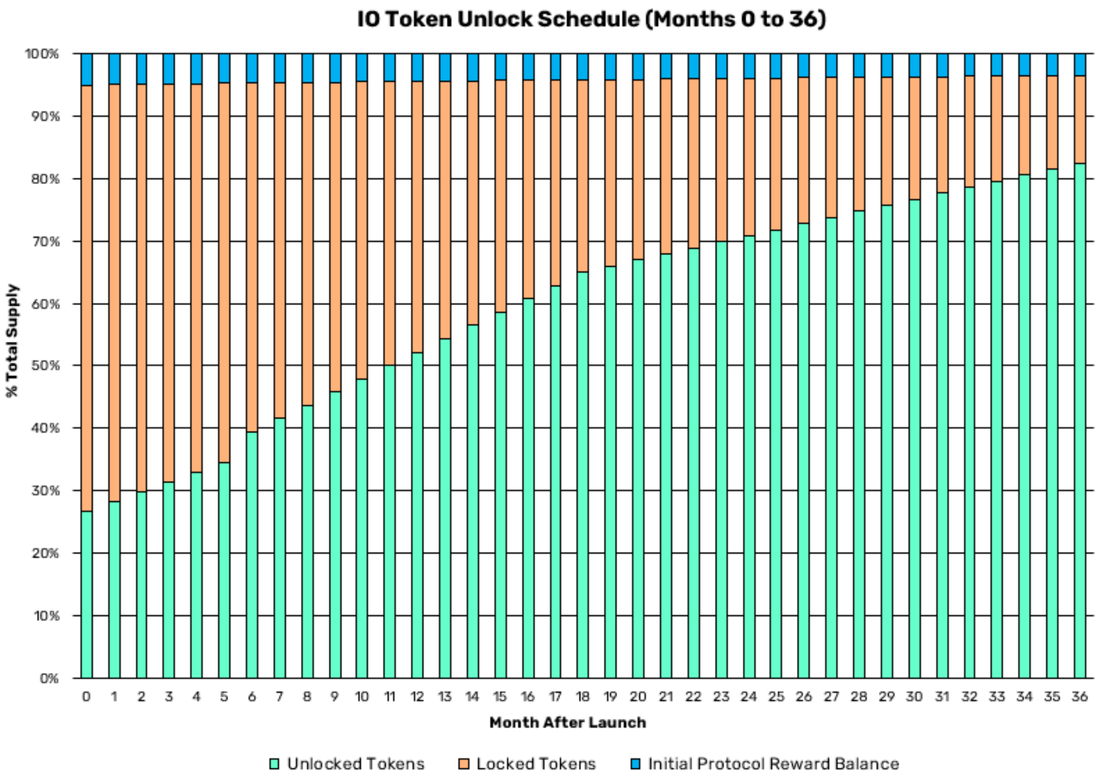

Chart 5.3-1: *IO Token Unlock Schedule (Months 0 to 36)*

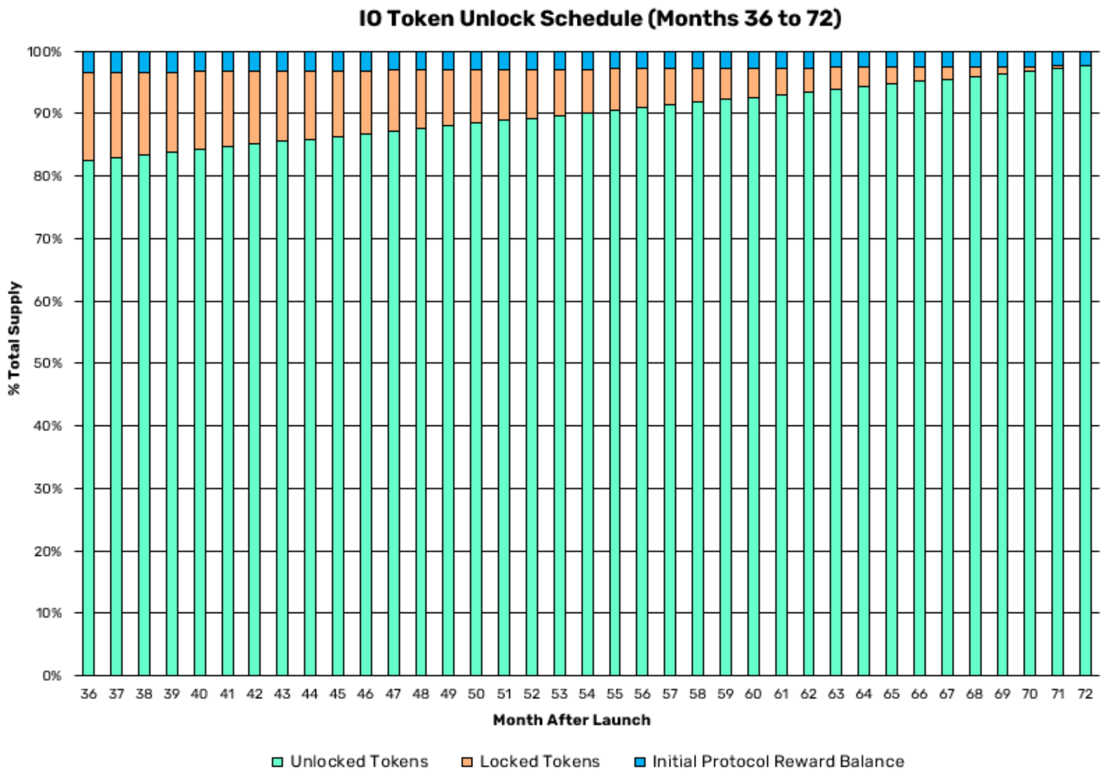

Chart 5.3-2: *IO Token Unlock Schedule (Months 36 to 72)*

5.4 COMMUNITY DISTRIBUTION

A portion of tokens have been set aside for "Community Distribution" for bootstrapping network adoption through various mechanisms such as airdrops. The rules, eligibility, and distribution mechanism shall be determined by the Foundation and disclosed accordingly.

5.5 INITIAL PROTOCOL REWARD BALANCE

To spur the network's protocol reward distributions and make it attractive for gateway operators to participate, especially during the network's early stages, an allocation of tokens will be "pre-loaded" into the protocol balance at launch. These tokens will act as a form of circulating supply inflation as the tokens will be locked in the protocol balance and subject to the rules of the reward distribution. Once distributed as rewards, the tokens will be unlocked and thereby increase the total circulating supply. Refer to Section 10 for additional details around the incentive and reward distribution protocol.

6 STAKING AND USER INTERACTIONS

6.1 OVERVIEW

This section outlines the key interactions among participants within the AR.IO Network, focusing on how the IO Token facilitates these activities. The range of interactions covers various roles and actions, including:

- **Gateway Participation:** Gateway operators must stake IO tokens to join and actively participate in the network.
- **Eligibility for Protocol Rewards:** Both individuals who stake tokens as gateway operators and those who delegate tokens to a gateway are positioned to receive protocol rewards.
- **ArNS Name Purchases:** Acquiring friendly names through the Arweave Name System (ArNS) requires IO tokens. These transactions directly contribute to the protocol, with the proceeds being redistributed through the Observation and Incentive Protocol.
- **Universal Currency:** Within the AR.IO ecosystem, IO tokens serve as a versatile currency, enabling network participants to make purchases and exchange value.

Moreover, IO tokens play a crucial role in driving ecosystem growth, fueling incentive programs, investments, bounties, and grants designed for active participants. Participation in the network is open and permissionless, allowing users to engage in various activities and qualify for rewards. Notably, individuals and teams can simultaneously fulfill multiple roles within the network, contributing to its diversity and vitality.

6.2 STAKING

Staking tokens within the AR.IO Network serves a dual primary purpose: it signifies a public commitment by gateway operators and qualifies them for reward distribution.

In the AR.IO ecosystem, "staking" refers to the process of locking a specified amount of IO tokens into a protocol-controlled vault. This act signifies an opportunity cost for the gateway operator, acting both as a motivator and a public pledge to uphold the network's collective interests. Once staked, tokens remain locked until the staker initiates an 'unstake' action or reaches the end of the vault’s preset lock period.

It is important to note that the IO Token is non-inflationary, distinguishing the AR.IO Network's staking mechanism from yield-generation tools found in other protocols. Staking in this context is about eligibility for potential rewards rather than direct token yield. By staking tokens, gateway operators (and their delegates) demonstrate their commitment to the network, thereby gaining eligibility for protocol-driven rewards and access to the network’s shared resources.

Details on the network benefits and reward distributions associated with staking are further elaborated throughout this paper.

6.2.1 GATEWAY STAKING

A gateway operator must stake tokens to join their gateway to the network, which not only makes them eligible for protocol rewards but also promotes network reliability. This staking requirement reassures users and developers of the gateway's commitment to the network’s objectives, and gateways that adhere to or surpass network performance standards become eligible for these rewards.

Notably, the AR.IO Network does not employ a stake slashing mechanism as a penalty for gateways that underperform or engage in harmful activities. Instead, the focus is on incentivizing positive behavior and high performance through rewards.

The protocol imposes specific values for gateway staking, including:

Function	Gateway Staking Values Description	Value
Network Join Stake	Minimum number of tokens required for a gateway to join the network (i.e., minimum gateway token stake).	50,000 ϕ
Reduce Stake Duration	Minimum withdrawal time before a gateway operator can remove tokens from their gateway stake (while still remaining above the minimum staking requirement). A gateway's total stake is immediately impacted once stake is removed or reduced.	30 block-days
Network Leave Duration	Minimum duration for a gateway to completely leave the network and have the network join stake returned to the operator.	90 block-days

Table 6.2.1: *Gateway Staking Values*

6.2.2 DELEGATED STAKING

To promote participation from a wider audience, the network shall allow anyone with available IO tokens to partake in delegated staking. In this, users can choose to take part in the risk and rewards of gateway operations by staking their tokens with an active gateway (or multiple gateways) through an act known as delegating. By delegating tokens to a gateway, a user increases the overall stake of that gateway. A delegated staker proxies their stake to gateways and therefore entrusts gateway operators to utilize that stake in maintaining a quality of service befitting the permaweb.

As protocol rewards are accumulated by a gateway, they can be distributed proportionately among its delegates. This ensures that the rewards are shared fairly, reflecting each delegate's contribution to the gateway's stake.

Additionally, gateway operators may choose to provide their delegates with non-protocolized incentives. These can take the form of faster data access, privileged data access, and other "off-chain perks".

Delegates have the flexibility to adjust their stakes, either by increasing them, or partially, or fully withdrawing them, subject to a minimum stake withdrawal period. Delegated stakers can use this withdraw stake action as a form of recourse should a gateway operator act maliciously. In the event a gateway exits the network, all staked tokens delegated to it are automatically returned to the respective delegates.

Gateway operators retain the discretion to enable or disable delegated staking. They can also set specific criteria for accepting delegated stakes, including restrictions on certain wallet addresses (allow list) and limits on the amount of stake accepted. Note that a gateway must already been joined to the network in order to accept delegated stakes.

The protocol imposes specific values for delegated staking, including:

Delegated Staking Values		
Constant	Description	Value
Global Minimum Stake	Minimum number of tokens that can be delegated to a gateway. Gateway operators shall have the option to set a higher value.	500 ϕ
Maximum Delegated Stakers	Maximum amount of delegated stakers that can delegate to a single gateway.	10,000
Delegate Reward Share Ratio	Gateway-specific selected percentage of protocol rewards to be shared proportionally amongst its delegates.	0 to 100%
Reduce or Remove Stake Duration	Minimum duration it takes for a delegated staker to unstake delegated tokens. A gateway's total stake is immediately impacted once delegated stake is removed or reduced.	30 block-days

Table 6.2.2: *Delegated Staking Values*

Note that any changes made by a gateway to its delegated stake settings take effect immediately. For example, should a gateway previously accepting delegates decide to no longer have that setting enabled then any existing delegates will have their stake returned after the minimum stake withdrawal duration.

