

Network + Token

White Paper

CRYP2NOVA

v3.0.0
May 5, 2026

Archived copy · cryp2nova.com

Table of Contents

TABLE OF CONTENTS 2
1 INTRODUCTION 4
1.1 Why ar.io? 4
1.2 What is ar.io? 4
1.3 TL;DR 5
1.4 Paper Outline 5
2 THE FOUNDATION 6
2.1 Overview 6
2.2 Guiding Philosophy 6
3 NETWORK COMPOSITION 7
3.1 Overview 7
3.2 Storage 7
3.3 Access 8
3.4 Application and Utility Layer 8
4 THE AR.IO SMART CONTRACT 10
4.1 Overview 10
4.2 Solana Integration 10
4.3 Gas Fees 11
4.4 Contract State 12
4.5 Time-Dependent State 12
4.6 Settings 15
4.7 Protocol Balance 15
4.8 Contract Immutability and Upgrades 15
5 THE ARIO TOKEN 17
5.1 Overview 17
5.2 Supply and Allocation 17
5.3 Token Unlock Schedule 18
5.4 Community Distribution 19
5.5 Initial Protocol Reward Balance 19
6 STAKING 20
6.1 Overview 20
6.2 Gateway Staking 20
6.3 Delegated Staking 21
6.4 Stake Redelegation 22
6.5 Redeeming Stake for ArNS 23
6.6 Expedited Withdrawal Fees 24
7 GATEWAY ARCHITECTURE 25
7.1 Overview 25
7.2 Gateway Functions 25
7.3 Gateway Modularity 26
7.4 ArNS Indexing and Routing 27
7.5 Content Moderation 27
8 GATEWAY NETWORK 28
8.1 Overview 28
8.2 Gateway Address Registry (GAR) 28
8.3 Data Sharing 29
8.4 Data Verification 30
9 AR.IO NAME SYSTEM (ARNS) 33
9.1 Overview 33
9.2 Name Registration 34
9.3 Primary Names 36
9.4 Ar.io Name Token (ANT) 37
9.5 Addressing Variable Market Conditions 39
9.6 Dynamic Pricing Model 39
9.7 Returned Name Premiums (RNP) 42
9.8 Gateway Operator ArNS Discount 42
10 OBSERVATION AND INCENTIVES (OIP) 43
10.1 Overview 43
10.2 Observation Protocol 43
10.3 Onchain Reports 44
10.4 Selection of Observers 45
10.5 Performance Evaluation 46
10.6 Reward Distribution 47
10.7 State Snapshot 48
10.8 Handling Deficient Gateways 48
11 OUTRO 49
12 APPENDIX 50
12.1 References and Further Reading 50
12.2 Glossary 50
12.3 ArNS Pricing Specification 53
12.4 Observer Report Details 55
12.5 Major Revisions 56

CRYP2NOVA

ar.io

Archived copy · cryp2nova.com

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**
- **Jonathan Policke**
- **Ariel Melendez**
- **David Whittington**
- **Dylan Fiedler**
- **Tim Bukher**

In addition, the Foundation warmly thanks everyone who provided invaluable input and “fresh eyes” during the history 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.

CRYP2NOVA

ar.io

Archived copy · cryp2nova.com

1 Introduction

1.1 Why ar.io?

The internet was designed to share information, not to preserve it. Web pages disappear. Platforms shut down. Content is altered or removed without notice. There is no structural guarantee that data available today will remain unchanged or accessible tomorrow.

Arweave solved the storage layer by creating a permanent, immutable, decentralized data store with an endowment model that sustains replication for hundreds of years. At the protocol level, the storage problem is addressed.

Storage alone, however, is not a complete solution. Permanent data that cannot be found, accessed, verified, or resolved to a human-readable address provides limited practical value. As AI systems begin relying on retrieved data for production decisions, as content provenance becomes a regulatory concern, and as platforms continue to sunset without preserving what they hosted, the need for a reliable access and verification layer over permanent storage grows.

Ar.io is that layer. The remainder of this paper describes how it works.

1.2 What is ar.io?

Note: The ar.io network mainnet launched on February 20, 2025, and v3.0.0 (described in this paper) hard-forks the protocol to Solana. All dates, token unlock schedules, demand factors, and other time-dependent values referenced herein are measured relative to that original launch date.

Ar.io is the world's first permanent cloud network, providing the infrastructure to ensure data, applications, and digital identities are timeless, tamper-proof, and universally accessible. Built on the foundation of the Arweave storage network, ar.io forms a global ecosystem of gateways, protocols, and services that connect users to the permaweb – a web where information is permanent and free from centralized control.

The Ar.io Network is an open, distributed, and ownerless system, supported by operators, developers, and end-users from around the world. Its decentralized nodes, known as AR.IO Gateways, act as "Permanent Cloud Service Providers" delivering the critical services needed to read, write, index and query data stored on the permaweb. These gateways provide a unified, resilient interface between users and the permaweb, featuring a permanent domain name system and seamless, location-independent access to permanent storage and applications.

Gateways operate using standardized protocols to maintain consistency across the network. They also engage in an observation and reporting protocol to monitor performance and ensure accountability, helping to maintain a healthy and reliable ecosystem.

The Ar.io Network is powered by a utility token, ARIO, which drives the network's functionality and accessibility. ARIO serves as a currency for services such as the Ar.io Name System (ArNS), staking to join the network as a gateway operator, delegated staking, and as rewards for contributing to the network's performance and reliability.

Archived copy · cryp2nova.com

Together, these elements form the backbone of a permanent cloud network designed to preserve data and expand the possibilities of the web.

1.3 TL;DR

This white paper details a decentralized and incentivized cloud network aimed at attracting more gateways to the Arweave network therefore making the permanent web more accessible to all. At the core of ar.io's incentivization mechanism is the ARIO Token, an SPL Token on Solana used as a utility token for joining the network, payments, gateway accountability, and protocol incentives. The network features modular and composable gateway infrastructure in addition to the Ar.io Name System (ArNS) – a system for assigning friendly domain names to permanent 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:** Defines the **Network Composition**, including storage, compute, and service layers that make up ar.io.
- **Section 4:** Describes the ar.io **Smart Contract**.
- **Section 5:** Introduces the **ARIO Token**, its functions, and its tokenomics.
- **Section 6:** Describes the critical role of **staking** in 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 **Ar.io 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.
- **Section 12:** Contains a **glossary** and other **appendix** information.

Archived copy · cryp2nova.com

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 ARIO 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.

Archived copy · cryp2nova.com

3 Network Composition

3.1 Overview

The ar.io network operates as a two-layer stack with applications built on top:

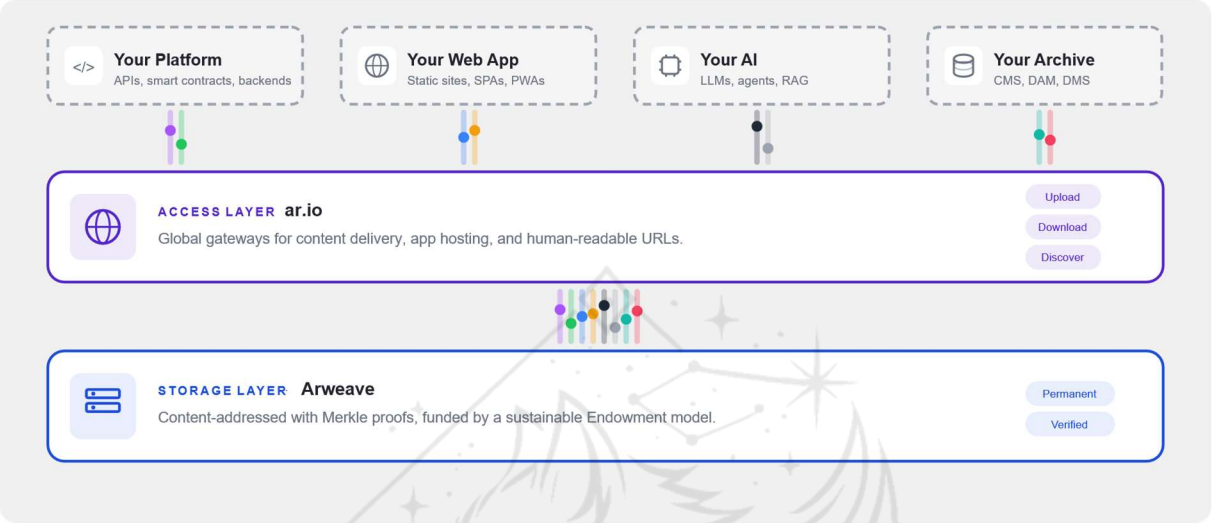


Diagram 3.1: *The Permanent Cloud Stack*

1. **Storage Layer: Arweave**
   provides content-addressed, immutable data storage with Merkle proofs, funded by a sustainable and protocolized endowment model.

2. **Access Layer: ar.io**
   provides global gateways for content delivery, data retrieval, indexing, verification, app hosting, and human-readable naming.

3. **Application Layer**
   is what users and developers build on top: platforms, web applications, AI systems, content management, and digital archives.

Arweave handles permanence. Ar.io handles access. Applications leverage both through standard protocols and SDKs. The following sections describe each layer.

Ar.io integrates decentralized protocols, services, and applications to power the permanent web alongside the traditional internet. Foundational components like Arweave and Solana are independently developed, while ar.io introduces essential services and incentives that enable seamless interaction and accessibility.

3.2 Storage

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

Archived copy · cryp2nova.com

Data stored on Arweave is immutable and globally replicated by a network of miner nodes. Unlike traditional blockchains that link blocks in a linear sequence, Arweave uses a "blockweave" structure that connects blocks into a web for efficient storage and retrieval. Miners secure this blockweave by running the Succinct Proof of Random Access (SPoRA) algorithm, which requires them to prove they can access randomly selected data to produce new blocks. Successful miners are rewarded in Arweave's native AR token, derived from transaction fees and the network's storage endowment, a protocol-controlled pool of tokens designed to fund storage costs for 200+ years.

Arweave is file type agnostic, supporting everything from text files to web applications, databases, and multimedia archives. Uploading data requires a one-time payment in AR proportional to file size. There are no ongoing fees. Once stored, data persists without user intervention.

Bundling extends the protocol's throughput by allowing multiple signed data items to be included in a single transaction. These bundles follow the ANS-104 standard, which supports multiple signature types from various blockchains including Arweave, Solana, and Ethereum. This cross-chain signature support allows users to upload and sign data using wallets from their preferred ecosystem.

3.3 Access

The Access Layer connects users and applications to permanent data through a decentralized network of gateway nodes. Gateways are specialized nodes that optimize for usability and performance, distinct from Arweave mining nodes that secure the blockweave. The gateway software is open source and permissionless to operate. Anyone can run one.

Gateways provide several core functions, including data retrieval and caching, transaction indexing and querying, data verification with cryptographic proofs, human-readable name resolution through the Ar.io Name System (ArNS), and application hosting. With RFC 9421 HTTP Message Signatures, gateways can also cryptographically sign their responses, giving clients non-repudiable records of what was served and by whom.

Gateways serve Arweave data directly and operate independently of the execution layer. If Solana experiences downtime, gateways continue serving data and resolving names from local state. This separation between data serving and protocol coordination is a deliberate architectural choice.

Gateway operators are sustained through ARIO token rewards and are held accountable through a peer observation protocol that monitors performance across the network (described in Section 10). The ar.io protocol executes on Solana, which provides the execution layer for staking, naming, observation, and reward distribution. The protocol architecture is described in Section 4.

3.4 Application and Utility Layer

The Application Layer is what developers, users, and agents build on top of permanent storage and decentralized access. Applications on ar.io fall into several categories:

Archived copy · cryp2nova.com

- **Platforms and backends**
  APIs, smart contracts, and server-side applications that use Arweave for permanent data storage and ar.io for retrieval and naming. These range from content publishing systems to data pipelines that require immutable records.

- **Web applications**
  Static sites, single-page applications, and progressive web apps hosted permanently on Arweave and served through ar.io gateways. Once deployed, these applications remain accessible without ongoing hosting costs or server maintenance.

- **AI systems**
  Large language models, autonomous agents, and retrieval-augmented generation (RAG) pipelines that consume or produce data requiring verifiable provenance. Ar.io's signed gateway responses (RFC 9421) provide audit trails for data retrieved by AI systems, and permanent storage ensures training data and model outputs remain available and attributable.

- **Archives**
  Content management systems, digital asset management platforms, and document management systems that use permanent storage for long-term preservation. These applications benefit from Arweave's immutability guarantees and ar.io's indexing and retrieval services.

Developers interact with the permanent cloud stack through the ar.io SDK, which abstracts the underlying protocol operations into standard read and write interfaces. The SDK supports both Solana and Arweave interactions, allowing applications to register names, upload data, query gateway state, and verify responses through a single package.

