Name Service

A SPL program for issuing and managing ownership of: domain names, Solana Pubkeys, URLs, Twitter handles, ipfs cid's etc..

This program could be used for dns, pubkey etc lookups via a browser extension for example, the goal is to create an easy way to identify Solana public keys with various links.

Broader use cases are also imaginable.

Key points:

- A Name is a string that maps to a record (program derived account) which can hold data.
- Each name is of a certain class and has a certain owner, both are identified
- by their pubkeys. The class of a name needs to sign the issuance of it.
- A name can have a parent name that is identified by the address of its record.
- The owner of the parent name (when it exists) needs to sign the issuance of
- the child name.
- The data of a name registry is controlled by the class keypair or, when it is
- set toPubkey::default()
- , by the name owner keypair.
- Only the owner can delete a name registry.

Remarks and use cases:

- Domain name declarations: One could arbitrarily set-up a class that we can call
- Top-Level-Domain names. Names in this class can only be issued with the
- permission of the class keypair, ie the administrator, who can enforce that
- TLD names are of the type".something"
- . From then on one could create and
- own the TLD".sol"
- and create a class of ".sol" sub-domains, administrating
- the issuance of the"something.sol"
- sub-domains that way (by setting the
- parent name to the address of the".sol"
- registry).

An off-chain browser extension could then, similarly to DNS, parse the user SPL name service URL input and descend the chain of names, verifying that the names exist with the correct parenthood, and finally use the data of the last child name (or also a combination of the parents data) in order to resolve this call towards a real DNS URL or any kind of data.

Although the ownership and class system makes the administration a given class centralized, the creation of new classes is permissionless and as a class owner any kind of decentralized governance signing program could be used.

- Twitter handles can be added as names of one specific name class. The class
- authority will therefore hold the right to add a Twitter handle name. This
- enables the verification of Twitter accounts for example by asking the user to
- tweet their pubkey or a signed message. A bot that holds the private issuing
- authority key can then sign the Create instruction (with a metadata_authority
- that is the tweeted pubkey) and send it back to the user who will then submit
- it to the program.
- In this case the class will still be able to control the data of the name registry, and not the user for example.

Therefore, another way of using this program would be to create a name ("verified-twitter-handles" for example) with thePubkey::default() class and with the owner being the authority. That way verified Twitter names could be issued as child names of this parent by the owner, leaving the user as being able to modify the data of their Twitter name registry. Edit this page Previous Memo Program Next Shared memory Program