6.3 USER TYPES

The table below represents the general user types that interact with the AR.IO Network:

General Network User Types				
User Type	Bootstrap a Gateway	ArNS Purchasing	Protocol Rewards	External Incentive Programs
Builders, Creators, and App End Users				✓
Token Holders	✓	✓		
Delegated Stakers			✓	
Gateway Operators			✓	✓

Table 6.3: *General Network User Types*

Note that these are broad categories. Individuals and teams can and will serve multiple roles within the network at the same time. For instance, a user can simultaneously hold tokens, operate a gateway, as well as delegate to another.

6.3.1 BUILDERS, CREATORS, AND APP END USERS

Builders (software developers), content creators, and app end users directly interact with the permaweb – they are the reason it was created.

- Builders develop applications and infrastructure that leverage the AR.IO Network.
- Builders may charge users certain fees whether in AR, IO, or other currencies, depending on their business or community goals.

- Builders may receive investments, bounties, or grants from the Foundation for creating applications, infrastructure, and services within the AR.IO Network.
- Creators leverage network tooling to create digital assets and communities.
- Users upload and interact with permaweb files, applications, and infrastructure.
- Users may receive incentives from the Foundation or Builders to encourage adoption.

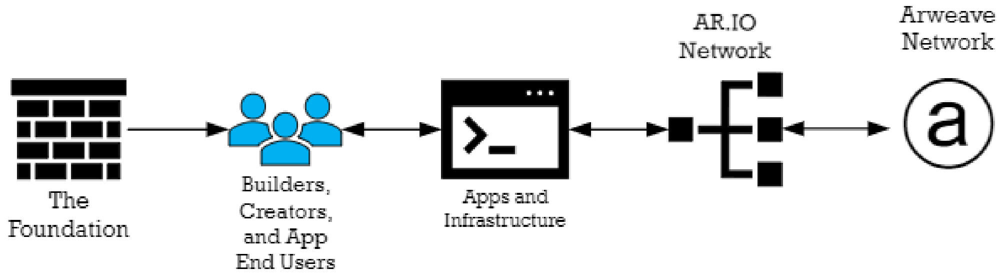

Diagram 6.3.1: *Builders, Creators, and App End User Interactions*

6.3.2 TOKEN HOLDERS

Token Holders are users that simply hold unlocked tokens at a given moment, whether long term or transiently. By having tokens available to transfer or spend, a user can perform the following general interactions with the network:

- Buy and sell tokens through exchanges.
- Buy and manage ArNS names.
- Bootstrap a gateway (i.e., join a gateway to the network)
- Delegate stake to an existing gateway.
- Lock tokens into a vault and withdraw eligible tokens from a timebound locked vault.
- Transfer tokens to peers (including directly to a peer's locked vault).

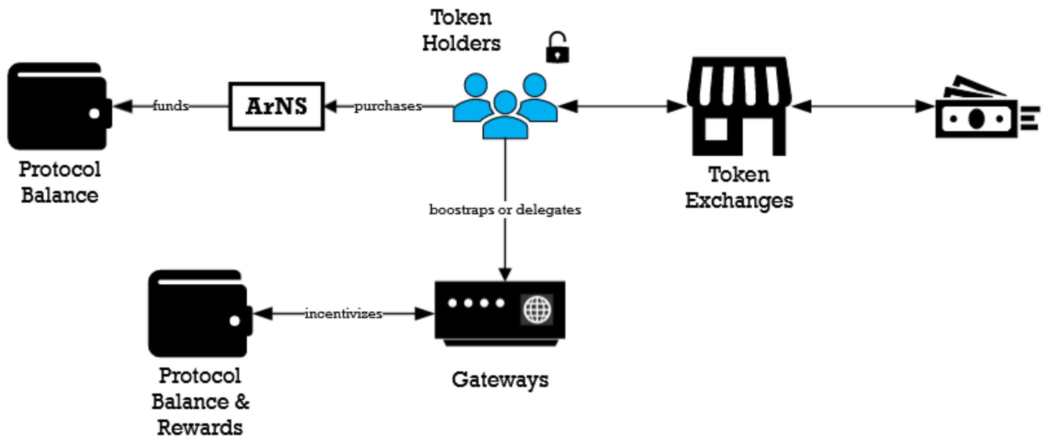

Diagram 6.3.2: *Token Holder Interactions*

6.3.3 GATEWAY OPERATORS

Gateway Operators are responsible for making the permaweb real by keeping their services online, healthy and updated to provide a great experience for users and apps. They are periodically prescribed as observers, tasked to keep the network healthy by identifying and reporting on poorly performing gateway endpoints. They provide a means for the protocol to award the gateways for performing optimally.

The following general interactions are performed by gateway operators with the network:

- Joining, withdrawing, or managing their gateway settings.
- Increasing or decreasing stake.
- Managing delegated staking configurations.
- Participate in the observation and incentive protocol.
- Save observation reports on-chain for community and protocol verification.
- Receive incentives for performing gateway and observation duties when prescribed.
- Receive potential support / grants from the Foundation for operating a gateway.

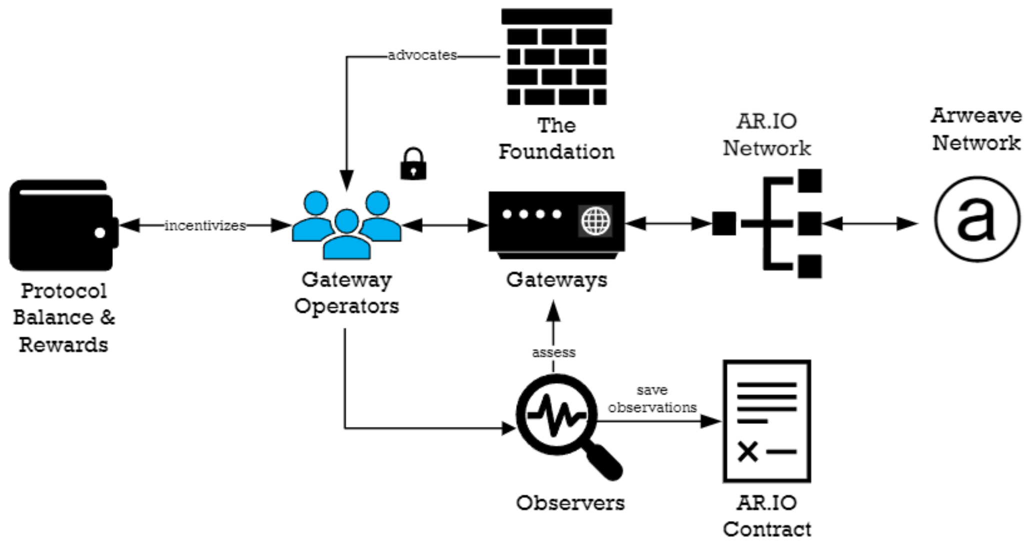

***Diagram 6.3.3:** Gateway Operator Interactions*

6.3.4 DELEGATED STAKERS

Delegated Stakers allocate their stakes to gateway operators, with the goal of gaining exposure to potential protocol rewards and/or to strengthen a gateway's total token stake. Their general interactions with the network include:

- Adjusting the amount of their delegated stake, either by increasing, decreasing, or withdrawing completely.
- Sharing in the protocol rewards distributed to the gateway to which they have delegated their stakes.

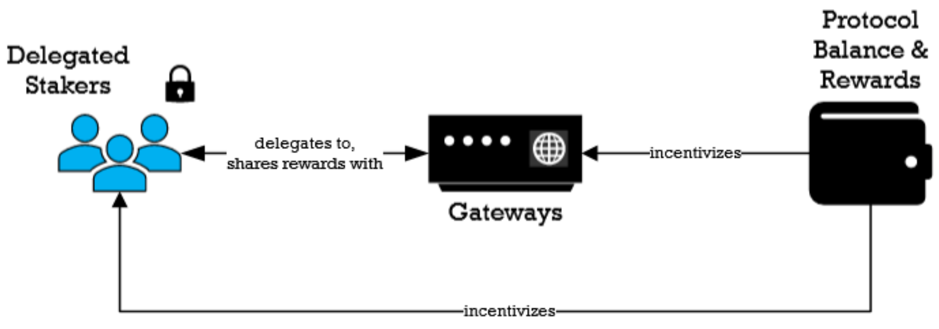

***Diagram 6.3.4:** Delegated Staker Interactions*

7 GATEWAY ARCHITECTURE

7.1 OVERVIEW

Gateways are the workhorses of the AR.IO Network. Their primary role is to act as a bridge between the Arweave network and the outside world. This means that a gateway's main task is to make it easier for users to interact with the Arweave network by simplifying the technical processes of writing, reading, and discovering data on the blockweave in a trust-minimized fashion.

7.2 AR.IO GATEWAY FUNCTIONS

The functions of an AR.IO gateway are broken down into the following categories:

Writing data involves:

- Proxying base layer transaction headers to one or more healthy and active Arweave nodes (miners) to facilitate inclusion in the mempools of as many nodes as possible.
- Proxying chunks for base layer Arweave transactions to Arweave nodes to help facilitate storage and replication of the chunks on the blockweave.
- Receiving and bundling so-called bundled data items (e.g., ANS-104 spec) as base layer transactions.

Reading involves retrieving:

- Transaction headers for a base layer Arweave transaction.
- Individual data chunks for a base layer Arweave transaction.
- Blocks from the blockweave.
- Storage pricing rates for data from the Arweave node network.
- Contiguous streams of chunks representing an entire base layer transaction.
- Bundled data items (e.g., ANS-104).
- Wallet information (e.g., token balance).

Discovering data involves:

- Facilitating efficient, structured queries for base layer transactions, bundled data items, and wallet data by:
    - examining incoming streams of data (i.e., directly ingested transactions and data items, blocks emitted by the chain, etc.).
    - managing index data in a database or analogous data store.
- Parsing and executing user queries.
- Facilitating friendly-path routing via Arweave manifest indexing.

Including other benefits and capabilities such as:

- Facilitating friendly-subdomain-name routing to Arweave transactions via a direct integration with the Arweave Name System (ArNS).
- Providing the modularity and configurability necessary for operating extensible gateways that can be deployed at small or large scales to meet the needs of specific applications, use cases, communities, or business models.
- Providing pluggable means for consuming telemetry data for internal and external monitoring and alerting.
- Facilitating configurable content moderation policies.
- Providing connectivity to a decentralized network of other AR.IO gateways, enabling data sharing and other shared workloads.

7.3 GATEWAY MODULARITY

A design principle of AR.IO gateways is that their core components should be interchangeable with compatible implementations.

The core services in the gateway are written in Typescript, with flexible interfaces to the various subsystems and databases. This allows operators to customize their gateway to meet their specific requirements. Gateway services can be turned on or off depending on the operator's needs. For example, an operator might choose to have their gateway serve data, but not actively index bundled data.