CRYP2NOVA

Archived copy · cryp2nova.com

4 The Ar.io Smart Contract

4.1 Overview

The ar.io smart contract encompasses all the functionality required to support the network's currency, utilities, and management. Written in Rust using the Anchor framework, the ar.io protocol is deployed as a cohesive set of Solana programs – leveraging Solana's high-throughput validator network for execution and finality. Program bytecode is published onchain and independently verifiable, keeping the ar.io protocol transparent and auditable by the community.

4.2 Solana Integration

The ar.io protocol runs as four Solana programs – `ario-core`, `ario-gar`, `ario-arns`, and `ario-ant` – that cooperate to implement the network's token logic, gateway registry, naming system, and NFT-backed name ownership. Each program owns a well-defined domain and is invoked by users, by the other ar.io programs via Cross-Program Invocation (CPI), or by an epoch cranker.

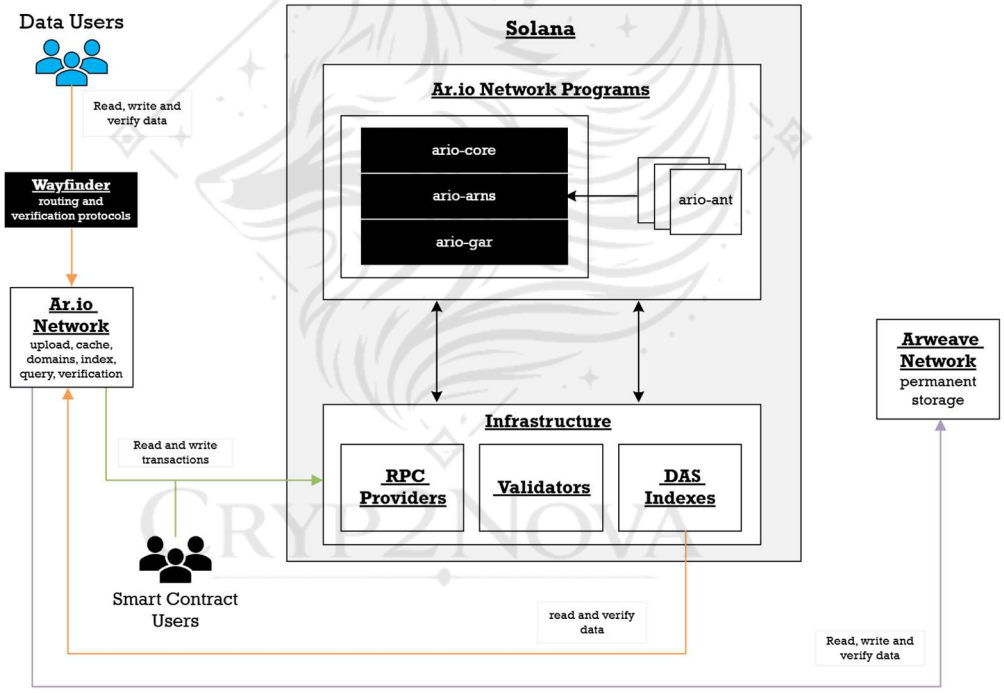


Diagram 4.2: *Solana Integration*

Programs: `ario-core` holds the ARIO token configuration and the protocol balance. ario-gar maintains the Gateway Address Registry – including gateway staking, delegations, and withdrawals – along with epoch state and the observation and reward-distribution logic. `ario-arns` manages the Name Registry, ArNS purchases, leases, and the dynamic pricing engine. `ario-ant` implements Ar.io Name Tokens as Solana NFTs, which carry ownership of individual ArNS names and their configuration. Keeping these domains in separate programs bounds each program's account set, makes upgrades independently reviewable, and allows external protocols to compose against any one surface.

Archived copy · cryp2nova.com

Accounts and PDAs: Solana stores all state in accounts rather than in a single contract state object. The ar.io protocol uses Program-Derived Addresses (PDAs) – deterministic, program-owned accounts – to hold config, epochs, gateways, names, and delegations. Two large accounts are organized as zero-copy registries so the protocol can enumerate them onchain: the `GatewayRegistry` in `ario-gar`, and the `NameRegistry` in `ario-arns`. ARIO balances are held in standard SPL Token accounts, so any Solana-native wallet or application can hold, transfer, or observe ARIO using the same tooling it uses for any other SPL token.

Transactions and Instructions: Users interact with the ar.io protocol by submitting Solana transactions signed with Ed25519 keypairs. A transaction is a list of one or more instructions executed atomically – if any instruction fails, the entire transaction is reverted. This lets a single transaction compose several protocol actions (for example, staking to a gateway and purchasing an ArNS name) into one all-or-nothing atomic operation, and lets protocol operations be bundled with non-ar.io instructions such as SPL token transfers or swaps.

Cross-Program Invocation (CPI): The four ar.io programs call into one another through CPI. For example, an ArNS purchase in `ario-arns` invokes `ario-core` to debit ARIO from the buyer and credit the protocol balance, and invokes `ario-ant` to mint the ANT NFT representing the purchased name. This keeps each program's responsibilities isolated while preserving atomicity – a failure anywhere in the chain reverts every state change the transaction would have made.

State Reads: Clients and gateways read protocol state directly from the Solana cluster via standard RPC methods (`getAccountInfo`, `getProgramAccounts`) and, for higher-throughput or indexed queries, via hosted indexer APIs such as Helius's Digital Asset Standard (DAS) endpoints. Because accounts are cluster-wide data, any validator or RPC provider can serve the current state without a protocol-specific computation layer.

Consensus and Finality: Execution and ordering are provided by the Solana validator network. Transactions are ordered by the current leader, propagated through the cluster, and marked finalized once they reach supermajority vote confirmation. The ar.io protocol inherits Solana's liveness, throughput, and finality guarantees directly and does not operate its own message-sequencing or compute layer.

CRYP2NOVA

4.3 Gas Fees

Because the ar.io protocol runs on Solana, every interaction with it is a Solana transaction and incurs Solana network fees denominated in SOL. These fees are paid to Solana validators and are independent of any protocol-level fees denominated in ARIO. Two categories apply:

1. **Transaction Fees:** Every interaction with the ar.io protocol (e.g., token transfers, staking, ArNS purchases) is submitted as a Solana transaction and pays a small native fee in SOL to the validator that includes it. These fees are set by the Solana protocol and apply uniformly to any transaction on the cluster.
2. **Account Rent:** Solana requires a minimum SOL balance – "rent" – to be deposited on any account that holds state, in proportion to the account's size. For example, spawning a new ANT requires a one-time rent deposit in SOL alongside the standard transaction fee, in addition to any ARIO-denominated protocol fees for the name purchase. The rent deposit is held by the account and is recoverable if the account is later closed. In this way, "rent" acts more like a refundable security deposit than a recurring fee.

Archived copy · cryp2nova.com

4.4 Contract State

The ar.io protocol state is decomposed into a set of onchain accounts stored on Solana, each owned by one of the ar.io programs and addressed by a Program Derived Address (PDA). Together these accounts form the immutable database of record for ar.io. Rather than a single global state object, the model is partitioned across purpose-specific accounts – registries, per-entity accounts, and standard SPL Token accounts – that programs read and mutate atomically via Cross-Program Invocation.

Two onchain registries support enumeration of the network's core entities: the Gateway Registry, and the ArNS Name Registry. Other protocol state – balances, vaults, observations, distributions, pricing parameters – lives in separate accounts as described below.

The state consists of, but is not limited to, the following structures:

Contract State Structure	
Data Structure	**Definition**
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 Ar.io Name System records and their corresponding attributes.
Balances	Unlocked ARIO holdings are held in standard SPL Token accounts on Solana, one per holder, rather than in an in-protocol balance table.
Vaults	A list of nested lists of token vaults, each with an end date at which the vaulted tokens are unlocked for transfer. Vaults have the option to be revokable by the sender/initiator.
Reserved	A list of reserved ArNS names.
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 4.4: *Contract State Data Structures*

4.5 Time-Dependent State

Some ar.io contract features depend on the passage of time to drive changes in state. These include:

Archived copy · cryp2nova.com

- Availability of reserved names
- Purchase prices of names
- Lease expiration statuses
- Protocol reward distributions
- Vaulted token unlocks

On Solana, time is read from the Clock sysvar, which exposes the current slot and a Unix timestamp (in seconds) derived from validator consensus. Protocol logic that depends on the passage of time uses these values as its reference, independent of the Arweave blockchain system.

On Solana, the ar.io protocol has no automatic or global state ticker. Time-dependent state changes – vault unlocks, lease expirations, demand factor updates, and similar transitions – are evaluated lazily when the relevant account is next accessed by an instruction that reads or mutates it. Because Solana programs execute only when invoked by a transaction, any such transition is realized at the moment of access rather than on a recurring schedule.

Epoch transitions, which coordinate observation and reward activity across the network, are driven by a permissionless six-step pipeline – create_epoch, tally_weights, prescribe_epoch, save_observations, distribute_epoch, and close_epoch. Each step is a distinct instruction that any party may submit; the protocol's cranker service performs them on cadence, but the pipeline is open to anyone willing to pay the Solana transaction fees to advance it.

Reads – whether via RPC queries or indexers – always return the current committed state of the relevant account. Any lazy transition that has not yet been realized by an access transaction is reflected in state only after that transaction commits; clients that need a time-adjusted view in the interim compute it offchain from the committed state and the current Clock sysvar values.

4.5.1 Cranking

Because the ar.io protocol has no automatic state ticker on Solana, epoch transitions are advanced by cranker services. A cranker can be any client that submits the instructions in the six-step epoch pipeline once their required preconditions are met.

Each instruction is submitted as a separate, permissionless Solana transaction. There is no privileged "tick" account, scheduled job, or on-chain timer. Instead, the role is filled by any party willing to pay the Solana transaction fees required to advance the pipeline.

Why the Pipeline Uses Multiple Transactions
The epoch pipeline is split across multiple transactions because Solana's per-transaction compute and account-size limits prevent the network's full gateway set from being processed atomically in a single transaction.

Two steps are batched:

- `tally_weights`
- `distribute_epoch`

Each batched transaction processes up to fifteen gateways and resumes across subsequent calls using cursors stored on the epoch account.

Archived copy · cryp2nova.com

The remaining four steps each fit in a single transaction:

- `create_epoch`
- `prescribe_epoch`
- `save_observations`
- `close_epoch`

Driving one full epoch to completion requires roughly thirty transactions.

Permissionless and Idempotent Execution
Cranking is both permissionless and idempotent. Each step is gated by an on-chain flag, such as:

- `weights_tallied`
- `prescriptions_done`
- `rewards_distributed`

These flags prevent a completed step from being executed again. If a completed step is submitted a second time, the transaction is rejected, with the submitter paying only the failed transaction fee.

This allows multiple cranker instances to run at the same time without coordination or conflict. That redundancy is intentional: it removes the cranker from the protocol's trust surface.

If one cranker stops responding, any other party can continue advancing the pipeline. This includes gateway operators, who have a direct economic incentive to ensure rewards are distributed for the work performed during their epoch.

Reference Implementations
There are two maintained reference cranker implementations:

- Standalone cranker service
  - Intended for operators who want dedicated cranker infrastructure.
- Observer-embedded cranker mode
  - Built into the ar.io observer so gateways can handle epoch progression alongside their normal observation duties.

Both implementations submit the same on-chain transactions. The choice between them is purely operational.

Anyone may permissionlessly implement their own cranker using the public instruction interface.

Archived copy · cryp2nova.com

4.6 Settings

The ar.io programs carry settings as constants within their source code. Changing any of these values requires a program upgrade signed by the upgrade authority and redeployed onchain; there is no runtime governance knob that can alter them. When the upgrade authority is permanently revoked, the programs become fully immutable and these settings can no longer change by any mechanism. This ensures that important network parameters cannot be tampered with or changed without serious consideration and intent.

The table below shows representative examples of settings which would require a program upgrade to modify:

Example Protocol Settings		
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 ARIO 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 Duration	Observation and Incentive	The number of seconds that must elapse for an epoch to complete and incentives distributed.

Table 4.6: *Example Protocol Settings*

4.7 Protocol Balance

The Protocol Balance is the primary sink and source of ARIO 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.

The Protocol Balance is held onchain under the control of the ar.io protocol: only the protocol itself can distribute tokens from this balance. 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 protocol is designed to be immutable: its programs are deployed onchain and, once fully immutable, their logic cannot be altered by any party, including their original authors. Immutability is not incidental – it is the guarantee that lets users, name holders, gateway operators, and builders rely on the protocol.

