ZNS Functionality for Audit

This document contains relevant documentation describing the intended functionality of the system.

Created based on excerpts from the ZNS proposal:
https://docs.google.com/document/d/1Z_IvPa4zqTHm4i1DOuVnn8bjzKPKkjF2IgPkcBh7VNA/edit#heading=h.mdetbp99thxe

This document is based on the **MVP** functionality described in the proposal, plus some additional functionality (staking controller) that has begun to be implemented.

High Level

1 ZNS Registry

ZNS at its core is a naming service just like the internet's DNS or the ENS (Ethereum Name Service). There are a few key differences between these ZNS and ENS however which need to be considered.

1.1 Ownership

ZNS elevates subdomains to exist independently from it's parent domain. Thus ZNS subdomains can just be considered to be domains that exist within another domain, these will be referred to as *child domains*.

In the domain **dorg.tech.zachary, dorg** is the parent of **tech**, which is the parent of **zachary.**

However even if you were to own the **dorg** domain, it does not inherently give you control over the dorg.**tech** or dorg.**tech.zachary** (child) domains.

Whoever owns the dorg.**tech** and dorg.**tech.zachary** child domains have absolute control over these domains.

1.2 Format

Conventional naming structures follow a

Subdomain.domain.TLD structure, such as '**mail.google.com**'

ZNS however eliminates the concept of **TLD** (top level domains) and also arranges “sub domains” to be left to right

domain.domain.domain such as ‘**dorg.tech.zachary**’

Where the parent to child relationship is from the left to right; **dorg** is the parent of **tech** which is the parent of **zachary**

1.3 Parent Requirements

ZNS requires that a parent domain exists before the registration of a child domain can happen.

Thus if the domain **dorg** existed, it would be impossible to make the **dorg.tech.zachary** sub domain unless the **dorg.tech** domain already existed.

1.4 Registering Domains

For any given domain, a child domain can be registered.

The child domain will have its name appended onto the end of the parent domain.

A child domain must not already have an owner to be registered.

Thus given a domain “**cars”,** to register the domain **cars.fast** then the **cars.fast** domain cannot already have an owner.

MVP Deliverable

This section details what the MVP (minimum viable product) deliverable consists of, how it will be executed, and estimates on timing

2 Requirements and Scope

These requirements must be met for an MVP

2.1 Ability to Traverse Domain Structure

For any given domain, there must be a way to see the parent domain and all children domains

2.2 Transfer Domains

It must be possible to transfer a domain from one user to another

2.3 Creation of Child Domains

It must be possible for child domains to be created.

This will be accomplished through either the Basic Controller or the Staking Controller.

2.4 Domain NFT

A domain must be represented by an NFT

2.5 Domain NFT Content

A domain NFT must have some sort of owner-configured content associated with it through the EIP-721 metadata extension interface.

- Properties (JSON) *(through a uri, these are off chain)*
    - Actual NFT Name
    - Media
    - Description
- Owner
- Creator
- Royalty Amount
    - % to send to creator

The creator of a domain NFT can decide whether or not the properties uri is locked to changes.

Sub graph can infer

- Name

2.6 Domain Details

A domain must have important information attached to it, such as the owner

- Owner

Sub graph can infer

- Child Domains
- Ownership history

2.7 Subgraph

Data served via a subgraph

2.8 Bid to create sub domains

This is an extension of 2.3

For a sub domain that is to be created:

It must be possible for a buyer to place a bid (in tokens) and the parent domain owner can accept the bid. Then the bid will be acted upon and the subdomain created.

This is accomplished by using signed messages (bids).

3 Requirements out of Scope

These requirements and features are out of scope for the MVP. Even though they may be imperative for a fully functional product, they are not to be included in the MVP scope.

Completion of these requirements will occur at a later date in a later version.

3.1 Release of Domains

An owner of a domain may release the domain from their control (and regain the tokens they bid for the domain if it's from the staking controller)

3.2 Domain Resolution

There is no resolution of domains into data.

3.3 Domain Content Tokens

There are no editions of domain content tokens.

4 Components in Scope

To satisfy the MVP requirements the following components must be built.

4.1 ZNS Registrar

The ZNS Domain Registrar is responsible for:

- Registering new domains inside of the ZNS
    - Only authorized addresses may do this
- Domain NFT's
    - Represent domain ownership
    - Allows for transferring of domains from one user to another
- Domain NFT Content
    - The metadata uri associated with a Domain NFT
    - The ability for a domain owner to change the metadata uri
    - Allow the original creator to lock the metadata uri

4.2 ZNS Basic Controller

In place of the ZNS Staking Controller a simpler controller can be created which would allow users to register domains while the ZNS Staking Controller is completed.

Both controllers can exist at the same time if needed, however the deprecation of the Basic Controller can be performed with a migration story.

The Basic Controller is responsible for:

- Allowing the owner of a domain to create new child domains
    - New domains are granted to the owner or an address of their choice

4.3 Staking Controller (v1)

The Staking Controller allows any user to place a bit for subdomain. The owner of the subdomain parent may approve the bid. After the bid is approved, a user may fulfill the bid, staking tokens in return for ownership of the subdomain.