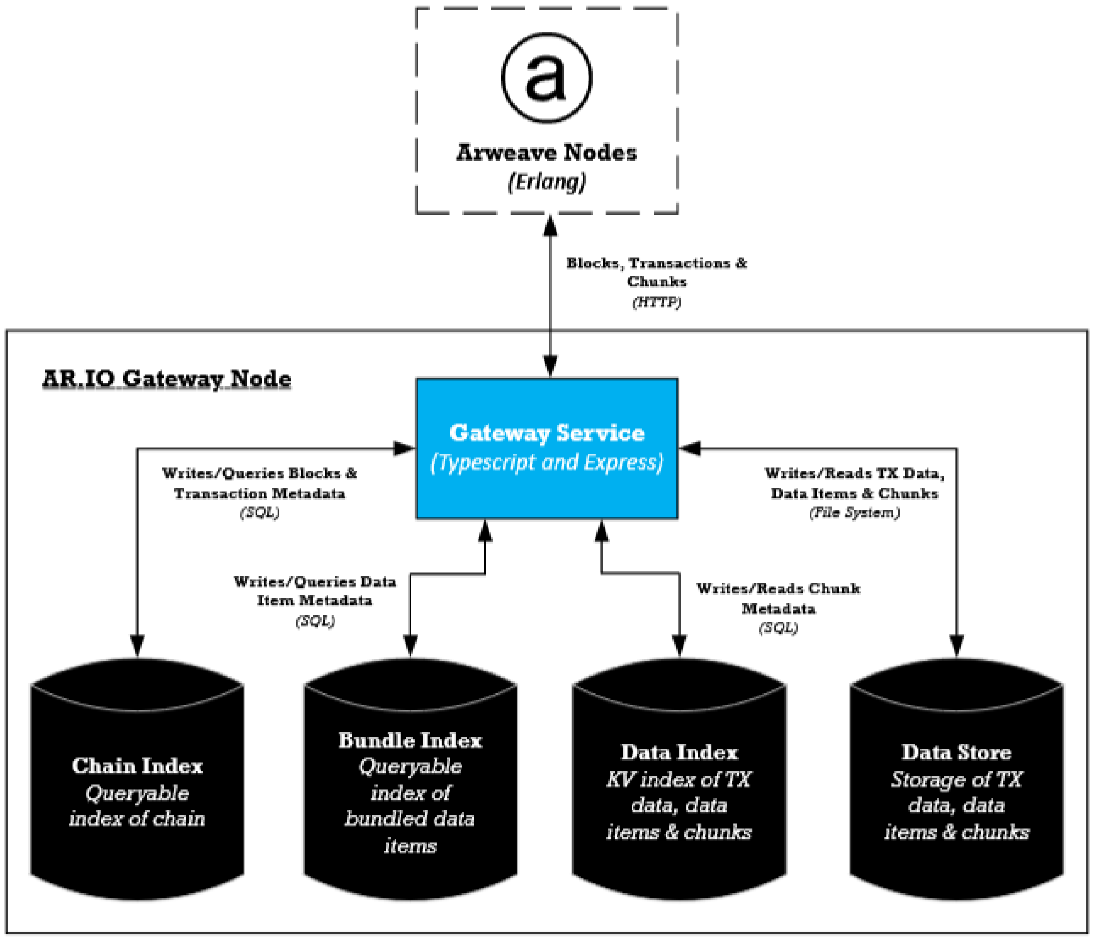
Arweave Nodes
(Erlang)
Blocks, Transactions & Chunks
(HTTP)
AR.IO Gateway Node
Gateway Service
(Typescript and Express)
Writes/Queries Blocks & Transaction Metadata
(SQL)
Writes/Reads TX Data, Data Items & Chunks
(File System)
Writes/Queries Data Item Metadata
(SQL)
Writes/Reads Chunk Metadata
(SQL)
Chain Index
Queryable index of chain
Bundle Index
Queryable index of bundled data items
Data Index
KV index of TX data, data items & chunks
Data Store
Storage of TX data, data items & chunks

Diagram 7.3: *Gateway System Diagram*

This flexibility also allows operators to utilize the technologies that are appropriate for the scale and environments in which they operate.

For example, small scale operators might want to use low-overhead relational databases to power their indexing while larger scale operators might opt to use cloud-native, horizontally scalable databases. Analogous examples for storage and caching exist as well.

Gateway Tech Stack Options				
Topology	**Chain Index**	**Bundle Index**	**Data Index**	**Data Store**
Small	SQLite	SQLite	SQLite	Local File System
Large	PostgreSQL	OpenSource	Cassandra	S3 Compatible

Table 7.3: *Example Gateway Tech Stacks*

7.4 ARNS INDEXING AND ROUTING

The Arweave Name System's (ArNS) state is managed by the IO token's SmartWeave smart contract. AR.IO gateways shall perform the following minimum functions relevant to ArNS:

- Actively track state changes in the contract.
- Maintain up-to-date indexes for routing configurations based on the state of the IO contract as well as the states of the Arweave Name Token (ANT) contracts to which each name is affiliated.
- Manage the expiration of stale records.
- Facilitate ArNS routing based on the subdomains specified on incoming requests where appropriate.
- Provide a custom HTTP response header for ArNS requests indicating the corresponding Arweave transaction ID.

7.5 CONTENT MODERATION

The AR.IO Network will adopt Arweave's voluntary content moderation model whereby every participant of the network has the autonomy to decide which content they want to (or can legally) store, serve, and see. Each gateway operating on the network has the right and ability to blocklist any content (or address) that is deemed in violation of its content policies or non-compliant with local regulations.

8 GATEWAY NETWORK

8.1 OVERVIEW

The AR.IO Network consists of AR.IO gateway nodes, which are identified by their registered Arweave wallet addresses and either their IP addresses or hostnames, as stored in the network's smart contract Gateway Address Registry (GAR).

These nodes adhere to the AR.IO Network's protocols, creating a collaborative environment of gateway nodes that vary in scale and specialization. The network promotes a fundamental level of service quality and trust minimization among its participants.

Being part of the network grants AR.IO gateways an array of advantages, such as:

- Simplified advertising of services and discovery by end users via the Gateway Address Registry.
- More rapid bootstrapping of key gateway operational data due to prioritized data request fulfillment among gateways joined to the network.
- Sharing of data processing results.
- Auditability and transparency through the use of AGPL-3 licenses, which mandate public disclosure of any software changes, thereby reinforcing the network's integrity and reliability.
- Improved network reliability and performance through an incentive protocol, which uses a system of evaluations and rewards to encourage high-quality service from gateways.
- Eligibility to accept delegated staking improving a gateway's discoverability and reward opportunities.

8.2 GATEWAY ADDRESS REGISTRY (GAR)

Any gateway operator that wishes to join the AR.IO Network must register their node in the AR.IO SmartWeave Contract's "Gateway Address Registry", known as the GAR. Registration involves staking a minimum amount of IO tokens and providing additional metadata describing the gateway service offered.

This metadata includes details such as:

	Gateway Registry Metadata
Item	**Description**
Gateway Wallet	The operator's Arweave wallet address, the primary identifier of the gateway. A single wallet can only be registered to one gateway at a time.
Observer Wallet	The wallet used to submit observation reports and receive observation rewards. (Can be different from the Gateway Wallet.)
IO Token Stake	The amount of IO tokens staked which must be above the network minimum.
Network Information	Fully Qualified Domain Name (FQDN) / port / protocol used to access this gateway through.
Delegated Staking Status	Elective to open delegation to the public with optional allow list for delegated staking support.
Delegate Reward Share Ratio	If enabled, indicates how this gateway's protocol rewards are distributed to delegates vs kept by the operator.
Settings	Other settings like a friendly label, a note, or gateway properties referenced by transaction ID.

Table 8.2: *Gateway Registry Metadata*

After joining the network, the operator's gateway can be easily discovered by permaweb apps, its health can be observed, and it can participate in data sharing protocols. A gateway becomes eligible to participate in the network's incentive protocol in the epoch following the one they joined in.

The gateway operator can modify their gateway's GAR configuration as needed, which includes increasing or decreasing token stake. Operators can completely remove their stake and leave the AR.IO Network following a minimum network exit wait time. This exit time ensures that gateways and their delegates cannot quickly escape from the impacts of malicious activity. Note that gateways that have signaled to leave the network (withdraw their minimum stake) remain eligible for gateway and observation incentive rewards while within the withdrawal period (see Section 10).

The GAR advertises the specific attributes of each gateway including its stake, delegates, and settings. This enables permaweb apps and users to discover which gateways are currently available and meet their needs. Apps that read the GAR can sort and filter it using the gateway metadata, for example, ranking gateways with the highest stake or reward performance at the top of the list. This would allow users to prefer the higher staked, more rewarded gateways over lower staked, less rewarded gateways.

In addition to operator set variables, the GAR also makes visible certain criteria associated with each gateway's performance in the network's Observation and Incentive protocol – the details of which can be found in the respective section of this paper.

8.3 DATA SHARING

A key advantage and incentive for networked AR.IO gateways over standalone gateways will be their ability to preferentially share various kinds of Arweave data among one another. Each gateway will advertise its registered Arweave wallet address, so other network participants know who they are.

Gateways will be able to identify AR.IO Network peers by evaluating the Gateway Address Registry (GAR) within the AR.IO SmartWeave contract. They will utilize that peer list to request as-yet-uncached data on behalf of their requesting clients or in service of their internal workflows. This can include requests for transaction data, data items, and chunks. The Arweave Network shall act as the backstop for all block data, transaction data, and chunk data.

Additionally, gateways that receive requests for cache-missed data from other gateways can provide a higher quality of service to other AR.IO gateways than that which is provided to general users, apps, and infrastructure. However, gateways are not forced to share data with one another, and can choose to not share their data if the intended recipient is acting maliciously. Such behaviors might include failure to reciprocate in data sharing, engaging in dishonest activities / observation, or distributing invalid data.

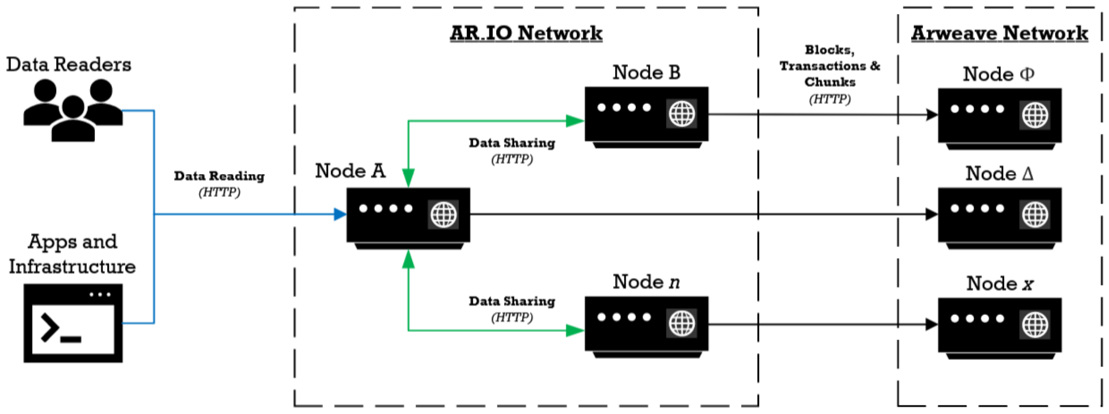

Diagram 8.3: *Data Access and Sharing Diagram*

9 ARWEAVE NAME SYSTEM (ARNS)

9.1 OVERVIEW

Arweave URLs and Transaction IDs are long, difficult to remember, and occasionally miscategorized as spam. The Arweave Name System (ArNS) aims to resolve these problems in a decentralized manner. ArNS is a censorship-resistant naming system stored on Arweave, powered by IO tokens, enabled through AR.IO gateway domains, and used to connect friendly domain names to permaweb dApps, web pages, data, and identities.

It's an open, permissionless, domain name registrar for anything on the permaweb.