When an upgrade is necessary, a protocol fork is initiated to propose changes. Forks must prioritize three things: ensuring the code is properly upgraded, aligning the community on the

Archived copy · cryp2nova.com

upgrade's purpose, and maintaining the full integrity of the parent contract's state. This collaborative approach reflects the community-driven philosophy of Arweave Draft 17, which emphasizes participation in protocol evolution.

Critically, this process requires users, gateway operators, and other participants to actively opt in to any change through node and software adoption rather than being upgraded silently by parties with key access. The diagram below (inspired by Draft 17) represents how a "pro-social" upgrade fork should be approached.

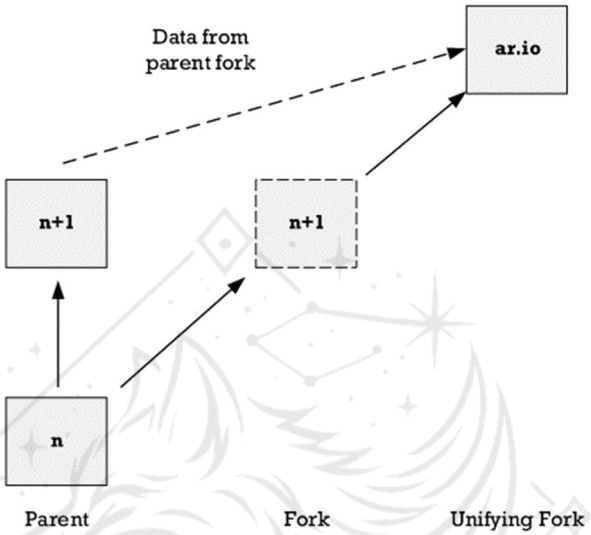


Diagram 4.8: *Ar.io Network Fork Process*

Separately, in cases where the Arweave base layer or the Solana network itself undergoes a validator-level hard fork (a chain split distinct from any ar.io protocol upgrade discussed above), the ar.io contract will continue to exist on both the original and the forked networks. This situation requires the ar.io community – including the Foundation, gateway operators, and end users – to carefully evaluate the nature of the fork and determine its alignment with ar.io's core principles.

In all such cases, the long-term health of the network and its foundational values should guide the community's response.

Archived copy · cryp2nova.com

5 The ARIO Token

5.1 Overview

ARIO is a Solana SPL token that powers ar.io and its suite of permanent cloud applications across its multiple functions. The ARIO Token uses, including:

- **Gateway Participation:** Gateway operators must stake ARIO 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 Ar.io Name System (ArNS) requires ARIO 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, ARIO tokens serve as a versatile currency, enabling network participants to make purchases and exchange value.

Moreover, ARIO tokens play a crucial role in driving ecosystem growth, fueling incentive programs, investments, bounties, and grants designed for active participants.

5.2 Supply and Allocation

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

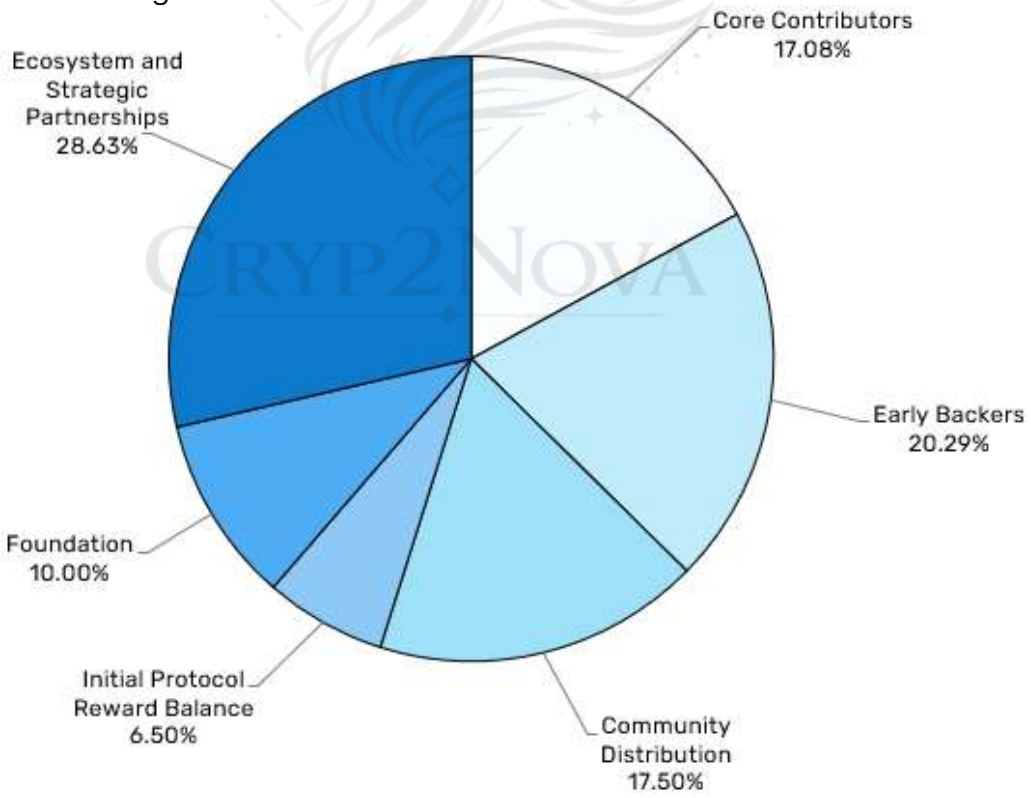


Chart 5.2: *ARIO Token Allocation*

Archived copy · cryp2nova.com

ARIO Token Allocation		
Item	**Allocation (ARIO)**	**% of Total**
Core Contributors	170,787,000	17.08%
Early Backers	202,909,825	20.29%
Community Distribution	175,000,000	17.50%
Initial Protocol Reward Balance	65,000,000	6.50%
Foundation	100,000,000	10.00%
Ecosystem and Strategic Partnerships	286,303,175	28.63%
Total:	**1,000,000,000**	**100.00%**

Table 5.2: *ARIO Token Allocation*

The protocol shall have a fixed total token supply – as such, the smart contract's mint authority shall be revoked. A protocol fork will be required should the 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 Contributors, 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 vesting and unlock schedule of the full token supply. Note that these begin from the network's original genesis (mainnet launch) date.

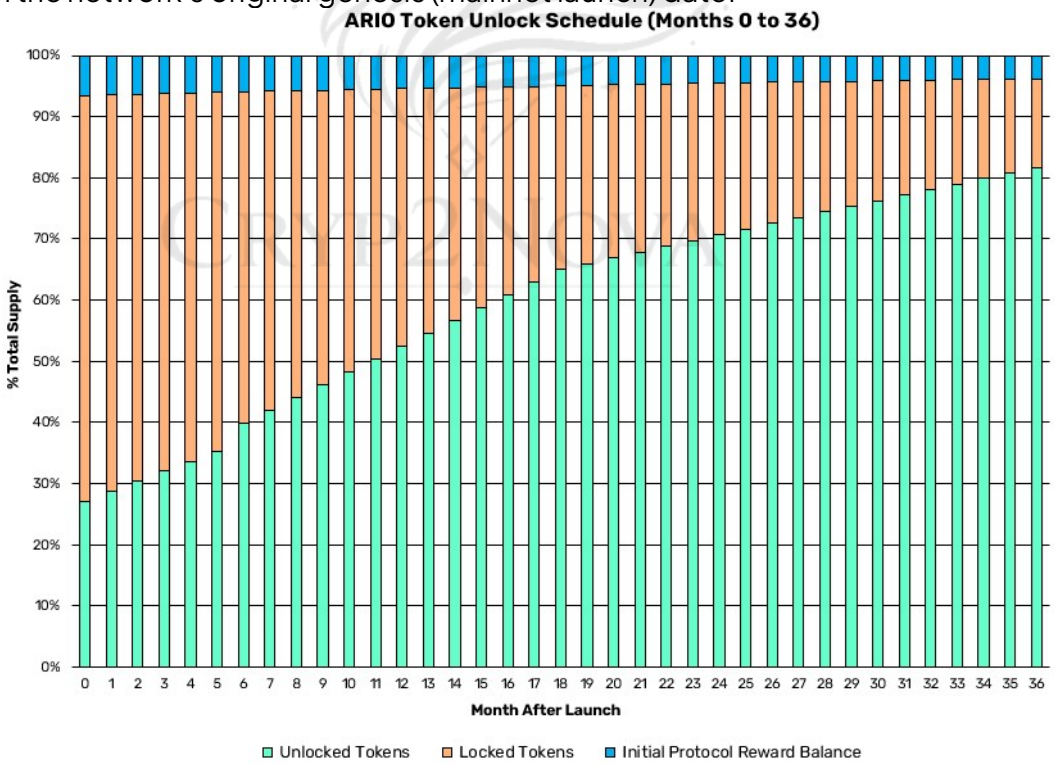


Chart 5.3-1: *ARIO Token Unlock Schedule (Months 0 to 36)*

Archived copy · cryp2nova.com

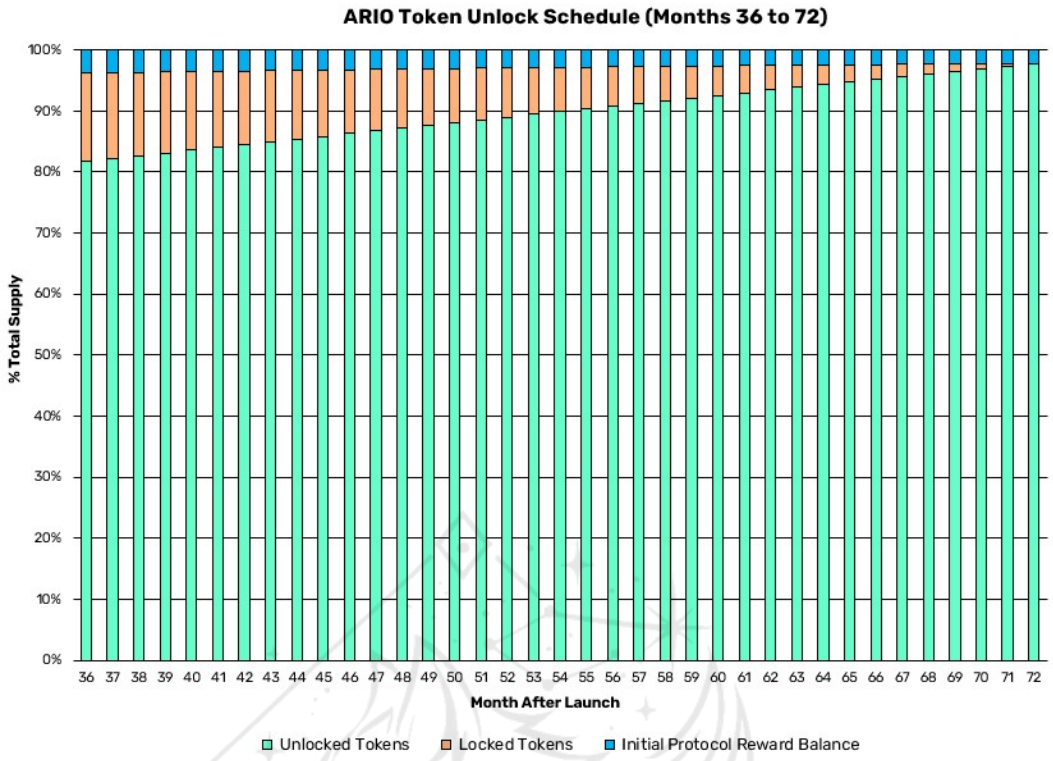


Chart 5.3-2: *ARIO Token Unlock Schedule (Months 36 to 72)*

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.

Note that the protocol balance is held in a PDA-owned SPL token account on Solana and can therefore be topped up by the Foundation or anyone else, providing additional funds to support network incentives and ensure sustained participation and growth.

Archived copy · cryp2nova.com

6 Staking

6.1 Overview

Staking tokens within ar.io serves a dual primary purpose: it signifies a public commitment by gateway operators and qualifies them and their delegates for reward distributions.

In ar.io, "staking" refers to the process of locking a specified amount of ARIO tokens into a protocol-controlled vault. This act signifies an opportunity cost for the staker, 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 / withdraw' action or reaches the end of the vault's lock period.

It is important to note that the ARIO Token is non-inflationary, distinguishing ar.io'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 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. Gateway operators may increase their stake above the minimum, known as excess stake. A gateway's total stake is impacted the following epoch once excess stake is added or removed.

Gateway operators shall have the option to "cancel" their excess stake withdrawal action if still within the withdrawal period and have the stake immediately re-staked to the original gateway.

The protocol imposes specific values for gateway staking, including:

Gateway Staking Values		
Function	**Description**	**Value**
Maximum Gateways	The gateway registry has a maximum capacity of gateways; if the registry is full, new gateways cannot join until existing ones leave.	3,000
Network Join Stake	Minimum number of tokens required for a gateway to join the network (i.e., minimum gateway token stake).	20,000 ARIO

Archived copy · cryp2nova.com

Gateway Staking Values		
Function	**Description**	**Value**
Excess Operator Stake Withdraw Duration	Minimum withdrawal time before a gateway operator can withdraw excess stake tokens from their gateway. Note that this withdrawal period is also applied when a gateway leaves the network.	30 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 days

Table 6.2: *Gateway Staking Values*

6.3 Delegated Staking

To promote participation from a wider audience, the network shall allow anyone with available ARIO 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. Delegated stakers shall have the option to "cancel" their stake withdrawal action if still within the withdrawal period and have the stake immediately re-staked to the original gateway. If a gateway leaves or is removed from the network, delegates must individually claim their stakes via the protocol, subject to withdrawal delays.

Gateway operators retain the discretion to enable or disable delegated staking and can set specific criteria for accepting delegated stakes, including restrictions on wallet addresses (allow list) and minimum stake amount limits. A gateway must already be part of the network to accept delegated stakes.

Existing delegated stakes are "grandfathered in" if settings change. For example, if a gateway increases its minimum stake above the amount of existing delegates, those stakes remain unaffected, but grandfathered delegates can only add or fully withdraw stake; partial reductions below the new minimum are not allowed.

Archived copy · cryp2nova.com

The protocol imposes specific values for delegated staking, including:

Delegated Staking Values		
Constant	**Description**	**Value**
Global Minimum Delegated Stake	Minimum number of tokens that can be delegated to a gateway. Gateway operators shall have the option to set a higher value.	10 ARIO
Delegate Reward Share Ratio	Gateway-specific selected percentage of protocol rewards to be shared proportionally amongst its delegates. It can be thought of as the inverse of "commission".	0 to 95%
Withdraw Delegated Stake Duration	Minimum duration it takes for a delegated staker to withdraw delegated tokens. A gateway's total stake is immediately impacted once delegated stake is removed or reduced.	30 days
Maximum Delegate Count	The maximum number of unique delegates that can stake to a single gateway.	10,000

Table 6.3: *Delegated Staking Values*

Note that any changes made by a gateway to its delegated stake settings during an epoch go into effect the following epoch. For example, if a gateway previously accepting delegates decides to disable this setting, existing delegates will have their stake returned after the minimum withdrawal duration starting the next epoch.

If a gateway has delegated stakers and disables "allow delegated staking," all delegates will have their tokens withdrawn, reducing the gateway's total delegated stake to 0. The gateway cannot re-enable delegated staking until all previous delegates have been withdrawn. This disincentivizes gateways from frequently toggling the setting to remove delegates.

6.4 Stake Redelegation

This feature enables existing stakers to reallocate their staked tokens between gateways, known as redelegation. Both delegated stakers and gateway operators with excess stake (stake above the minimum network-join requirement) can take advantage of this feature. Redelegation is intended to offer users flexibility and the ability to respond to changing network conditions.

Restrictions:

- **Fees:** To balance flexibility with network stability, users are entitled to **one (1) free** redelegation every **seven (7)** days from their last redelegation action. Additional redelegations within this timeframe incur percentage-based protocol fees, increasing in **10%** increments with each redelegation:
  - Redelegation 1: No fee
  - Redelegation 2: 10% fee
  - Redelegation 3: 20% fee
  - ...capped at **60%**

Archived copy · cryp2nova.com

All fees collected from redelegations are sent to the protocol balance.

- All redelegations must respect minimum stake requirements at both the source and destination gateways.
- Vaulted tokens in the withdrawal process may also be redelegated, in line with the above conditions.
- A gateway operator may redelegate stake from one gateway to their own operator stake.
- Gateway minimum network-join stakes are not eligible for redelegation.

6.5 Redeeming Stake for ArNS

Staked tokens generally have restricted liquidity to maintain a healthy degree of stability in the network. However, an exception to these restrictions allows both delegated stakers and gateway operators to redeem eligible staked tokens to fund specific ArNS-related services. By leveraging their stake, participants can further engage with ArNS, strengthening the name system's utilization and impact across the network.

Eligible ArNS Interactions:

- Purchasing a name
- Extending a lease
- Upgrading a lease to permabuy
- Increasing undernames capacity
- Requesting (setting) a primary name

Funding Mechanisms:
Each ArNS purchase is funded from a single source, specified by the user at the time of purchase:

1. **Balance (Default):** The user's liquid ARIO token balance funds the purchase.
2. **Delegated Stake:** A specified active delegated stake or withdrawing delegated stake from a single gateway funds the purchase.
3. **Operator Stake:** A gateway operator's excess operator stake or withdrawing excess stake from a single gateway funds the purchase.
4. **Any:** A combination of balance, stake withdrawal vaults, and active stakes is used to fund the operation. Balances are always drawn first, followed by stakes in a specific order.

In "any" and "stakes" funding modes, the user's liquid balance is drawn first; any remaining shortfall is drawn from staked sources in most-liquid to least-liquid order ("stakes" mode skips the balance step), specifically:

1. **Draw from active withdrawal vaults:** any vault owned by the user is eligible – delegate-stake vaults and operator-stake decrease vaults alike – ordered from earliest end timestamp to latest.
2. **Draw from active stakes in excess of the minimum:** this includes delegated stakes across gateways and, for users running their own gateway, their excess operator stake (above the operator minimum). Sources are ordered from the gateway at which the user has the most excess stake down to the least.
3. **Draw from active minimum stakes:** starting with the stake at the gateway with the lowest performance ratio (worst-performing first) and working up to the highest. If draining a

Archived copy · cryp2nova.com

delegation below the gateway's minimum delegation amount leaves a positive remainder, that remainder is moved into a new withdrawal vault at the gateway with the standard 30-day lock.

To address transaction size limits, a single funding plan may draw from at most five (5) sources in total and at most three (3) distinct delegation gateways, and at most one (1) operator-stake source. If the chosen sources cannot cover the full purchase within these limits, the transaction is rejected with a structured insufficient-funds error reporting the shortfall and the per-source amounts available and no balances or stakes are modified. Users with stake spread across multiple gateways or sources may not submit separate transactions to fund a purchase from each.

Restrictions:

- **No Additional Fees:** ArNS interactions using staked tokens do not impose additional fees.
- **Minimum Stake Requirements:** Reallocations must respect minimum stake requirements at the source gateway.
- **Eligible Stake Sources:** Active delegated stakes and gateway operator excess stake may be used for ArNS interactions. A gateway's minimum network-join stake is not eligible for redemption.

6.6 Expedited Withdrawal Fees

Gateway operators and delegated stakers can shorten the standard withdrawal delay period after initiating a withdrawal (or being placed into an automatic withdrawal by protocol mechanisms); this action is subject to a dynamic fee. At any point during the delay, users can choose to expedite access to their pending withdrawal tokens by paying a fee to the protocol balance, calculated based on how much sooner they want to receive their funds. Once triggered, the tokens are returned immediately to the user's wallet.

Fee Structure:

- The maximum fee starts at **50%** if the user opts for an immediate withdrawal at the beginning of the withdrawal period.
- The fee decreases linearly over the course of the standard withdrawal delay, reaching a minimum of **10%** by the final day of the withdrawal window:

```
Fee (%) = maxFee - [(maxFee - minFee) x elapsedTime] / totalWithdrawTime
```

A gateway's minimum stake is held as operator stake and is not withdrawable while the gateway is active; only stake above the minimum (excess stake) can be placed into a withdrawal during normal operation. When a gateway voluntarily leaves the network, the minimum stake becomes an exit vault and is eligible for expedited withdrawal on the same terms as other withdrawals. If a gateway is instead removed from the network for sustained poor performance (see Section 10), the minimum stake is slashed to the protocol balance and is therefore unrecoverable; any excess stake the operator held above the minimum becomes a standard withdrawal and remains eligible for expedited withdrawal.

Archived copy · cryp2nova.com

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 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.

Serving data involves:

- Facilitating efficient, structured queries for base layer transactions, bundled data items, and wallet data by:
    - Full range request support (RFC 7233) for partial-content delivery and streaming.
    - Signing gateway responses using HTTP Message Signatures (RFC 9421) so that clients can verify the response came from the registered gateway.
    - Optional paid data delivery via the x402 payment protocol.
    - Hedged request routing to network peers to mitigate slow or failed responses from any single source.

Archived copy · cryp2nova.com

Including other benefits and capabilities such as:

- Facilitating friendly-subdomain-name routing to Arweave transactions via a direct integration with the Ar.io 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.

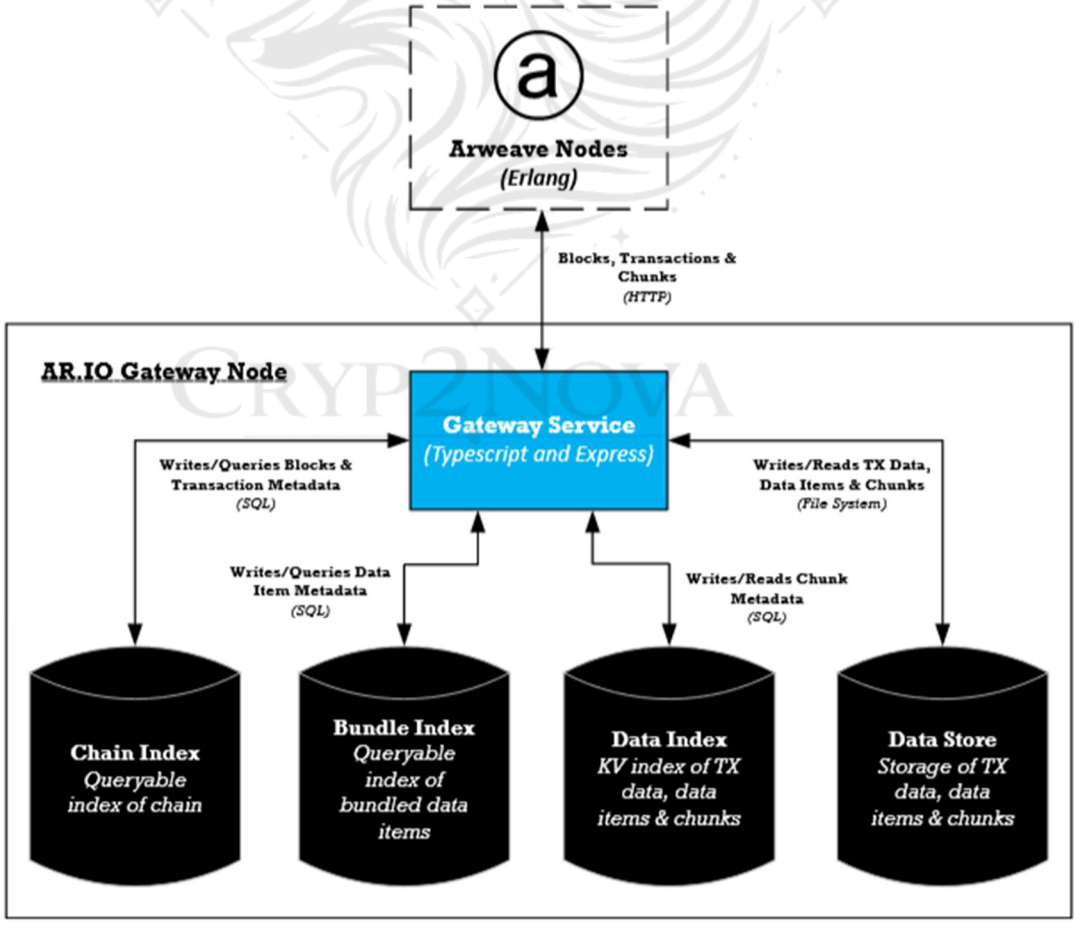


Diagram 7.3: *Gateway System Diagram*

Archived copy · cryp2nova.com

Operators can also optionally run other sidecar services on their gateway, such as bundlers or data verification tools. Gateways that operate a bundler can package and settle data item transactions on the Arweave network, adding a layer of resiliency by ensuring they do not need to rely on external bundlers. This also allows the gateway to contribute to network throughput and data permanence while earning potential upload fees.

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**	**Data Index**	**Data Store**
Small	SQLite	SQLite	Local File System
Large	PostgreSQL	ClickHouse	S3 Compatible

Table 7.3: *Example Gateway Tech Stacks*

7.4 ArNS Indexing and Routing

The Ar.io Name System's (ArNS) state is managed by the ario-arns Solana program and can be retrieved via any Solana RPC endpoint or indexer. 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 ARIO contract as well as the states of the Ar.io 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 and ANT public key.