This system works similarly to traditional DNS services, where users can purchase a name in a registry and DNS Name servers resolve these names to IP addresses. The system shall be flexible and allow users to purchase names permanently or lease them for a defined period based on their use case. With ArNS, the registry is stored permanently on Arweave (with SmartWeave), making it immutable and globally resilient. This also means that apps and infrastructure cannot just read the latest state of the registry but can also check any point in time in the past, creating a "Wayback Machine" of permanent data.

Users can register a name, like `ardrive`, within the ArNS Registry. Before owning a name, they must create an Arweave Name Token (ANT), a SmartWeave Token and open-source protocol used by ArNS to track the ownership and control over the name. ANTs allow the owner to set a mutable pointer to any type of permaweb data, like a page, dApp or file, via its Arweave Transaction ID.

Each AR.IO gateway acts as both a SmartWeave cache and an ArNS Name resolver. They will generate the latest state of both the ArNS Registry and its associated ANTs and serve this information rapidly for apps and users. AR.IO gateways will also resolve that name as one of their own subdomains, e.g., `https://ardrive.arweave.net` and proxy all requests to the associated Arweave Transaction ID. This means that ANTs work across all AR.IO gateways that support them: `https://ardrive.ar-io.dev`, `https://ardrive.g8way.io/`, etc.

Users can easily reference these friendly names in their browsers, and other applications and infrastructure can build rich solutions on top of these ArNS primitives.

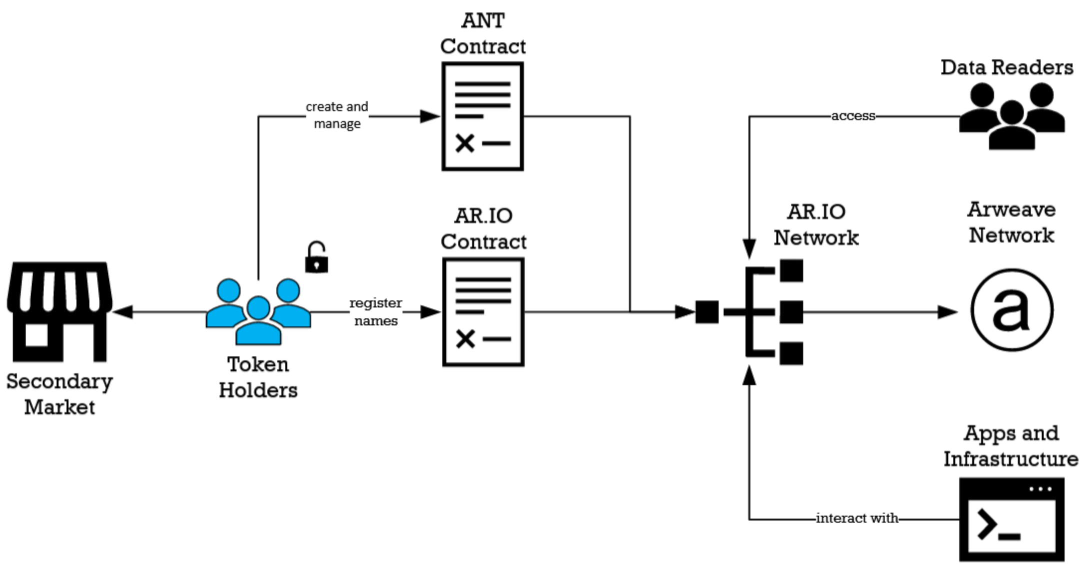

Diagram 9.1: *Arweave Name System Interactions*

9.2 NAME REGISTRY

The ArNS Registry is a list of all the registered names and their associated ANT smart contract addresses. Registering a name requires spending IO tokens based upon the name length and purchase type. The system shall allow users to either lease a name on a yearly basis (maximum up to 5 years) or purchase that name permanently.

The registry uses the following key rules, embedded within the SmartWeave Contract:

- The genesis prices of names are set within the contract itself; these are the starting conditions.
- Name prices vary based on name length, purchase type (lease vs buy), lease duration, and the current Demand Factor. See the Dynamic Pricing section for more details.
- Name records in the registry each include a pointer to its Arweave Name Token Smart Contract address, its lease end time, and undername allocation.
- Anyone with available IO Tokens can extend any name's active lease.
- Anyone with available IO Tokens can purchase additional undername capacity for any actively registered name.
- When a lease expires, there is a grace period where it can still be extended before anyone else can repurchase the name with a new ANT.

Once added, name records cannot be removed from the registry. A leased name's associated ANT smart contract address cannot be changed until the lease has expired and a new one is purchased. Care must be taken by the owners of permanent name purchases to ensure that their ANT supports an `evolve` ability should it be desired to add or modify functionality in the future as these name purchases are permanently tied to the associated ANT. Owners of permanently purchased names must understand the consequences of private key loss, which results in not being able to update any data pointers for this name.

9.2.1 NAME VALIDATION RULES

All names registered shall meet the following criteria:

1. Valid names include only numbers 0-9, characters a-z and dashes.
2. Dashes cannot be leading or trailing characters.
3. Dashes cannot be used in single character domains.
4. 1 character minimum, 51 characters maximum.
5. Shall not be an invalid name predesignated to prevent unintentional use/abuse such as `www`.

9.3 ARWEAVE NAME TOKEN (ANT)

To establish ownership of a record in the ArNS Registry, each record contains both a friendly name and a reference to an Arweave Name Token, ANT. Name Tokens are unique SmartWeave tokens that give their owners the ability to update the Arweave Transaction IDs that their associated friendly names point to.

The ANT SmartWeave Contract is a standardized contract that implements the specific ArNS Record specification required by AR.IO gateways who resolve ArNS names and their Arweave Transaction IDs. It also contains other basic functionality to establish ownership and the ability to transfer ownership and update the Arweave Transaction ID.

Name Tokens have an owner, who can transfer the token and control all of its modifiable settings. These settings include modifying the address resolution time to live (`ttl`) for each name contained in the ANT, and other settings like the ANT Name, Ticker, and an ANT Controller. The controller can only manage the ANT and set and update records, name, and the ticker, but cannot transfer the ANT. Note that ANTs are initially created in accordance with network stands by an end user who then has to ability to transfer its ownership or assign a controller as they see fit.

Secondary markets could be created by ecosystem partners that facilitate the trading of Name Tokens. Additionally, tertiary markets could be created that support the leasing of these friendly names to other users. Such markets, if any, would be created by third parties unrelated to and outside of the scope of this paper or control of the Foundation.

The table below indicates some of the possible interactions with an ANT and who can perform them:

ANT Interactions			
Type	**ANT Owner**	**ANT Controller**	**Any IO Token Holder**
Transfer ANT	✓		
Add / remove controllers	✓		
Set records (pointers)	✓	✓	
Update records, name, ticker	✓	✓	
Extend / renew lease	✓	✓	✓
Increase undernames	✓	✓	✓

Table 9.3: *ANT Interactions*

9.3.1 UNDER_NAMES

ANT owners and controllers can configure multiple subdomains for their registered ArNS name known as "under_names" or more easily written "undernames". These undernames are assigned individually at the time of registration or can be added on to any registered name at any time.

Under_names use an underscore "_" in place of a more typically used dot "." to separate the subdomain from the main ArNS domain.

This means users can trust `dapp_ardrive` just like you would trust `ardrive` since the owner of `ardrive` is the only one who can configure `dapp_ardrive`.

Some other features that undernames allow include:

- Undernames are configured in the ANT that is referenced for a given name. ANT owners can add more undernames as `subDomains` in the ANT's `records` object, each of which can point to a different Arweave Transaction ID.
- Each registered name is provided with an allocation of 10 undernames by default. Additional undername capacity can be purchased individually and as needed.
- Other users could never register a name that resembles an undername on `ardrive` since "_" is not allowed to be registered in the ArNS registry.
- Another user can register `dapp-ardrive` but this is a separate ArNS domain altogether. In traditional DNS, it's like the difference in trusting `dapp-ardrive.io` (suspicious!) over the legitimate `dapp.ardrive.io`
- Unlike primary names, undernames can include an underscore "_" character within them, e.g. `version_dapp_ardrive` but must not be longer than the total `MAX_NAME_LENGTH` of an ArNS name. The total amount of characters for a name string consisting of undernames and underscore separators is 63 characters.

Undernames give more versatility and utility to owning an ArNS name.

9.4 WHY ALLOW LEASES

The Arweave protocol assigns a Transaction ID to each piece of content uploaded to the network. These identifiers are unique and never change. Contrasting this, ArNS is intentionally fluid and flexible – it allows friendly-name-identifiers to either be permanently assigned to a single transaction or updated to another ID as needed. There can be any number of names pointing to a single transaction.

To further the system's value while respecting that it should remain useful for 100s of years, ArNS allows for names to either be purchased for an indefinite time or leased on a yearly basis. The benefits of allowing leases (over indefinite ownership only) are as follows:

1. **Limited Name Availability:** Unlike the practically infinite available character space of Transaction IDs, there are only so many appealing short and memorable ArNS names available. By enabling leases, the system can prune inactive / unmaintained leased names to make them available to others.
2. **Squatter Prevention:** Names are priced according to their ownership length with permanent names having a premium. By offering a "cheaper" lease model, the system hopes to deter "land-rush squatting" of valuable names.
3. **Affordable:** Likewise, some use cases may have budget constraints and not be able to afford high-value indefinite purchases. Leases lower the barrier to entry for those projects.
4. **Temporary Needs:** Some projects may not even need a name indefinitely – reasons for that could be a short duration promotion or simply a test case.
5. **Ongoing Revenue Stream:** Leases enable a continuous stream of revenue into the protocol, thereby supporting ongoing operation of gateways and network development.
6. **Risk Mitigation:** Data stored on the permaweb is… permanent. Meaning that the ANT contracts themselves are immutable. Should a user accidentally "brick" an ANT contract holding a permanent name then that name is forever locked in that state. While this could be a practical use case for some, it would likely lead to user frustration for many. Contrarily, if a leased name is "bricked" then the ecosystem can simply wait for the lease to expire before that name can be reutilized.

It is important to reiterate that the system is flexible – just because a name is leased does not mean its registration cannot be continuously funded. If one would like to purchase a name indefinitely and have it linked to a single transaction indefinitely then the choice is entirely up to the user! In addition, regardless of purchase type, the history of each name's registration and use will be preserved on the permaweb and available for all to view or utilize.

9.5 ADDRESSING VARIABLE MARKET CONDITIONS

The future market landscape is unpredictable, and the AR.IO Network smart contract is designed to be immutable, operating without governance or mechanisms for manual intervention. In addition, the traditional method of employing a pricing oracle to fix prices relative to a stable currency is not viable due to the infancy of the available solutions as well as the inherent reliance on outside dependencies that would bring. Considering this, ArNS is designed to be self-contained and adaptive, with name prices set to reflect network activity and market conditions in aggregate, evolving steadily over time.

To achieve this, ArNS incorporates:

1. A dynamic pricing model that utilizes a "Demand Factor" to adjust fees in line with ArNS purchase activity.
2. An auction system intended to seek fair market value for premium names, allowing initial prices to adapt based on actual user interest and bids. In this context, "premium names" are defined by their character count and purchase type (lease or permanent).

ArNS is designed to align name valuations with their actual market value, effectively adapting within the constraints of its immutable smart contract framework.

9.6 DYNAMIC PRICING MODEL

The Arweave Name System (ArNS) introduces an adaptive pricing model for registering names within the AR.IO Network. The core objective is to strike a balance between market demand and pricing fairness, leveraging both static and dynamic pricing elements. The system differentiates prices based on character lengths of names and offers varied purchasing options such as leasing, permanent acquisition, and undernames.

A unique feature of the ArNS pricing mechanism is the integration of a Demand Factor (DF), a dynamic multiplier that adjusts name prices in response to market demand. The DF is determined by comparing the total revenue in IO tokens from the current period to a moving average of revenues in a window of preceding periods. Depending on whether revenue is above, below, or equal to this average, the DF can increase or decrease. These DF maximum value is unbounded while the minimum is limited to prevent volatile reduction of prices; however, the minimum may be adjusted via step pricing (see Section 9.6.3).

ArNS names are designed to be both accessible and affordable, adjusting to market trends within a fair, maintenance-free pricing framework.

A detailed description of the variables and formulas used for dynamic pricing can be found in the Appendix.

9.6.1 PRICING SCENARIOS

There are several pricing models for leasing and purchasing names:

- **Leased Name, Instant Purchase:** Allows a user to lease a name for a certain duration and have it available for use immediately by the lessee.
- **Permanent Name, Instant Purchase:** Allows a user to purchase a name indefinitely and have it available for use immediately by the owner.
- **Leased Name, Auctioned:** Allows a user to lease a name for a duration but as the name is considered premium, it must go through an auction process before it can be acquired and used by the lessee.
- **Permanent Name, Auctioned:** Allows a user to purchase a name indefinitely but as the name is considered premium, it must go through an auction process before it can be acquired and used by the owner.

The table below summarizes the available purchase options for the name space:

Name Purchase Options		
Type	**Lease**	**Permanent**
1-4 Characters	Auction	Auction
5-12 Characters	Instant	Auction
13-51 Characters	Instant	Instant

Table 9.6.1: *Name Purchase Options*

9.6.2 DYNAMIC PRICING MECHANICS

Names are initially priced according to the Genesis Registration Fee (GRF), as set in the SmartWeave contract, with prices varying based on the length of the name. As the network's activity progresses, these fees give way to Base Registration Fees (BRF), which are modified by periodic step adjustments. The Demand Factor (DF) is a crucial component that dynamically scales prices, fluctuating with the network's revenue trends.

Revenue in the network accumulates within the Protocol Balance through various streams, such as instant name leases or purchases, auction completions, lease extensions, and under_name transactions. This cumulative revenue impacts the Demand Factor, which in turn influences the current name prices.

The DF is adjusted by comparing the recent period's revenue against a Revenue Moving Average (RMA) from the preceding seven periods. Based on this comparison, the DF can either increase, to reflect greater demand, or decrease, in response to diminished revenue, all within predetermined limits to prevent drastic fluctuations in pricing

The pricing system articulates various fees:

- The Adjusted Registration Fee (ARF) is the BRF modified by the DF.
- The Annual Fee is set as a proportion of the ARF.
- Instant Lease Registration and Permabuy prices are derived from the ARF, adding the calculated annual fees over the desired years.
- Auction parameters, including floor and ceiling prices for leases and permabuys, are directly influenced by the ARF.

The distinction between instant buy and auction for a name is determined based on its character length and whether it is a lease or permabuy. The auction process itself begins at a ceiling price – substantially higher than the floor – calculated as a multiple of the floor price which includes the ARF and relevant annual fees. As the auction progresses, the price decreases until a bid is placed or the floor is reached, determining the sale price.

The DF's adjustments are controlled by the network's recent revenue performance compared to the RMA. A rise in revenue results in an increase of the DF, reflecting stronger market demand, whereas a decline suggests reduced demand. This adaptive mechanism helps keep the pricing model in tune with real market activities.

Under_names are bundled with name registrations with additional ones available for purchase. The cost for extra under_names is a percentage of the current BRF, altered by the DF.

9.6.3 STEP PRICING MECHANICS

The dynamic model shall utilize a "Step Pricing" concept that acts as a stabilizing mechanism to counteract swift and dramatic market shifts, ensuring registration costs remain aligned, predictable, and not subject to volatile price decreases. Step pricing adjusts the Base Registration Fees when the Demand Factor reaches its minimum value for an extended period, updating the BRF to align with the current ARF, and resetting the DF to a neutral value. This allows for base prices to lower in extended droughts of low demand or high token value resulting in lower revenue generated to the protocol balance.

The below chart represents Step Pricing in action due to declining protocol revenue:

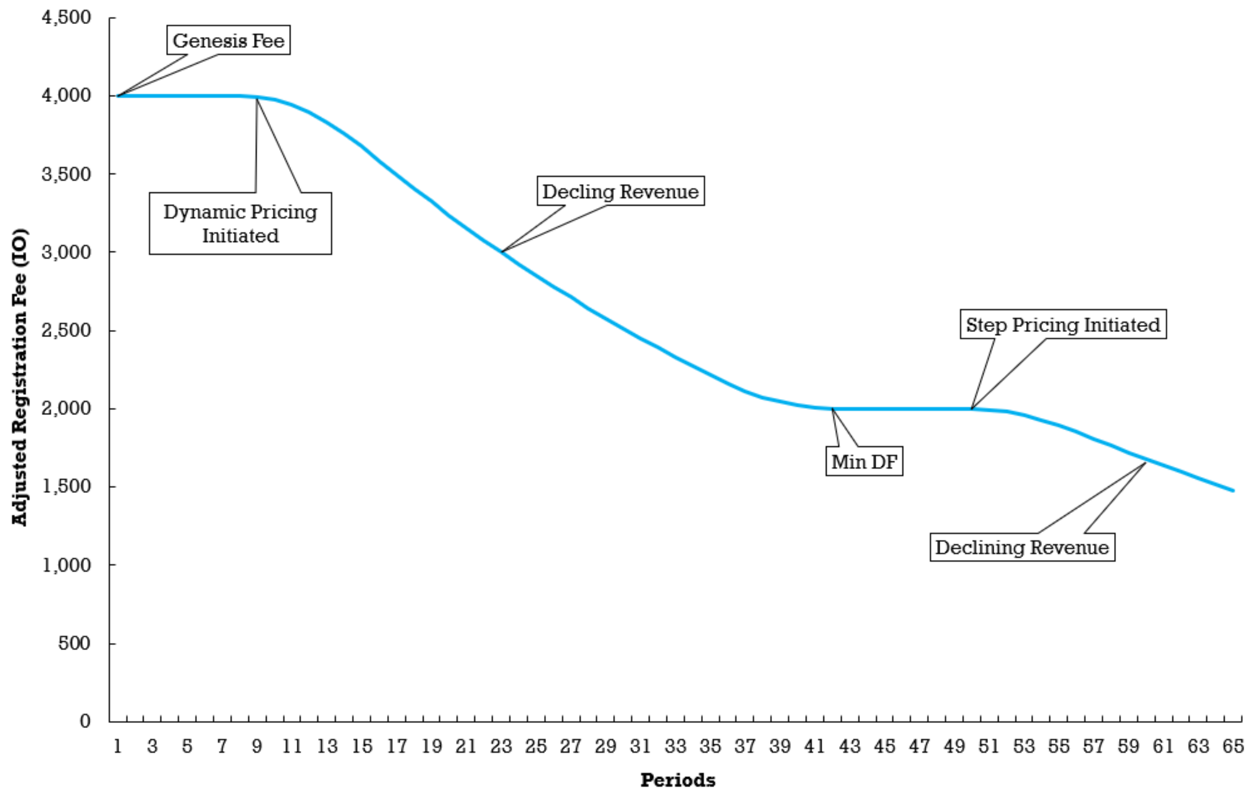

Diagram 9.6.3-1: *Step Pricing in Action – Declining Demand*

To contrast this, the below chart represents network conditions with continuously increasing protocol revenue with an unbounded Demand Factor:

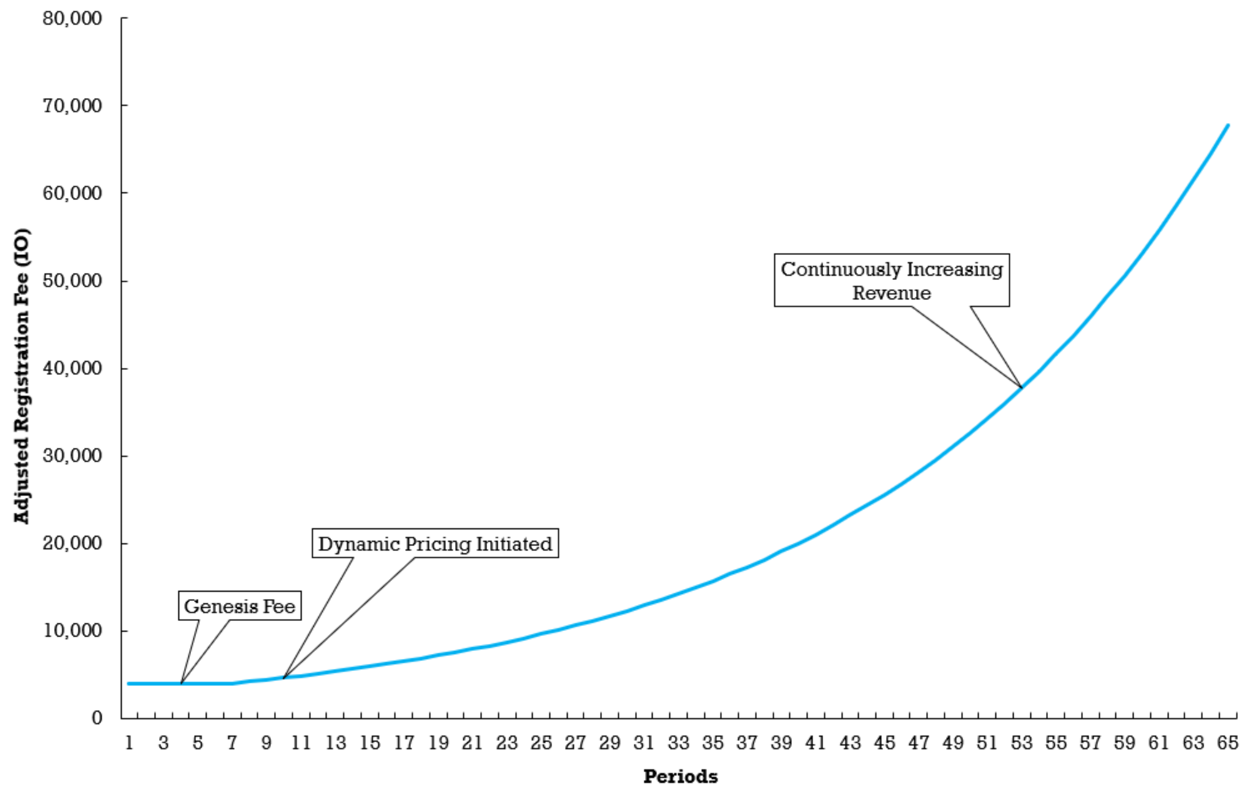

Diagram 9.6.3-2: *Dynamic Pricing in Action – Continuously Increasing Demand*

9.7 BID INITIATED DUTCH AUCTIONS (BIDA)

Auctions within ArNS serve a pivotal role, particularly for names considered premium by the community where market value can be highly variable. The auction format enables true market price discovery, allowing name price registration prices to reflect their real-time worth as determined by actual demand. As such, certain names must go to public auction based on their character length and purchase type. These auctions shall follow a Dutch Auction model whereby the first user interested in a name must initiate the process by placing the first bid.

The auction system works as follows: the first bid must be greater than or equal to the assigned floor price for the name. Once the first bid is placed, the timer for the auction begins. The auction begins at a price much higher than the floor price. As time passes, the purchase price progressively decreases until someone purchases it, or it hits the floor price, and the initial bidder receives the name.