7.5 Content Moderation

Ar.io adopts 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, ArNS name, or address that is deemed in violation of its content policies or non-compliant with local regulations. Note, however, that overly restrictive content policies may impact a gateway's likelihood of receiving protocol rewards (see Section 10).

Archived copy · cryp2nova.com

8 Gateway Network

8.1 Overview

The ar.io network is a coordinated set of gateways registered in the Gateway Address Registry (GAR). Membership commits a gateway to a shared protocol – observable performance, peer data sharing, signed responses, and reward distribution – that turns independent operators into a single, trust-minimized access layer over Arweave.

Gateway operations are independent of the execution layer. Gateways read protocol state from Solana (registered gateways, ArNS records, staking positions) but serve Arweave data directly. If Solana experiences downtime, gateways continue serving cached data and resolving names from local state. Protocol operations like staking, name registration, and reward distribution resume when Solana does.

Network membership grants a gateway capabilities and obligations beyond what a standalone gateway offers, including:

- Discoverability and routing through the Gateway Address Registry (GAR)
- Prioritized peer data sharing, including hedged request routing across registered peers
- Verifiable responses through trust headers and RFC 9421 HTTP Message Signatures
- Reward eligibility through the Observation and Incentive Protocol (Section 10)
- Delegated staking access, allowing third parties to stake ARIO toward the gateway and improving its operator's chance to improve network security

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 smart contract's "Gateway Address Registry", known as the GAR. Registration involves staking a minimum amount of ARIO 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 Solana wallet address (public key), the primary identifier of the gateway. A single wallet can only be registered to one gateway at a time.
Observer Wallet	The Solana wallet (public key) used to submit observation reports and receive observation rewards. Can be different from the Gateway Wallet. Must be unique across the network. Uniqueness is enforced on-chain via an ObserverLookup PDA. Two gateways cannot register the same observer wallet.
ARIO Token Stake	The amount of ARIO tokens staked which must be above the network minimum.

Archived copy · cryp2nova.com

Gateway Registry Metadata	
Item	**Description**
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, features e.g. indexing filters, or other gateway properties referenced by Transaction ID.
Services	Other services enabled on this gateway e.g. bundling, x402

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 network following an exit wait period. 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) or be removed for deficient performance begin the withdrawal process in the following epoch and become ineligible for gateway and observation incentive rewards, as do any associated delegates (see Section 10).

The GAR advertises the specific attributes of each gateway including its stake, delegates, settings and services. 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, reward performance, or feature set at the top of the list. This would allow users to prefer the higher staked, more rewarded gateways with certain capabilities 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

Ar.io gateways retrieve data from multiple sources using a configurable priority chain. When a gateway receives a request for data it does not have cached locally, it queries sources in the following default order:

1. **Trusted gateways**
   Upstream gateways explicitly configured by the operator, with per-gateway trust flags. Data from trusted sources is cached directly. Data from untrusted sources is cached only after hash verification.

Archived copy · cryp2nova.com

2. **Ar.io network peers**
   Gateways registered in the GAR. Peer selection uses a consistent hash ring that routes requests to gateways most likely to have the data cached, reducing redundant upstream fetches. Requests are hedged: if the primary peer does not respond within a configurable delay (default 500ms), a second peer is queried in parallel.

3. **Direct Arweave node**
   The base layer network serves as the backstop for all block data, transaction headers, and chunks. For bundled data items, the gateway resolves the item's position within its parent bundle and retrieves only the relevant byte range from the root transaction's chunks rather than downloading the entire bundle.

Operators configure this retrieval order based on their deployment requirements.

To prevent routing loops when gateways fetch from each other, the network tracks request provenance via the `X-AR-IO-Via` header and enforces a maximum hop count (default 3). Requests that exceed the hop limit or that would create a cycle skip remote forwarding and fall back to local cache or other non-peer sources.

A negative data cache tracks data IDs that consistently fail to resolve. Repeated misses are handled with exponential backoff, reducing upstream load for data that does not exist on the network.

Root transaction lookups, which map bundled data items to their parent L1 transactions, use a separate priority chain: local database, gateway peers, CDB64 index (a pre-built constant database covering billions of historical data items), and GraphQL fallback. The CDB64 index supports path-based navigation of nested bundles, enabling resolution of deeply bundled data items without downloading intermediate layers.

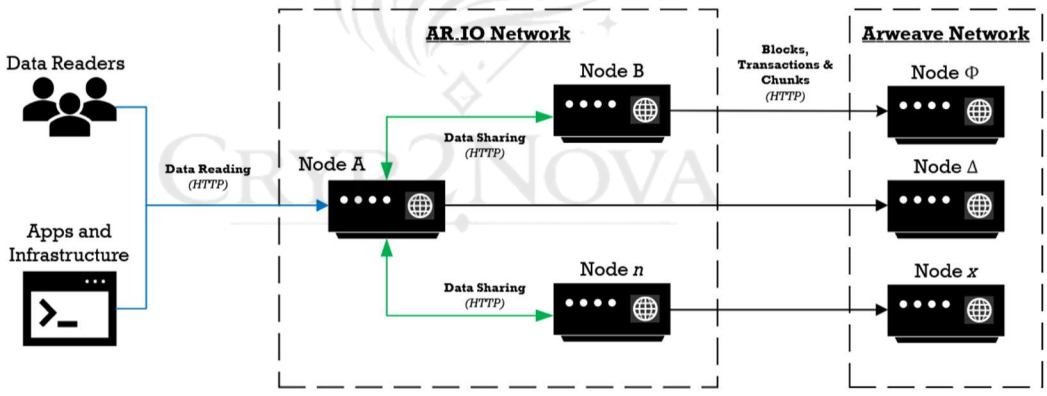


Diagram 8.3: *Data Access and Sharing Diagram*

8.4 Data Verification

Ar.io gateways verify data by linking content hashes of transactions and data items to data roots on the Arweave base layer chain. This verification happens continuously in the background and is exposed to clients through response headers and cryptographic signatures.

Archived copy · cryp2nova.com

8.4.1 Internal Verification

Each gateway runs a background verification worker that continuously re-verifies cached data using Merkle tree cryptography. The worker computes the Merkle data root from the actual data stream and compares it against the indexed root. For base layer bundles that have already been verified, the gateway computes hashes of individual data items, establishing a chain from the data root through the verified bundle to each contained item.

If verification fails, the data is automatically re-imported from Arweave. This ensures that cached data remains consistent with the base layer over time.

8.4.2 Trust and Verification Headers

Gateways expose verification status to clients via HTTP response headers:

- **X-AR-IO-Verified**: true only when data is served from local cache AND has been verified against the base layer. Data served from network sources is not marked as verified, even if the gateway's database records indicate prior verification, because the hash cannot be confirmed in-flight during streaming.
- **X-AR-IO-Stable**: true when the data exists beyond Arweave's maximum fork depth, indicating finality.
- **X-AR-IO-Trusted**: true when the data was retrieved from a source the operator has configured as trusted.
- **Content-Digest (RFC 9530)**: SHA-256 hash of the response body, set on cache hits and HEAD requests. Enables direct body integrity verification by clients.
- **X-AR-IO-Data-Id**: the Arweave transaction ID or data item ID of the content being served.
- **X-AR-IO-Hops**: the number of inter-gateway hops the request traversed.
- **X-Cache**: HIT or MISS, indicating whether the data was served from local cache.

For bundled data items (ANS-104), gateways set two categories of additional headers. Position headers describe the item's location within its parent bundle: root transaction ID, data item offset, size, and parent offset. These allow clients to independently verify the data item's inclusion in a verified bundle.

Metadata headers expose the data item's ANS-104 fields at the HTTP layer: `X-Arweave-Owner-Address` (the data item signer's wallet address) and `X-Arweave-Tag-*` headers (one per tag, with the tag name as the header suffix). This allows clients to read data item tags and owner information directly from HTTP response headers without parsing the ANS-104 binary format. If the tag set exceeds a configurable byte budget, `X-Arweave-Tags-Truncated` is set to indicate partial tag exposure.

8.4.3 HTTP Message Signatures (RFC 9421)

Gateways can opt in to signing response headers using RFC 9421 HTTP Message Signatures. An Ed25519 sub-key signs all trust-relevant headers at response time, producing a Signature and Signature-Input header pair on each response. The signing key is attested by the operator's Arweave RSA wallet, with the attestation uploaded to Arweave. This links each signed response to the operator's staked on-chain identity.

Archived copy · cryp2nova.com

Signatures are bound to the request method and path, preventing replay across different requests. The Ed25519 signing key doubles as a valid Solana address.

Signing is header-only by design. Body integrity for Arweave content is achieved through the signed data ID (which is a content hash) and through the Content-Digest header when available. This preserves streaming performance for large responses.

8.4.4 Client-Side Verification

Clients can verify gateway responses at multiple levels:

- Signature verification. The public key is embedded in the Signature-Input header and is verifiable via the Web Crypto API in modern browsers.
- Identity verification. Following the attestation chain from the signing key to the operator's Arweave wallet confirms the signer is a registered gateway.
- Body integrity. Comparing the Content-Digest header (when present) against a locally computed hash, or walking the Arweave/ANS-104 signature chain from the signed data ID to the content.
- Cross-gateway comparison. Querying the same data from multiple gateways and comparing their signed responses.

The Wayfinder protocol provides client-side routing and verification across the gateway network, with configurable strategies including balanced (random), fastest ping, and static gateway selection.

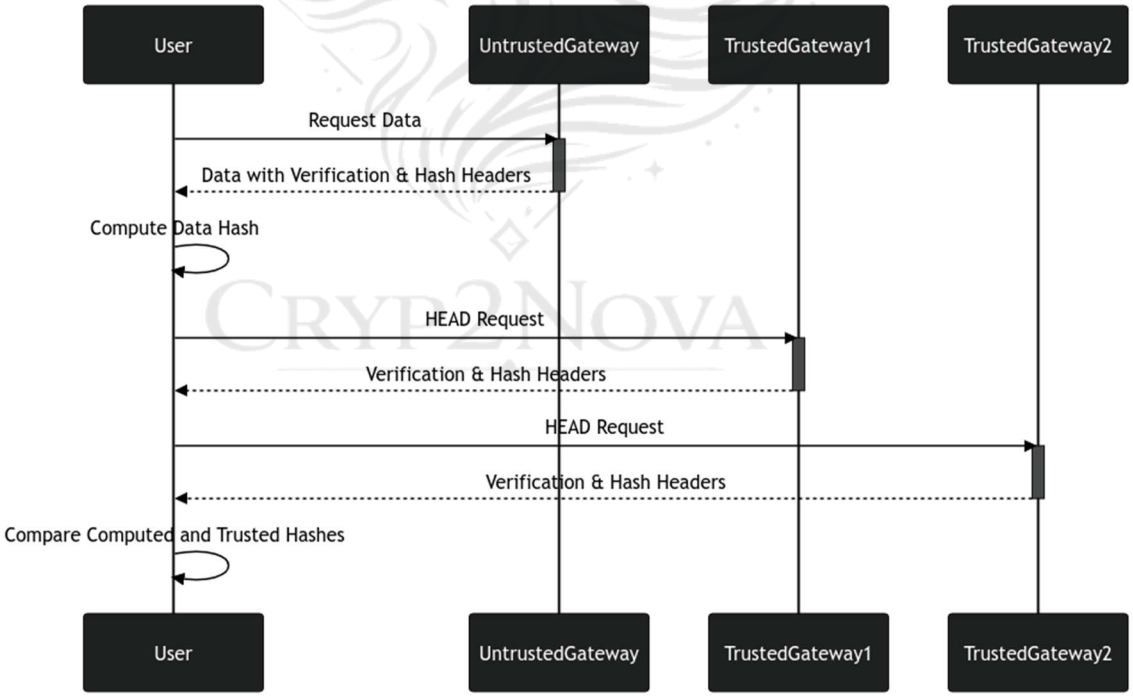


Diagram 8.4.4: *Client Side Data Verification Diagram*

Archived copy · cryp2nova.com

9 Ar.io Name System (ArNS)

9.1 Overview

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

ArNS introduces Smart Domains - modular and composable domain names represented by smart contracts that provide true ownership, flexibility, and an unlimited range of customizations. These domains enable seamless integration with decentralized applications, allowing for dynamic, user-controlled routing on the permaweb.

It is an open, permissionless, domain name registrar that doesn't rely on a single TLD.

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 duration based on their use case. With ArNS, the registry is stored onchain via the ario-arns Solana program, making it globally accessible and verifiable by any Solana RPC node or indexer.