This does not mean that the initial bidder must wait until the auction concludes. At any time, the initial bidder can place a second bid to purchase the name for the current purchase price, or a separate bidder can discover this auction and do the same. The benefit of this system, versus an English Auction system, is that there will only ever be 2 bids, the bid to initiate the auction, and the final bid to purchase. This makes for a more compact and scalable SmartWeave Contract state.

Auction end dates are denoted by Arweave block height and established at the start of the auction. For example, the duration for an auction could be 14 days or approximately 10,100 blocks.

9.7.1 AUCTION PRICE CURVE

For auctioned names, the progressively declining price of the name shall follow a power-law decay function:

$P(t) = \max(P_0 * (1 - (k * t))^p, P_{floor})$

where:

- `P(t), nameAuctionPrice` is the amount of IO tokens required to win a purchase name in an auction.
- `P₀, nameAuctionStartPrice` is the starting price of this auction.
- `t, elapsedAuctionTime` is the number of blocks that have elapsed since the start of the auction.
- `k, decayRate` determines how quickly the price decreases over time.
- `p, exponentVariable` determines the "shape" of the declining cost curve.
- `Pfloor, nameAuctionFloorPrice` is the amount of IO used as the floor price and minimum bid to start this name purchase auction.

The values of `k` and `p` shall be optimized so that the name price approaches the floor price (P_{floor}) by the end of the auction period.

Below is a representative auction curve resulting from this formula:

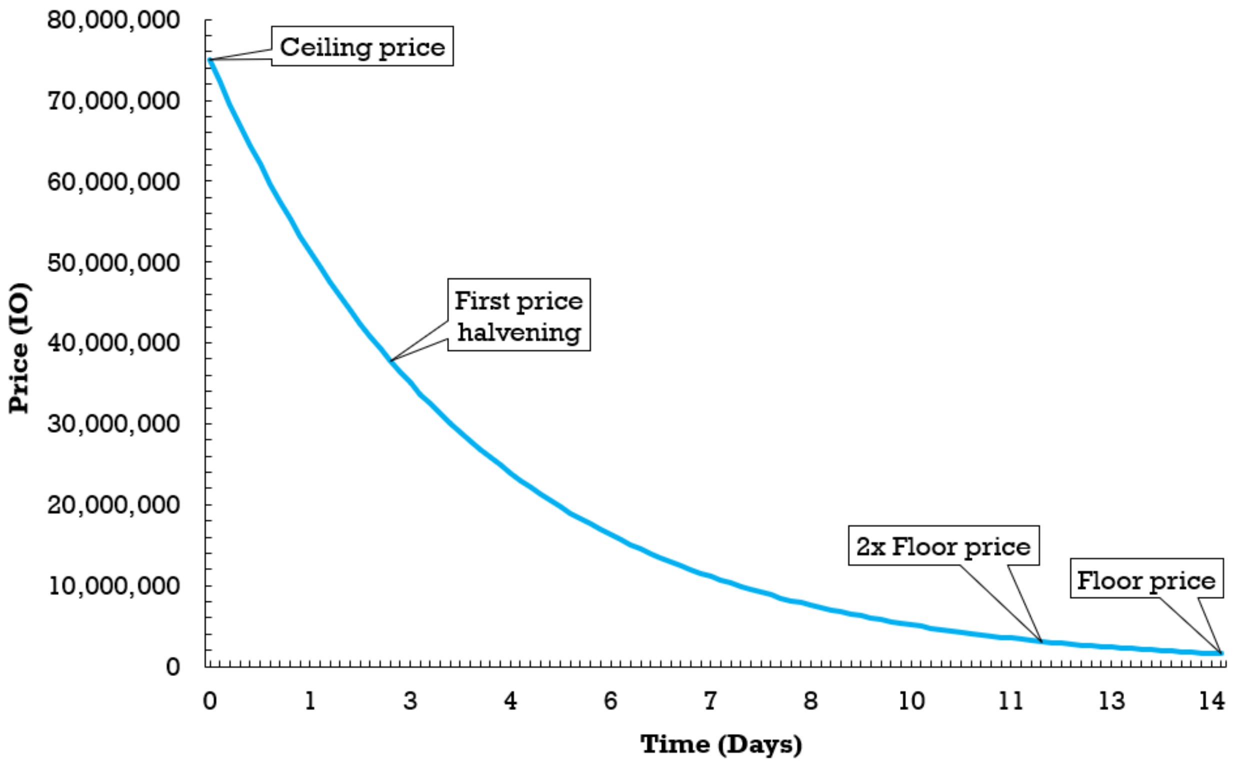

Diagram 9.7.1: *Example Auction Price Curve*

10 OBSERVATION AND INCENTIVES

10.1 OVERVIEW

The Observation and Incentive Protocol is designed to maintain and enhance the operational integrity of gateways on the AR.IO Network. It achieves this through a combination of incentivizing gateways for good performance and tasking those gateways to fulfill the role of "observers". The protocol is intentionally simple and adaptable, employing a smart contract-based method for on-chain "voting" to assess peer performance while being flexible on how that performance is measured. This setup permits gateway and observer nodes to experiment and evolve best practices for performance evaluation, all while operating within the bounds of the network's immutable smart contract, thus eliminating the need for frequent contract updates (forks).

In this protocol, observers evaluate their gateway peers' performance to resolve ArNS names. Their aim is to ensure each gateway in the network accurately resolves a subset of names and assigning a pass / fail score based on their findings.

A key component of the protocol is its reward mechanism. This system is predicated on gateway performance and compliance with observation duties. Gateways that excel are tagged as "Functional Gateways" and earn rewards, while those that do not meet the criteria, "Deficient Gateways" risk facing penalties – namely, the lack of rewards.

Funds for incentive rewards are derived from the protocol balance, which consists of IO tokens initially allocated at network genesis as well as those collected from ArNS asset purchases. Every epoch, this balance is utilized to distribute rewards to qualifying gateways and observers based on certain performance metrics.

10.2 OBSERVATION PROTOCOL

The Observation protocol is organized around epochs, periods of time that are broken into an observation reporting and tallying phase. The protocol is followed across each epoch, promoting consistent healthy network activity that can form pro-social behaviors and react to malicious circumstances.

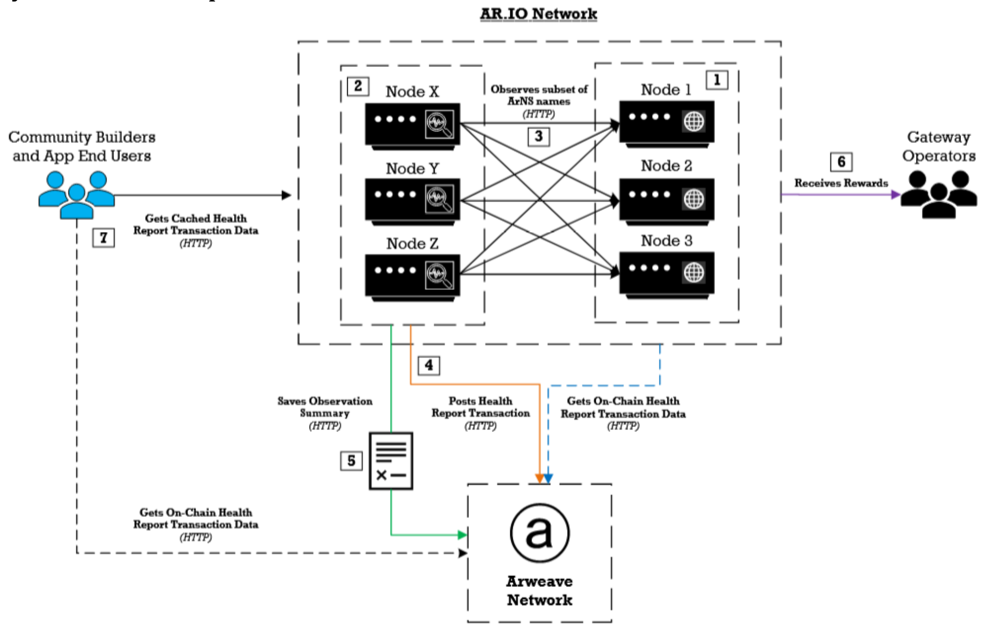
AR.IO Network
Community Builders and App End Users
Gets Cached Health Report Transaction Data (HTTP)
7
2
Node X
Node Y
Node Z
Observes subset of ArNS names (HTTP)
3
1
Node 1
Node 2
Node 3
6
Receives Rewards
Gateway Operators
4
Saves Observation Summary (HTTP)
Posts Health Report Transaction (HTTP)
Gets On-Chain Health Report Transaction Data (HTTP)
5
Gets On-Chain Health Report Transaction Data (HTTP)
Arweave Network

Diagram 10.2: *Observation and Incentive Protocol*

Diagram 10.2 Legend:

1. To participate in the epoch, a gateway must have already staked IO tokens and joined the network before it starts.
2. Each epoch (approximately 7 block-days), a random pool of active gateways will be selected (prescribed) to perform observation duties.
3. Within the epoch, observers are tasked with evaluating a subset of ArNS names for each gateway in the network.
4. By the end of the epoch's observation reporting period, the observer must upload its standardized health observation report to Arweave.
5. The observer must also submit a SmartWeave interaction to the AR.IO contract to save its report transaction ID and a summary of all failed gateways for tallying by the incentive protocol.
6. After the observation reporting period and tallying periods have closed, the payout is performed on the next contract state tick.
    a. This payout rewards gateways and observers who have performed their duties.
    b. Gateways that did not meet the performance threshold will not receive rewards.
    c. Observers that did not perform their duties are not rewarded and in addition, are penalized on any gateway rewards received.
7. Community builders and application users can verify and leverage the report and distribution information to make more informed decisions on which gateway to use.

10.3 ON-CHAIN REPORTS

The to-be-evaluated ArNS names include a set of names randomly determined by the protocol, known as "prescribed names", which are common across all observers within the epoch, as well as a set of "chosen names" picked at the discretion of each individual observer. "Prescribed names" are assigned to act as a common denominator / baseline while "chosen names" allow each observer to evaluate names that may be important to their operation.

Each observer shall assess the performance of the selected ArNS names (across all gateways) and summarize those findings in a report which details the following:

- **General Information**: Observer's Arweave address, starting and concluding block heights for the epoch.
- **Gateway Operator Assessment**: The expected and actual Arweave addresses of observed gateways, along with a summary verdict (pass or fail), and accompanying reasons for failure.
- **Detailed ArNS Evaluations**: For each gateway, it includes the domain name, evaluated ArNS names, the associated block height, transaction IDs, data hashes, a "pass or fail" score, reasons for failure (if any), and performance metrics like time to the first byte.

A comprehensive list of report criteria can be found in the Appendix.

Observers shall upload their completed reports (in JSON format) to the Arweave network as an on-chain audit trail. In addition, observers shall submit an interaction to the AR.IO SmartWeave contact detailing each gateway that they observed to have "failed" their assessments. These "votes" are tallied and used to determine the reward distribution.

Observer reports and interactions may be submitted up to the end of an epoch.

10.4 SELECTION OF OBSERVERS

The observer selection process commences at the beginning of each epoch and employs a random-weighted selection method. By combining random selection with weighted criteria like stake, tenure, and past rewards, the process aims to ensure both fairness and acknowledgment of consistent performance. This method allows for a systematic yet randomized approach to selecting gateways for observation tasks.

Criteria for Selection

Up to 50 gateways can be chosen as observers per epoch. If the GAR contains 50 or fewer gateways, then every gateway is designated as an observer for that epoch. If there are greater than 50, then randomized selection shall be utilized.

The weighted selection criteria will consider the following for each gateway:

- **Stake Weight (SW):** This factor considers how financially committed a gateway is to the network. It is the ratio of the total amount of IO tokens staked by the gateway (plus any delegated stake) relative to the network minimum and is expressed as:

```
SW = (Gateway Stake + Delegated Stake) / (Minimum Network Join Stake)
```

- **Tenure Weight (TW):** This factor considers how long a gateway has been part of the network, with a maximum value capped at four (4). This means that the maximum value is achieved after 2 block-years of participation in the network. It is calculated as:

```
TW = (Gateway Network Tenure) / (6 block-months)
```

- **Gateway Reward Ratio Weight (GRRW):** This factor is a proxy for a gateway's performance at resolving ArNS names. The weight represents the ratio of epochs in which a gateway received rewards for correctly resolving names relative to their total time on the network. To prevent division by zero conditions, it is calculated as:

```
GRRW = (1 + Passed Epochs) / (1 + Participated Epochs)
```

- **Observer Reward Ratio Weight (ORRW):** This factor is a proxy for a gateway's performance at fulfilling observation duties. The weight reflects the ratio of epochs in which a gateway, as an observer, successfully submitted observation reports relative to their total periods of service as an observer. To prevent division by zero conditions thus unfairly harming a newly joined gateway, it is calculated as:

```
ORRW = (1 + Submitted Epochs) / (1 + Selected Epochs)
```

Weight Calculation and Normalization

For each gateway, a composite weight (CW) is computed, combining the Stake Weight, Tenure Weight, Gateway Reward Ratio Weight, and Observer Reward Ratio Weight.

The formula used is:

```
CW = SW x TW x GRRW x ORRW
```

These weights are then normalized across the network to create a continuous range, allowing for proportional random selection based on the weighted scores. The normalized composite weight (N_CW) for each gateway indicates its likelihood of being chosen as an observer and is calculated by dividing the gateway's CW by the sum of all CWs. Any gateway with a composite weight equal to zero shall be ineligible for selection as an observer during the associated epoch.

Random Selection Process
The selection of observers is randomized within the framework of these weights. A set of unique random numbers is generated within the total range of normalized weights. For each random number, the gateway whose normalized weight range encompasses this number is selected. This system ensures that while gateways with higher weights are more likely to be chosen, all gateways maintain a non-zero chance of selection, preserving both fairness and meritocracy in the observer assignment process. The current epoch's selected / prescribed observers shall be saved in the contract state to ensure that any activities during that epoch do not affect the selection of observers.

10.5 PERFORMANCE EVALUATION

Consider the following classifications:

- **Functional or Passed Gateways:** are gateways that meet or surpass the network's performance and quality standards.
- **Deficient or Failed Gateways:** are gateways that fall short of the network's performance expectations.
- **Functional or Submitted Observers:** are selected observers who diligently perform their duties and submit observation reports and contract interactions.
- **Deficient or Failed Observers:** are selected observers who do not fulfill their duty of submitting observation reports and contract interactions.

At the end of an epoch, the smart contract will assess the results from the observers and determine a pass / fail score for each gateway:

- If greater than or equal to 50% of submitted observer contract interactions indicate a PASS score, then that gateway is considered Functional and eligible for gateway rewards.
- Else, if greater than 50% of submitted observer contract interactions indicate a FAIL score, then that gateway is considered Deficient and ineligible for gateway rewards.

These results will determine how reward distributions are made for that epoch. Rewards shall be distributed after 50 blocks in the following epoch have elapsed. This delay ensures that all observation contract interactions are safely confirmed by the Arweave network without risk of "forking out" prior to the evaluation and reward distribution process.

10.6 REWARD DISTRIBUTION

Each epoch, **0.25%** of the protocol balance is earmarked for distribution as rewards. From this allocation, two distinct reward categories are derived:

1. **Base Gateway Reward (BGR)**: This is the portion of the reward allocated to each Functional Gateway within the network and is calculated as:

   ```
   BGR = [Epoch Reward Allocation x 95% / Total Gateways in the Network]
   ```

2. **Base Observer Reward (BOR)**: Observers, due to their additional responsibilities, have a separate reward calculated as:

   ```
   BOR = [Epoch Reward Allocation x 5% / Total Selected Observers for the Epoch]
   ```

Distribution Based on Performance
The reward distribution is contingent on the performance classifications derived from the Performance Evaluation:

- **Functional Gateways**: Gateways that meet the performance criteria receive the Base Gateway Reward.
- **Deficient Gateways**: Gateways falling short in performance do not receive any gateway rewards.
- **Functional Observers**: Observers that fulfilled their duty receive the Base Observer Reward.
- **Deficient Observers**: Observers failing to meet their responsibilities do not receive observer rewards. Furthermore, if they are also Functional Gateways, their gateway reward is reduced by **25%** for that epoch as a consequence for not performing their observation duty.

Gateways shall be given the option to have their reward tokens "auto-staked" to their existing stake or sent to their wallet as unlocked tokens. The default setting shall be "auto-staked".

Distribution to Delegates
The protocol will automatically distribute a Functional Gateway's shared rewards with its delegates. The distribution will consider the gateway's total reward for the period (including observation rewards), the gateway's "Delegate Reward Share Ratio", and each delegate's stake proportional to the total delegation. Each individual delegate reward is calculated as:

$$DR_i = \text{Total Rewards} \times \text{Reward Share Ratio} \times (\text{Delegate's Stake} / \text{Total Delegated Stake})$$

Token reward distributions to delegates will be "auto-staked" in that they will be automatically added to the delegate's existing stake associated with the rewarded gateway. The delegated staker is then free to withdraw their staked rewards at any time (subject to withdrawal delays).

Undistributed Rewards
In cases where rewards are not distributed, either due to the inactivity or deficiency of gateways or observers, the allocated tokens shall remain in the protocol balance and carry forward to the next epoch. This mechanism is in place to discourage observers from frivolously marking their peers as offline in hopes of attaining a higher portion of the reward pool.

10.7 HANDLING INACTIVE GATEWAYS

To maintain network efficiency and reduce contract state bloat, gateways that are offline for six (6) consecutive epochs, and thus fail to receive rewards, will automatically trigger a "Network Leave" action and be subject to the associated stake withdrawal durations for both gateway stake and any delegated stake.

11 OUTRO AND FUTURE VISION

11.1 OUTRO

This white paper has presented a comprehensive network design that includes numerous features aimed at making Arweave more accessible and attractive to gateway operators. The AR.IO Network's incentivization mechanism, backed by the IO Token, represents value, trust, and reputation to participants.

The token has various utilities within the network, including:

- **Gateways:** Gateway operators must stake a minimum amount of IO tokens to join their gateway to the network. Operators may opt to enlarge their staked balance above the minimum to increase their gateway's visibility on the network and chances to enforce its protocols.

- **Delegated Staking:** Users with IO tokens can participate in delegated staking by allocating their tokens to gateway operators. This process increases the operator's stake, potentially enhancing their visibility and influence within the network. It also allows general users to share in the rewards of gateway operations.

- **ArNS:** Users must spend IO tokens to register ArNS friendly names, extend leases, and increase undernames. The proceeds from these expenditures then circle back to gateway operators and delegates through the reward protocol.

Overall, the AR.IO Network's incentivization mechanism is designed to reward healthy gateway operators and active participants while discouraging poor performers and malicious actors. The system incorporates immutable smart contract elements to ensure economic and namespace integrity while allowing flexibility and experimentation for future adaptations. This completes the main objective: defining the framework for a healthy and sustainable decentralized gateway network.

11.2 FUTURE VISION

The previous sections of this paper offer a feature complete network with the ability to continually grow and adapt to user needs. But this paper only scratches the surface of the exciting innovations that can be created by the community. The following examples are presented to the community as potential approaches for further decentralizing and scaling the AR.IO Network into the future:

	Future Enhancements	
Item	**Gap Description**	**Potential Approach**
SmartWeave Transactions	SmartWeave support for Arweave base layer transactions limits the total throughput to that only of the Arweave chain, which is capped by the protocol at 1,000 transactions per block, roughly every 2 minutes.	Support can be added for sequenced, bundled SmartWeave transactions made to the IO contract along with a state resolution service that provides fast access to the latest state information.

Future Enhancements		
Item	**Gap Description**	**Potential Approach**
Data Searching	AR.IO Gateway Architecture does not support dynamically searching across the Arweave network for transaction header and chunk information if it is not found within the specific nodes trusted by the gateway.	A back-end process can be added to automatically scan the Arweave network and collect data that is missing on AR.IO.
Gateway Revenue	Gateway operators need a revenue model to sustain their operations and maximize the financial benefits of running a gateway.	Implement a gateway account management system that allows operators to charge for services like bandwidth usage, index querying, and advanced features, providing a revenue stream to support and incentivize gateway operation.
ar:// protocol (ARCSS)	Decentralized applications require resiliency and censorship resistance across the top-level domains used by the individual AR.IO gateway nodes.	A potential approach could be a protocol that abstracts top level domain names from permaweb data. The ar:// schema protocol was designed to seamlessly route requests for ArNS Names, base layer transactions, and bundled data items without the user providing a top-level domain.
Third Party Integrations	Gateways may rely on external services and Layer 2 networks, such as instant finality for bundling networks, of which they have no control over and whose quality-of-service issues may cause downstream impacts to their users as well as result in potential gateway penalties.	Two potential approaches would be for the gateway operator to establish a contractual agreement and Service Level Agreement (SLA) with its underlying service providers. Alternatively, a protocol that allows these external providers to take on risk, register their service and stake IO tokens, could subject them to the same incentivization and penalty mechanisms as gateways for providing a quality-of-service.
Empowering Content Creators	Content creators require effective ways to monetize their contributions to the permaweb.	Introduce a decentralized ad-serving mechanism within gateways, allowing content creators to generate revenue through ads displayed alongside their content on the AR.IO Network
User Verification of Gateway Performance	The proposed protocol relies on a gateway's peers to assess its performance. This does not offer a way for the experience of end users to be considered when determining protocol reward distributions.	Incorporate a system that enables users to provide feedback on how well a gateway performs. An approach could be through a vouching system whereby users can signal on-chain whether they have had a positive experience with a certain gateway or perhaps through advertising the number of vouched delegates a gateway has amassed. This system could promote more feature rich and performant gateway operations.