Users can register a name, like *ardrive*, within the ArNS Registry. Ownership is represented by an Ar.io Name Token (ANT), which tracks control of the name. ANTs allow the owner to set a mutable pointer to permaweb data – such as a page, app, or file – via an Arweave transaction ID.

Each ar.io gateway acts as an ArNS Name resolver. They will fetch the latest state of the ArNS Registry from Solana RPC and the state of its associated ANTs, serving 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.ar.io 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.net, https://permagate.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.

Archived copy · cryp2nova.com

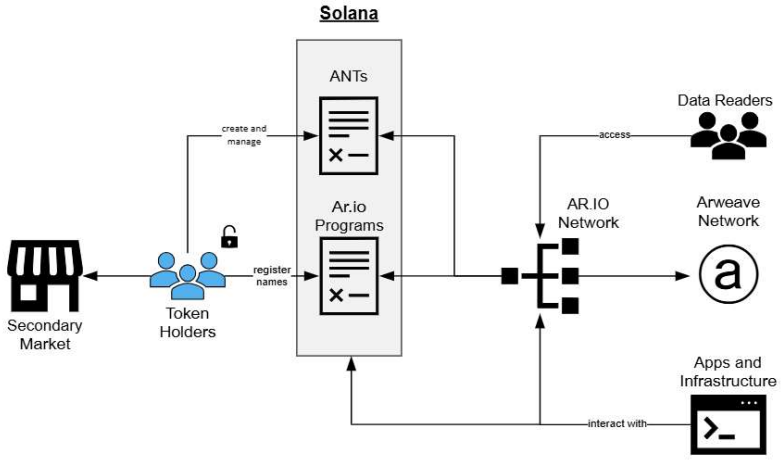


Diagram 9.1: *Ar.io Name System Interactions*

9.2 Name Registration

There are two distinct types of name registrations that can be utilized based upon the needs of the user:

- **Lease:** a name may be leased on a yearly basis. A leased name can have its lease extended or renewed but only up to a maximum active lease of **five (5) years** at any time.
- **Permanent (permabuy):** a name may be purchased for an indefinite duration.

Registering a name requires spending ARIO tokens corresponding to the name's character length and purchase type.

9.2.1 Name Registry

The ArNS Registry is a list of all registered names and their associated ANT Mint Addresses. Key rules embedded within the ario-arns Solana program include:

- **Genesis Prices:** Set within the contract as starting conditions.
- **Dynamic Pricing:** Varies based on name length, purchase type (lease vs buy), lease duration, and current Demand Factor.
- **Name Records:** Include a pointer to the Ar.io Name Token mint address, lease end time (if applicable), and undername allocation.
- **Reassignment:** Name registrations can be reassigned from one ANT to another.
- **Lease Extension:** Anyone with available ARIO Tokens can extend any name's active lease.
- **Lease to Permanent Buy:** Anyone with available ARIO Tokens can convert a name's lease to a permanent buy.
- **Undername Capacity:** Additional undername capacity can be purchased for any actively registered name. A maximum amount of 10,000 undernames can be assigned to a root-level ArNS name.
- **Name Removal:** Name records can only be removed from the registry if a lease expires, or a permanent name is returned to the protocol.

Archived copy · cryp2nova.com

9.2.2 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.
   (a) 43-character names are disallowed to prevent collision with Arweave TXIDs.
5. Shall not be an invalid name predesignated to prevent unintentional use/abuse such as `www`.

9.2.3 Lease Expirations

When a lease term ends, there is a grace period of **two (2) weeks** where the lease can be renewed or converted to a permanent purchase before it fully expires. If this grace period elapses, the name is considered expired and returns to the protocol for public registration. Once expired, a name's associated undername registrations and capacity also expire.

A recently expired name's registration shall be priced subject to the "Returned Name Premium" mechanics detailed below.

9.2.4 Lease to Permabuy Conversions

An actively leased name may be converted to a permanent registration by users. The price for this conversion shall be treated as if it were a new permanent name purchase as detailed in the "Dynamic Pricing" section below.

This functionality allows users to transition from leasing to permanent ownership based on changing needs and available resources. It generates additional protocol revenue through conversion fees, contributing to the ecosystem's financial health and reward system. Additionally, by maintaining fair value for name conversions, it ensures prices reflect current market conditions, promoting a balanced and fair environment.

9.2.5 Permanent Name Return

Users have the option to "return" their permanently registered names back to the protocol. This process allows users to relinquish their ownership, returning the name to the protocol for public re-registration. Only the Owner of a name can initiate a name return.

When a permanent name is returned, the name is subject to a "Returned Name Premium", like expired leases. A key difference is that if the name is repurchased during the premium window, the proceeds are split between the returning owner and the protocol balance.

This feature can be leveraged by users who no longer need their name due to a shift in focus, project changes, rebranding, or any other reason. Name returns help protect the utility of the overall namespace as, rather than being abandoned or put up for sale on a secondary market, an unused permanent name can simply be returned to circulation for use by others.

ar.io
Archived copy · cryp2nova.com

9.2.6 Why Allow Leases

The Arweave protocol assigns a Transaction ID to each piece of content uploaded to the network, while Solana assigns a unique ID to each ANT. 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 or process.

To further the system's value while respecting that it should also remain useful for hundreds 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 and Process 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 for testing.
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 permanent cloud is… permanent. Meaning that the ANT contracts themselves are immutable. Should a user accidentally "brick" an ANT 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 allowing that name to 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 permanent cloud and available for all to view or utilize.

9.3 Primary Names

The Ar.io Name System (ArNS) supports the designation of a "Primary Domain Name" for users, simplifying how domains and names are displayed across ecosystem applications.

Setting a Primary Name

Users can set one of their owned ArNS names as their Primary Name, subject to a fee equivalent to the cost of a single undername on a 51-character name of the same purchase type, adjusted by the current Demand Factor. This allows applications to use a single, human-readable identifier for a wallet, improving user experience across the network.

Archived copy · cryp2nova.com

Rules and Requirements

- Users must own the ArNS name they wish to set as Primary.
- Only one Primary Name can be set per wallet at a time.
- Each Primary Name is unique across the network – the same name cannot be set as the Primary Name for more than one wallet.
- Primary Names can be changed or removed anytime by the owner.
- Expired names lose their Primary Name status, requiring re-registration by new owners.
- The base name's ANT owner can remove any Primary Name set on one of the base name's undernames at any time, regardless of who set it.

9.4 Ar.io 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 Ar.io Name Token (ANT). ANTs are Metaplex Core NFTs on Solana. Each ANT is a unique digital asset whose holder can update the Arweave Transaction IDs that the associated friendly name points to, transfer ownership, and manage the ANT's records, controllers, and metadata.

ANTs are managed by the `ario-ant` Solana program, which enforces the Ar.io Name Token specification required by ar.io gateways to resolve ArNS names to their Arweave Transaction IDs. The program handles record updates, controller management, metadata changes, and ownership reconciliation on transfer.

9.4.1 Ownership

Each ANT has an owner, who can transfer the token and control its modifiable settings. These settings include modifying the address-resolution time-to-live (TTL) for each record contained in the ANT, and other settings like the ANT Name, Ticker, and ANT Controllers. Each ANT supports up to ten (10) controllers; controllers can manage the ANT and set or update records, name, and ticker, but cannot transfer the ANT or assign additional controllers. Note also that each ANT contains a root record ("@") which always resolves the base name itself – the root record can be updated to point to a different Arweave Transaction ID but cannot be removed. Note that ANTs are initially created in accordance with network standards by an end user who can then transfer ownership or assign controllers as they see fit.

Because ANT logic lives in the `ario-ant` Solana program rather than in each individual token, owners do not need to manage per-ANT upgradeability – protocol-level changes apply to all ANTs when the program is upgraded. However, loss of the private key for the wallet holding a permanently purchased ANT can still result in the name being "bricked," as no other party can transfer the NFT or update its records.

9.4.2 Compatibility

Because ANTs are Metaplex Core NFTs, they are compatible with the existing Solana NFT marketplace ecosystem – ANTs can be listed, traded, and transferred on established marketplaces without bespoke tooling. When an ANT is transferred via a marketplace, the `ario-ant` program reconciles ownership on the next interaction with the ANT: all previously assigned controllers are cleared, ensuring that the prior owner's delegates lose access on handover. Tertiary markets supporting the leasing of ArNS names to other users may also be created by third parties unrelated to and outside of the scope of this paper or control of the Foundation.

Archived copy · cryp2nova.com

9.4.3 Interactions

The table below indicates some possible interactions with the ArNS registry, corresponding ANTs, and who can perform them:

Type	Name Interactions			
	ANT Owner	ANT Controller	Under_ name Owner	Any ARIO Token Holder
Transfer ownership	✓			
Add / remove controllers	✓			
Set or change primary root name	✓			
Set or change primary undername			✓	
Reassign name to new ANT	✓			
Return a permanent name	✓			
Set records (pointers)	✓	✓		
Update root records, name, ticker	✓	✓		
Update undername records, name, ticker	✓	✓	✓	
Update descriptions and keywords	✓	✓	✓	
Create and assign undernames	✓	✓		
Extend / renew lease	✓	✓	✓	✓
Increase undernames	✓	✓	✓	✓
Convert lease to permanent	✓	✓	✓	✓

Table 9.4: *ANT Interactions*

9.4.4 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 initial allocation of 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

Archived copy · cryp2nova.com

- Unlike top level 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 sixty-three (63) characters.

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

Undername Ownership
While the parent ANT owner controls which undernames exist and what they point to, individual undernames can be assigned their own owner. This allows the parent name holder to delegate control of a specific undername (e.g., dapp_ardrive) to a different party without transferring the parent name itself. Ownership and configuration of each undername is tracked in the ANT's records.

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 manual intervention. Using a pricing oracle to fix name prices relative to a stable currency is not viable due to the infancy of available solutions and reliance on external dependencies. To address these challenges, ArNS is self-contained and adaptive, with name prices reflecting network activity and market conditions over time.

To achieve this, ArNS incorporates:

1. A dynamic pricing model that adjusts fees using a "Demand Factor" based on ArNS purchase activity.
2. A Returned Name Premium (RNP) system that applies a timed, descending multiplier to registration prices for names that have recently expired or been returned to the protocol.

This approach ensures that name valuations adapt to market conditions within the constraints of an immutable, maintenance-free smart contract framework.

9.6 Dynamic Pricing Model

ArNS employs an adaptive pricing model to balance market demand with pricing fairness for name registration within the network. This model integrates static and dynamic elements, adjusting prices based on name length and purchase options like leasing, permanent acquisition, and undername amounts. A key element is the Demand Factor (DF), which dynamically adjusts prices according to network activity and revenue trends, ensuring prices reflect market conditions while remaining accessible and affordable.

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

9.6.1 Dynamic Pricing Mechanics

Names are initially priced according to the Genesis Registration Fee (GRF), as set in the smart 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

Archived copy · cryp2nova.com

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 name leases or purchases, 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 preceding 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 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 9.6.2).

The pricing system articulates various fees:

- **Adjusted Registration Fee (ARF):** BRF modified by the DF.
- **Annual Fee:** A proportion of the ARF.
- **Name Registration Prices:** Based on the ARF, purchase type, and length.

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.

A fixed amount of 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.

9.6.2 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.

Archived copy · cryp2nova.com

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

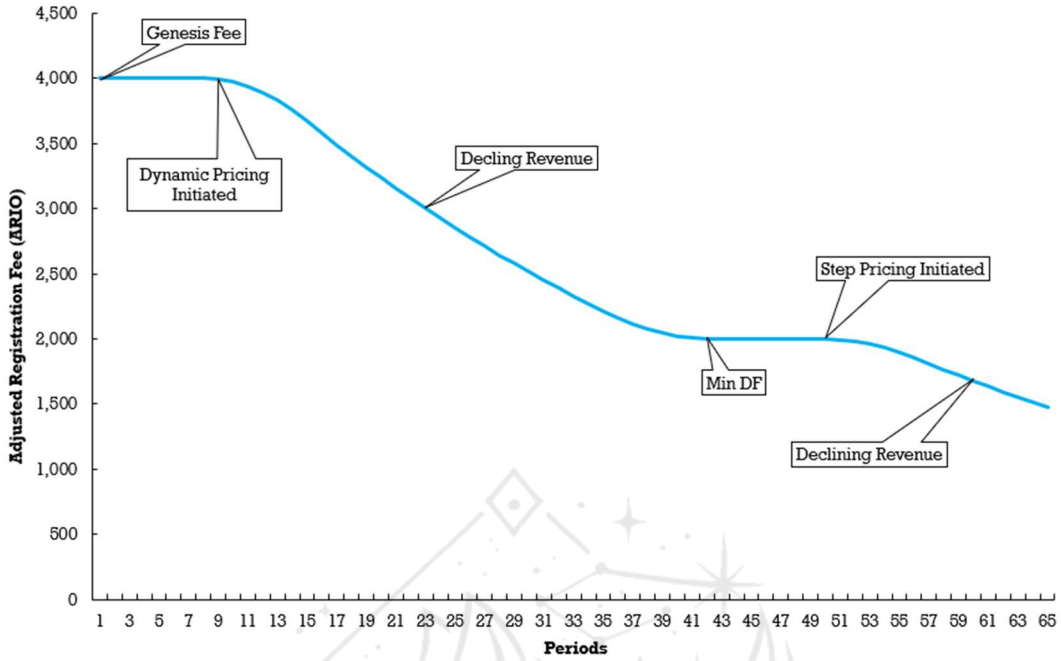


Diagram 9.6.2-1: *Step Pricing in Action – Declining Demand*

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

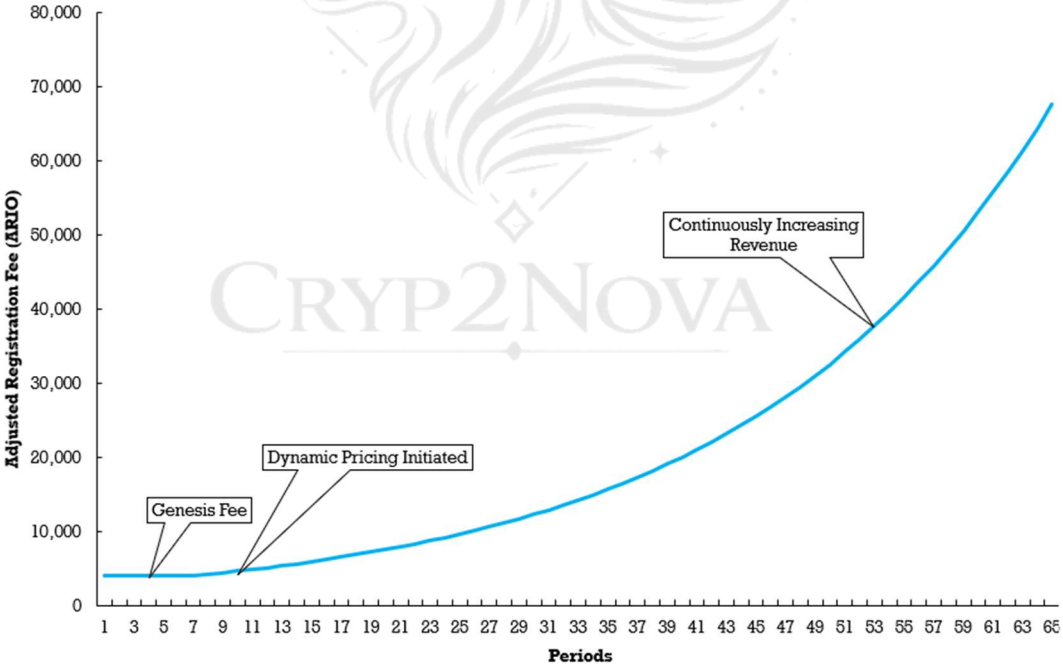


Diagram 9.6.2-2: *Dynamic Pricing in Action – Continuously Increasing Demand*

Archived copy · cryp2nova.com

9.7 Returned Name Premiums (RNP)

ArNS applies a Returned Name Premium (RNP) to names that re-enter the market after expiration or permanent return. This premium starts at a maximum value and decreases linearly over a predefined window, ensuring fair and transparent pricing for re-registered names.

The RNP multiplier is applied to the registration price of both permanently purchased and leased names. Below is an example chart illustrating the Returned Name Premium over time assuming a stable DF:

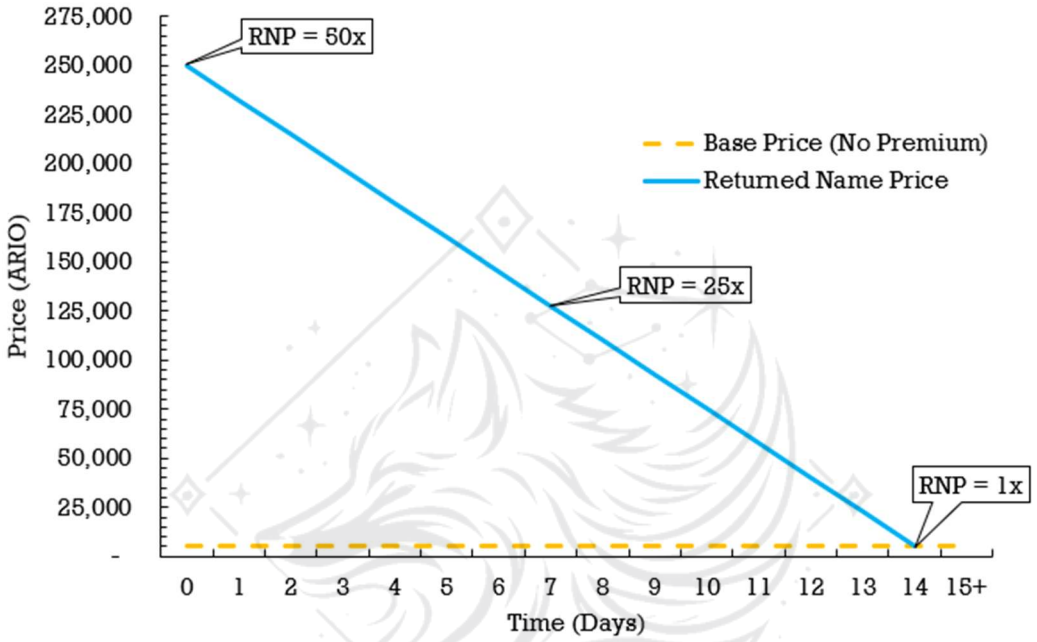


Diagram 9.7: *Example Returned Name Price*

9.8 Gateway Operator ArNS Discount

Gateway operators who demonstrate consistent, healthy participation in the network are eligible for a **20% discount** on certain ArNS interactions.

To qualify:

- The gateway must maintain a "Gateway Performance Ratio Weight" (GPRW) of **0.9** or higher.
- The gateway must have a "Tenure Weight" (TW) of **1.0** or greater, indicating at least a 6-month prior commitment to the network.
- A gateway marked as "Leaving" shall not be eligible for this discount.

Eligible ArNS Discounted Interactions:

- Purchasing a name
- Extending a lease
- Upgrading a lease to permabuy
- Increasing undernames capacity

These performance metrics are detailed in Section 10 (Observation and Incentives).

Archived copy · cryp2nova.com

10 Observation and Incentives (OIP)

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 onchain “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 ARIO 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 **daily 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.

Archived copy · cryp2nova.com

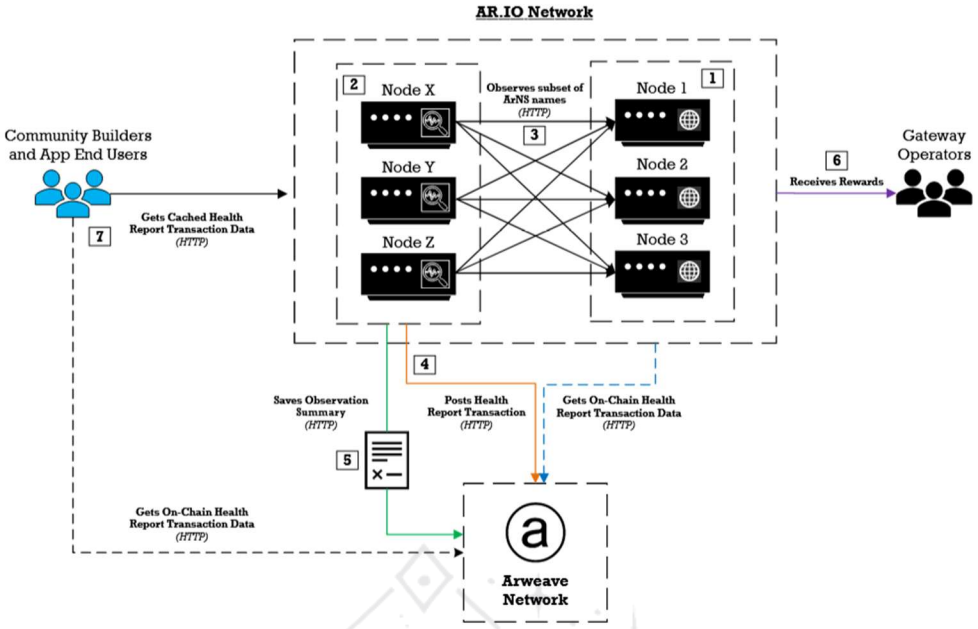


Diagram 10.2: *Observation and Incentive Protocol*

Diagram 10.2 Legend:

1. To participate in the epoch, a gateway must have already staked ARIO tokens and joined the network before it starts.
2. Each epoch, 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 an onchain transaction to the ar.io protocol that records its report Transaction ID together with a pass or fail report for all gateways – for tallying by the incentive protocol.
6. After the observation reporting period and tallying periods have closed, the payout is performed once the distribution step of the epoch pipeline is executed.
    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 Onchain Reports

The to-be-evaluated ArNS names include a set of **two (2)** 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 **eight (8)** "chosen names" selected independently by each observer as part of the Observation and Incentive Protocol. "Prescribed names" are assigned by the protocol to act as a common denominator / baseline across observers, while "chosen names" are an observer-side requirement that defends against gateways optimizing only for the known prescribed set.

Archived copy · cryp2nova.com

The protocol records each observer's aggregate pass/fail results; the selection of chosen names is not tracked onchain.

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 Solana wallet address (public key), starting and concluding timestamps for the epoch.
- **Gateway Operator Assessment**: The expected and actual Solana wallet addresses (public keys) 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 timestamp, 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 onchain audit trail. In addition, observers shall submit an onchain transaction to the ar.io protocol containing a compact gateway results bitmap (one bit per gateway, pass/fail) together with the report's Arweave Transaction ID. These bitmaps 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 **fifty (50)** gateways can be chosen as observers per epoch. If the GAR is below that amount, 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 ARIO 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-years** of participation in the network. It is calculated as:

Archived copy · cryp2nova.com

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

- **Gateway Performance Ratio Weight (GPRW):** 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:

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

- **Observer Performance Ratio Weight (OPRW):** 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:

OPRW = (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 Performance Ratio Weight, and Observer Performance Ratio Weight.

The formula used is:

```
CW = SW x TW x GPRW x OPRW
```

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 with entropy 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 as well as prescribed ArNS names to be evaluated shall be saved in the contract state at the beginning of the epoch to ensure that any activities during that epoch do not affect the selection of observers or awards distribution.

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.

ar.io
Archived copy · cryp2nova.com

- **Functional or Submitted Observers:** are selected observers who diligently perform their duties and submit observation reports and onchain observation submissions.
- **Deficient or Failed Observers:** are selected observers who do not fulfill their duty of submitting observation reports and onchain observation submissions.

At the end of an epoch, the ar.io protocol 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 submissions indicate a PASS score, then that gateway is considered Functional and eligible for gateway rewards.
- Else, if greater than 50% of submitted observer submissions 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 immediately after results have been tallied.

10.6 Reward Distribution

Each epoch, a portion of the protocol balance is earmarked for distribution as rewards. This value shall begin at **0.1%** per epoch for the first year of operation, then linearly decline down to and stabilize at **0.05%** over the following 6 months. 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 **90%** / 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 **10%** / 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.

Gateway rewards will be "auto-staked" in that they will be automatically added to the gateway's operator stake from which they can be withdrawn as needed.

Archived copy · cryp2nova.com

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:

```
DRi = Total Rewards x Reward Share Ratio x (Delegate's Stake/Total Delegated Stake)
```

Delegated stakers will *also* 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). A delegate who fully withdraws their stake before their accrued rewards have been settled forfeits those unsettled rewards, which are redistributed pro rata to the gateway's remaining delegates rather than reverting to the protocol balance.

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. Note that if a gateway (and its delegates) leaves the network or a delegate fully withdraws stake from a gateway, they become ineligible to receive rewards within the corresponding epoch and the earmarked rewards will not be distributed.