Table 11.2: *Future Enhancements*

12 APPENDIX

12.1 REFERENCES AND FURTHER READING

The following resources were used as reference material for this section and can provide the interested reader with additional information:

- **AR.IO White Paper v1.0:** https://v1-0-0_whitepaper_ar-io.arweave.dev
- **Arweave Draft 17:** https://draft-17.ar-io.dev/
  (txID = azo-0qw6bb9u5doGdMR-atcIRV_ylJCV4K4Kwv85GO4)
- **The Arwiki:** https://arwiki.wiki/
- **Arweave GitHub repository:** https://github.com/ArweaveTeam
- **Warp Academy:** https://academy.warp.cc/
- **Arweave 2.6 Specification:** https://2-6-spec.ar-io.dev/
  (txID = kk_T_k8VAk6b_mAN7sWq1lBLSNaYxcyL7phtt4l1Nyo)
- **AR.IO Network Software**
  - **AR.IO Core:** https://github.com/ar-io/ar-io-node
  - **AR.IO Observer:** https://github.com/ar-io/ar-io-observer

12.2 GLOSSARY

Many novel terms and acronyms are used by the Arweave ecosystem as well as some new ones introduced in this paper. The list below is intended to serve as a non-exhaustive reference of those terms:

- **Arweave Name System (ArNS):** a decentralized and censorship-resistant naming system enabled by AR.IO gateways which connects friendly names to permaweb applications, pages, and data.
- **Arweave Name Token (ANT), "Name Token":** a SmartWeave based token, that is connected to each registered ArNS Name. Each ANT gives the owner the ability to update the subdomains and Arweave transaction IDs used by the registered name as well as transfer ownership and other functions.
- **Arweave Network Standards (ANS):** Drafts and finalized standards for data formats, tag formats, data protocols, custom gateway features and anything that is built on top the Arweave Network. Specific standards are denoted by an associated number, e.g., ANS-###.
- **Base Layer Transaction:** refers to one of up to 1,000 transactions that make up a single Arweave block. A base layer transaction may contain bundled data items.
- **Bundle, bundling:** an Arweave concept introduced in ANS-104 that allows for a way of writing multiple independent data transactions into one base layer transaction. Bundled transactions contain multiple independent transactions, called data items, wrapped into one larger transaction. This offers two major network benefits:
  - A scaling solution for increasing the throughput of uploads to the Arweave network,
  - Allows delegation of payment for an upload to a third party, while maintaining the identity and signature of the person who created the upload, without them needing to have a wallet with funds.
- **Bundled Data Item (BDI):** A data item / transaction nested within an ANS-104 bundled transaction.
- **Bundler:** a third-party service that bundles data files on a user's behalf.
- **Chunk:** A chunk is a unit of data that is stored on the Arweave network. It represents a piece of a larger file that has been split into smaller, manageable segments for efficient storage and retrieval.
- **Decentralized, decentralization, etc:** a nonbinary, many axis scale enabling a system or platform to be: permissionless, trustless, verifiable, transparent, open-source, composable, resilient, and censorship resistant. Ultimately, something that is decentralized is not prone to single points of failure or influence.
- **Epoch:** a specific duration (e.g., one block-week) during which network activities and evaluations are conducted. It serves as a key time frame for processes such as observation duties, performance assessments, and reward distributions within the network's protocols.

- **Gateway:** a node operating on the Arweave network that provides services for reading from, writing to, and indexing the data stored on the permaweb. Sometimes referred to as "permaweb nodes".
- **Gateway Address Registry (GAR):** a decentralized directory maintained in the AR.IO SmartWeave Contract. It serves as the authoritative list of all registered gateways on the AR.IO Network. The registry provides detailed metadata about each gateway to facilitate discovery, health monitoring, and data sharing among permaweb apps and users. The GAR is designed to be easily queryable, sortable, and filterable by end users and clients, allowing for tailored selections based on various criteria to meet specific use cases.
- **Indexing:** The act of organizing transaction data tags into queryable databases.
- **Layer 2 Infrastructure:** Layer 2 refers to the technology / infrastructure stack built "above" a base layer. In this use, the AR.IO Network would be considered Layer 2 infrastructure to the base Arweave protocol.
- **Manifest (aka Path Manifest, Arweave Manifest):** special "aggregate" files uploaded to Arweave that map user-definable sub-paths with other Arweave transaction IDs. This allows users to create logical groups of content, for example a directory of related files, or the files and assets that make up a web page or application. Instead of having to manually collate these assets, manifests group them together so that an entire website or app can be launched from a single manifest file. Gateways can interpret this structure, so that users can then reference individual transactions by their file name and/or path.
- **Mempool:** short for "memory pool," is a component of Arweave mining nodes that temporarily stores valid transactions that have been broadcasted to the network but have not yet been added to a block.
- **Miner (aka Arweave Node):** a node operating on the Arweave network responsible for data storage and recall.
- **Observer:** a gateway selected to evaluate the performance of peer gateways in resolving ArNS names. Observers assess and report on the operational efficacy of other gateways.
- **Period:** refers to a predefined time span (e.g., a block-day) that serves as a cycle for network activities such as dynamic pricing. It is a fundamental unit of time for operational and protocol processes within the network.
- **Permaweb:** The permaweb is the permanent and decentralized web of files and applications built on top of Arweave.
- **Permaweb Service Token (PST):** An alternate term used within the Arweave ecosystem to denote a SmartWeave Token (SWT).
- **Protocol Balance:** The primary sink and source of IO tokens circulating through the AR.IO Network. This balance is akin to a central vault or wallet programmatically encoded into the network's smart contract from which ArNS revenue is accumulated and incentive rewards are distributed.
- **Protocol Rewards:** IO Token incentive rewards distributed by the protocol to the network's eligible users and gateway operators.
- **Seeding:** Refers to the act of propagating new data throughout the network. Miner nodes seed Arweave base layer transaction data to other miners, while gateways ensure that the transactions they receive reach the Arweave nodes. Both gateways and Arweave nodes seed base layer transactions and data chunks.
- **Staking (of tokens):** Refers to the process of locking IO tokens into a protocol-controlled vault, temporarily removing them from circulation until unlocked. This action represents an opportunity cost for the staker and serves as a motivator to prioritize the network's collective interests.
- **SmartWeave:** Arweave's smart contract protocol.
- **SmartWeave Token (SWT):** A smart contract token built on SmartWeave.
- **Transaction ID (txID):** Every transaction and data file uploaded to Arweave is assigned a unique identifier code known as the Transaction ID. These txID's can be referenced by users to easily locate and retrieve files.
- **Trust-minimization:** Relates to enacting network security by minimizing the number of entities and the degree to which they must be trusted to achieve reliable network interactions. A network with trust-minimizing mechanisms means that it has reduced exposure to undesirable third-party actions and built-in incentives to reward good behavior while punishing bad behavior.
- **Vault:** Token vaults are protocol level mechanisms used to contain staked tokens over time. Each vault contains a starting block height, ending block height (if applicable), along with a balance of tokens.

12.3 ARNS PRICING SPECIFICATION

The following is a technical description of the ArNS Dynamic Pricing and Auction mechanics.

Definitions:

- **Genesis Registration Fee (GRF):** Starting price for name registrations varies by character length. Superseded by Base Registration Fees as the protocol evolves. Suggested genesis fees are outlaid in the table below:

Genesis Registration Fees	
Name Length	**Fee (IO)**
1	5,000,000
2	500,000
3	100,000
4	25,000
5	10,000
6	5,000
7	2,500
8	1,500
9	1,250
10	1,250
11	1,250
12	1,250
13-51	1,000

- **Base Registration Fee (BRF):** The fundamental price for names, varying by character length, adjusted periodically.
- **Demand Factor (DF):** A global price multiplier, reflecting namespace demand, adjusted each period based on revenue trends.
- **Protocol Revenue:** Accumulated IO tokens from name purchases/leases, auction closures, lease extensions, and under_name sales.
- **Revenue Moving Average (RMA):** The average of protocol revenue from the past (7) periods.
- **Period (P):** The time unit for DF adjustments, equivalent to (1) block-day.
- **n:** The current period indicator.
- **Price:** The cost for instant permabuy or lease of a name.
- **Under_names:** Subdomain equivalents, denoted by an underscore "_" prefixing the base domain.

General Pricing Criteria:

- **Adjusted Registration Fee (ARF):** ARF = BRF x DF
- **Annual Fee:** Annual Fee = ARF x 20%
- **Leases:**
  - **Instant Lease Registration Price:** Instant Lease Price = ARF + (Annual Fee x years)
  - **Lease Auction Floor:** Lease Floor = ARF + Annual Fee
  - **Lease Auction Start (Ceiling):** Lease Ceiling = Lease Floor x 50
  - **Lease Extension / Renewal Price:** Lease Renewal Price = Annual Fee x years (max 5 years)
- **Permanent Purchases:**
  - **Instant Permabuy Price:** Permabuy Price = ARF + (Annual Fee x 10 years)
  - **Permabuy Auction Floor:** Permabuy Floor = ARF + (Annual Fee x 10 years)
  - **Permabuy Auction Start (Ceiling):** Permabuy Ceiling = Permabuy Floor x 50

Under_name Fee:

- **Initial Allocation:** 10 under_names included with each name registration.
- **Additional Purchases:**
    - **For Leases:** Lease Under_name Fee = BRF x DF x 0.1%
    - **For Permabuys:** Permabuy Under_name Fee = BRF x DF x 0.5%
- **Maximum Under_names:** Up to 10,000 per ArNS name.

Auction vs. Instant Buy Criteria:

- Names offered via instant purchase or auction vary based on character length and transaction type:

Name Purchase Options		
Type	**Lease**	**Permanent**
1-4 Characters	Auction	Auction
5-12 Characters	Instant	Auction
13-51 Characters	Instant	Instant

Bid Initiated Dutch Auction Mechanism:

- Auction duration capped at 2 block-weeks, starting at the ceiling price and decreasing until a bid is placed above the floor.
- If no counterbids are placed within the auction window, then the auction goes to the initiator at the floor price.
- **Auction Price Decay Function:** For auctioned names, the progressively declining price follows a power-law decay function:

```
P(t) = max(P0 * (1 - (k * t))^p, Pfloor)
```

  - *P(t), nameAuctionPrice:* Amount of IO tokens required to win a name purchase in an auction.
  - *P_0, nameAuctionStartPrice:* Starting price of the auction.
  - *t, elapsedAuctionTime:* Number of blocks since the start of the auction.
  - *k, decayRate:* Rate at which the price decreases over time (e.g., `0.000002`).
  - *p, exponentVariable:* Determines the shape of the declining cost curve (e.g., `190`).
  - *P_{floor}, nameAuctionFloorPrice:* Minimum bid to start the name purchase auction.

Demand Factor (DF) Dynamics:

- Adjusts based on protocol revenue comparison to the RMA.
- **Mechanics:**
    - **Increase DF:** When recent revenue is higher than or equal to (but non-zero) the RMA, DF increases by 5.0%.
    - **Decrease DF:** When recent revenue is less than the RMA or both are zero, DF decreased by 2.5%.
- **Limits on DF:**
    - **Maximum Value:** Unbounded.
    - **Minimum Value:** 0.5 (Prices halve from base at least).

Step Pricing:

- Synchronizes BRF with ARF after 14 consecutive periods at the DF floor.
- Resets DF to 1 following a step pricing adjustment.

12.4 OBSERVER REPORT DETAILS

Each observer shall assess the performance of the selected ArNS names (across all AR.IO gateways) and summarize those findings in a report which details the following:

General information:

- The observer's Arweave address.
- The starting block height of the epoch.
- The block height at which the report was generated.

Overall Gateway Operator assessment:

- Gateway FQDN.
- The Arweave address that the observer expects to be the owner / operator of the gateway.
- The Arweave address that the observed gateway actually reports.
- A final "pass or fail" rollup determination for each observed gateway.
- Failure reason (if applicable).

ArNS assessments:

- Observed ArNS name (for all prescribed and chosen names).
- The block height at which the name was assessed.
- The expected status code.
- The resolved status code.
- The transaction ID that the observer expects the associated name to resolve to.
- The transaction ID that the gateway actually resolves to.
- The data hash that the observer expects the associated name to resolve to.
- The data hash that the gateway actually resolves to.
- The "pass or fail" score associated with the observed name, at the observer's discretion.
- Failure reason (if applicable).
- Timing / performance information associated with the name resolution such as time to first byte and total duration.

The above is repeated for the entire name pool and across each gateway in the GAR.

Example Observation Report:
https://arweave.net/GG1YCFc7wQxKvQ1qD1lTEp2OAMBs4VzrpfdmeeLyjDI

Note that reports may be compressed in accordance with network standards to reduce data size.