10.7 State Snapshot

Note that at the beginning of each epoch, a snapshot of the contract state shall be and used as the criteria for reward distributions in that epoch. This includes items such as:

- Selection of observers,
- Selection of ArNS prescribed names,
- Gateway and delegate stakes and associated settings
- Potential reward amounts

10.8 Handling Deficient Gateways

To maintain network efficiency and reduce contract state bloat, gateways that are marked as deficient, and thus fail to receive rewards, for **thirty (30)** consecutive epochs shall be subject to a "Network Leave" action. Once the thirty (30)-epoch threshold is met, the removal may be submitted to the protocol by any party as a permissionless transaction; the protocol independently verifies that the threshold has been reached before enacting the Leave and be subject to the associated stake withdrawal durations for both gateway stake and any delegated stake. In addition, the gateway shall have its **minimum network-join stake slashed by 100%**. The slashed stake shall be immediately sent to the protocol balance.

ar.io
Archived copy · cryp2nova.com

11 Outro

This white paper has presented a network design aimed at making Arweave more accessible to users and more attractive to gateway operators. Ar.io's incentivization mechanism, backed by the ARIO 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 ARIO 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, chances to enforce its protocols, and gain exposure to larger rewards.

- **Delegated Staking:** Users with ARIO 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 ARIO 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.

These token utilities sit on top of a broader stack. Permanent storage on Arweave only becomes useful when data can be found, retrieved, verified, and resolved to a human-readable address – the role ar.io is designed to fill. Platforms, web applications, AI retrieval pipelines, and digital archives can all build on the same access and naming layer, with the ar.io SDK providing a single interface across Solana and Arweave. As AI systems increasingly depend on retrieved data and content provenance becomes a regulatory concern, the value of a verifiable, decentralized access layer extends well beyond serving content.

Together, these mechanisms reward healthy gateway operators and active participants while discouraging poor performers and malicious actors. Immutable smart contract elements protect economic and namespace integrity, while the upgrade path described in §4.8 leaves room for the protocol to evolve. The result is a framework for a healthy and sustainable decentralized gateway network – and, with it, the access, naming, and verification layer that makes permanent storage usable.

Archived copy · cryp2nova.com

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**
  - **v3.0.0 (this version):** https://whitepaper.ar.io/
  - **v2.1.0:** https://v2-1-0_whitepaper.ar.io
    (txID = 7yhpE0Xx2FGPyw7l1rkfSm1xq4GAeRXdlFvwrw1FOKk)
  - **v2.0.0:** https://v2-0-0_whitepaper.ar.io
    (txID = wJFmcLrUkxZLOYT9youhT8p8apy2Lon_wZcQ9GBn_88)
  - **v1.0.0:** https://v1-0-0_whitepaper.ar.io
    (txID = lNjWn3LpyhKC95Kqe-x8X2qgju0j98MhucdDKK85vc4)
- **Arweave Draft 17:** https://draft-17_whitepaper.ar.io
  (txID = azo-0qw6bb9u5doGdMR-atcIRV_ylJCV4K4Kwv85GO4)
- **The Arwiki:** https://arwiki.wiki/
- **Arweave GitHub repository:** https://github.com/ArweaveTeam
- **Arweave 2.6 Specification:** https://2-6-spec.ar.io
  (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
  - **Ar.io Smart Contract:** https://github.com/ar-io/ar-io-network-process
  - **SDK APIs:** https://github.com/ar-io/ar-io-sdk
  - **Metaplex Core (ANT NFT standard):** https://developers.metaplex.com/core

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:

- **Ar.io Name System (ArNS):** a decentralized and censorship-resistant smart domain system enabled by ar.io gateways which connects friendly names to permaweb applications, pages, and data.
- **Ar.io Name Token (ANT), "Name Token":** an NFT on Solana 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 wraps multiple independent data transactions (data items) into a single base layer transaction, increasing

Archived copy · cryp2nova.com

upload throughput and allowing a third party to pay for an upload while preserving the original signer's identity.

- **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.
- **Cluster:** Solana's term for the network of validators that collectively run the blockchain (e.g., mainnet, testnet, devnet). Used in this paper to refer to the Solana mainnet cluster on which the ar.io protocol is deployed.
- **Cranker:** A permissionless party (bot or any user) that submits transactions to drive lazy protocol operations such as the epoch pipeline, gateway removal, and state advancement. Any party can act as a cranker; the protocol verifies eligibility before enacting the requested action.
- **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 day) 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 ar.io that provides services for reading from, writing to, and indexing the data stored on the Arweave. Sometimes referred to as "permaweb nodes".
- **Gateway Address Registry (GAR):** a decentralized directory maintained in the ar.io smart contract. It serves as the authoritative list of all registered gateways on the Ar.io Network and provides metadata about each gateway to facilitate discovery, health monitoring, and data sharing.
- **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):** an aggregate file on Arweave that maps user-defined sub-paths to other Arweave Transaction IDs, allowing an entire website or directory of files to be launched and referenced by path from a single transaction.
- **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 day) that serves as a cycle for ArNS activities such as dynamic pricing.
- **Permaweb:** The permaweb is the permanent and decentralized web of files and applications built on top of Arweave.
- **Program Derived Address (PDA):** A deterministic Solana account address derived from a program ID and a set of seeds. PDAs allow a Solana program to own and manage accounts without a private key, and are used throughout the ar.io protocol to hold configuration, registries, stake, and vault state.

Archived copy · cryp2nova.com

- **Program ID:** Every Solana program is deployed to a unique onchain address known as its Program ID. Transactions reference a program by its ID when invoking one of its instructions.
- **Protocol Balance:** The primary sink and source of ARIO tokens circulating through the ar.io network. On Solana this balance is held in an SPL Token account controlled by the protocol, from which ArNS revenue is accumulated and incentive rewards are distributed.
- **Protocol Rewards:** ARIO 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.
- **Solana:** A high-performance Layer 1 blockchain. As of v3.0.0, Solana is the base-layer protocol on which the ar.io smart contract and token are deployed.
- **Solana Program:** Stateless executable code deployed onchain on Solana. Programs process signed transactions by executing instructions and read or modify state held in Solana accounts.
- **SPL Token:** A token issued under the Solana Program Library (SPL) token standard, analogous to Ethereum's ERC-20. ARIO is an SPL Token on Solana.
- **Stake Redelegation:** The process by which stakers move their delegated tokens from one gateway to another.
- **Stake Redemption:** A feature allowing stakers to use their staked tokens for ArNS-related activities, such as purchasing names, extending leases, or increasing undername capacity.
- **Staking (of tokens):** Refers to the process of committing ARIO tokens to protocol-controlled accounts, 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.
- **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.
- **Transaction (Solana):** The unit of interaction with Solana. Each transaction is signed by the submitter and contains one or more instructions that are processed atomically – either all instructions succeed, or the transaction reverts with no state change.
- **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 lock tokens over time. Each vault contains a starting time, ending time (if applicable), and a balance of tokens. Vaults have the option to be revocable by the sender/initiator. The minimum vault size is 100 ARIO.
- **Wayfinder protocol:** The Wayfinder protocol provides applications with a pattern for dynamically switching / routing between network gateways. It also allows for abstraction of top-level domain names from Arweave data and verifies the responses from ar.io Gateways. It forms the basis of the ar:// schema, so users can seamlessly access ArNS names, Arweave base layer transactions, and bundled data items without the user providing a top-level domain.

Archived copy · cryp2nova.com

12.3 ArNS Pricing Specification

The following is a detailed description of the ArNS Dynamic Pricing 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 (ARIO)**
1	500,000
2	100,000
3	10,000
4	5,000
5	2,500
6	1,500
7	800
8	500
9	400
10	350
11	300
12	250
13-51	200

Table 12.3: *Genesis Registration Fees*

- **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 ARIO tokens from name registrations, lease extensions, and under_name sales.
- **Period (P):** The time unit for DF adjustments, equivalent to **one (1) day**, denoted in seconds.
- **Revenue Moving Average (RMA):** The average of protocol revenue from the past **seven (7) periods**.
- **n:** The current period indicator.
- **Price:** The cost for 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:**
  - **Lease Registration Price:** Lease Price = ARF + (Annual Fee x # Years)

Archived copy · cryp2nova.com

    - **Lease Extension / Renewal Price:** Lease Renewal Price = Annual Fee x Years (max 5 years)
    - **Grace period:** Two (2) weeks
- **Permanent Purchases:**
    - **Permabuy Price:** Permabuy Price = ARF + (Annual Fee x 20 years)
    - **Lease to Permabuy Price:** Same as above

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%

Primary Name Fee:

- **Set or change primary name:** Equal to the associated fee for a single under_name purchase of a 51-character name of equivalent purchase type to the new primary name, regardless of the new primary name's length.

Demand Factor (DF) Mechanics:

- **Intent:** Adjusts based on protocol revenue comparison to the RMA.
- **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 decreases by 1.5%.
- **Maximum DF Value:** Unbounded
- **Minimum DF Value:** 0.5
- **Starting Demand Factor: 1**, the initial value of the DF at network launch conditions.

Step Pricing Mechanics:

- Synchronizes BRF with ARF after **seven (7) consecutive periods** at the minimum DF value.
- Resets DF to 1 following a step pricing adjustment.

Return Name Premium (RNP) Mechanics:

- **Intent:** Start at the maximum premium and decrease linearly until the name is purchased. If the name is not purchased before the premium window closes, the name reverts to standard pricing and is no longer classified as "recently returned."
- **RNP Window (duration):** Fourteen (14) periods
- **Returned Name Premium:** The premium multiplier follows a linearly declining function:

$$RNP = 50 - 49 / 14 \times t$$

    - *RNP:* The Returned Name Premium multiplier applied to the purchased name price.
    - *t:* Amount of time (or time-intervals) elapsed since the start of the return window.
- **RNP Registration Price:** Price = RNP x (Lease or Permabuy) Registration Price
- **Permanent name return proceeds split:** 50% to returning name owner, 50% to protocol balance.

Gateway Operator Discount:

- **Requirements:** 0.9 GPRW and 1.0 TW.
- **Discount:** 20% on new ArNS name registrations, lease extensions, lease upgrades, and undername purchases.

Archived copy · cryp2nova.com

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 Solana public key.
- The starting timestamp of the epoch.
- The timestamp at which the report was generated.

Overall Gateway Operator assessment:

- Gateway FQDN.
- The Solana public key that the observer expects to be the owner / operator of the gateway.
- The Solana public key 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 timestamp 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://turbo-gateway.com/GG1YCFc7wQxKvQ1qD1lTEp2OAMBs4VzrpfdmeeLyjDI

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

Archived copy · cryp2nova.com

12.5 Major Revisions

This section documents the major revisions to the white paper between versions.

v3.0.0

- **General**
  - The network's smart contract and token infrastructure has migrated from AO to Solana. The paper has been updated throughout to reflect Solana as the network's execution layer.
  - Durations (epoch length, periods, withdrawal delays) are now denominated in seconds rather than the milliseconds.
- **Section 1 – Introduction**
  - Refreshed this section.
- **Section 2 – The Foundation**
  - No major changes.
- **Section 3 – Network Composition**
  - Refreshed this section.
- **Section 4 – The Ar.io Smart Contract**
  - The contract architecture has been redesigned for Solana, replacing the prior AO-based model with a multi-program structure and account-based state.
- **Section 5 – Tokenomics**
  - No major changes.
- **Section 6 – Staking**
  - Added a maximum capacity of 3,000 gateways in the gateway registry.
  - Added a maximum capacity of 10,000 delegated stakers per gateway.
  - Increased minimum network-join stake to 20,000 ARIO.
  - Reduced operator excess stake withdraw time to 30 days.
  - Reduced delegated stake withdraw time to 30 days.
  - Clarified expedited-withdrawal eligibility: exit vaults from a voluntary leave are eligible; minimum stake slashed on network removal is unrecoverable.
  - Operator stake is now eligible for redemption for ArNS purchases.
- **Section 7 – Gateway Architecture**
  - Added a Serving function group covering range requests, signed responses, x402, hedged peer routing, and content moderation hooks.
- **Section 8 – Gateway Network**
  - Reframed slightly around what network membership grants a gateway.
- **Section 9 – Ar.io Name System (ArNS)**
  - Disallowed 43-character ArNS root names to prevent collision with Arweave TXIDs.
  - Undername ownership details added.
  - Specified the Primary Name fee formula (cost of a single undername on a 51-character name of the same purchase type, adjusted by the Demand Factor) in place of "a small fee."
  - A given name cannot be the Primary Name for more than one wallet, and a base name's ANT owner can remove any Primary Name set on one of its undernames.
  - ANTs are now implemented as Solana NFTs.
- **Section 10 – Observation and Incentives**
  - No major functional changes.
- **Section 11**
  - No major changes.
- **Section 12.1 – References and Further Reading**
  - Updated accordingly.
- **Section 12.2 – Glossary**
  - Updated accordingly.
- **Section 12.3 – ArNS Pricing Specification**
  - Reduced base genesis fees for 1, 2, 3, and 4-character name purchases.
- **Section 12.4 – Observer Report Details**
  - No major changes.

Archived copy · cryp2nova.com