Ethereum Name Service

CSY54

Outlines

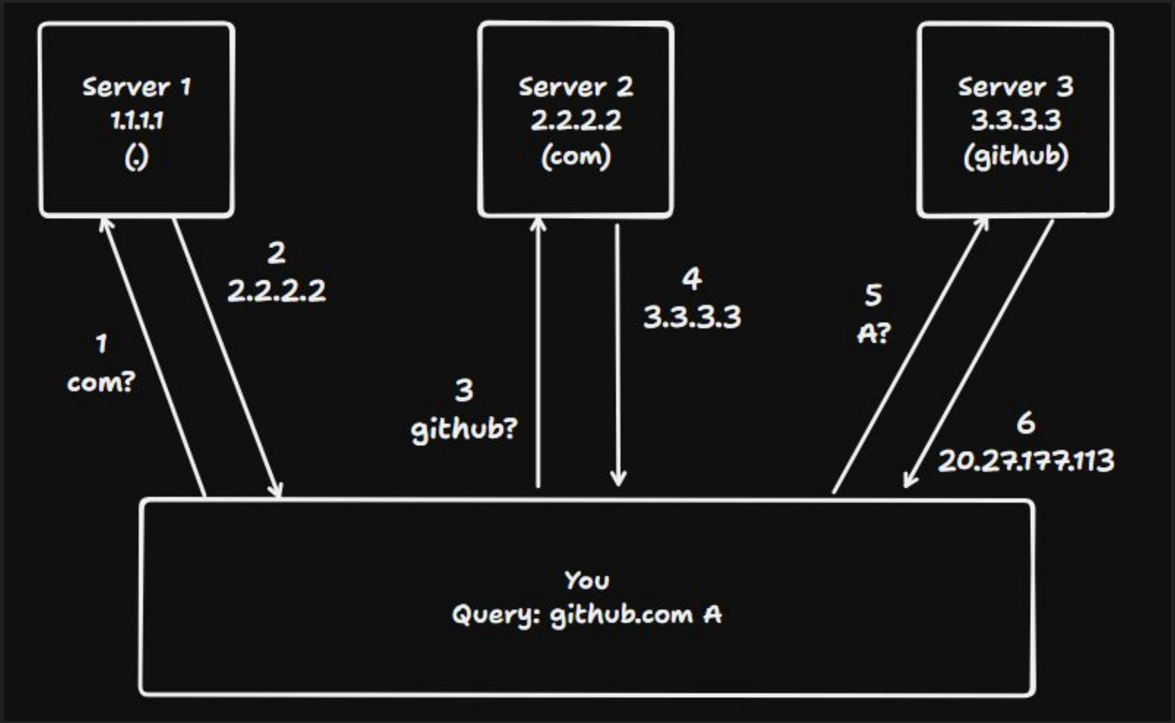
- What is ENS?
- My deployed version
- Demo
- Misc
- Q&A

What is ENS?

What is ENS?

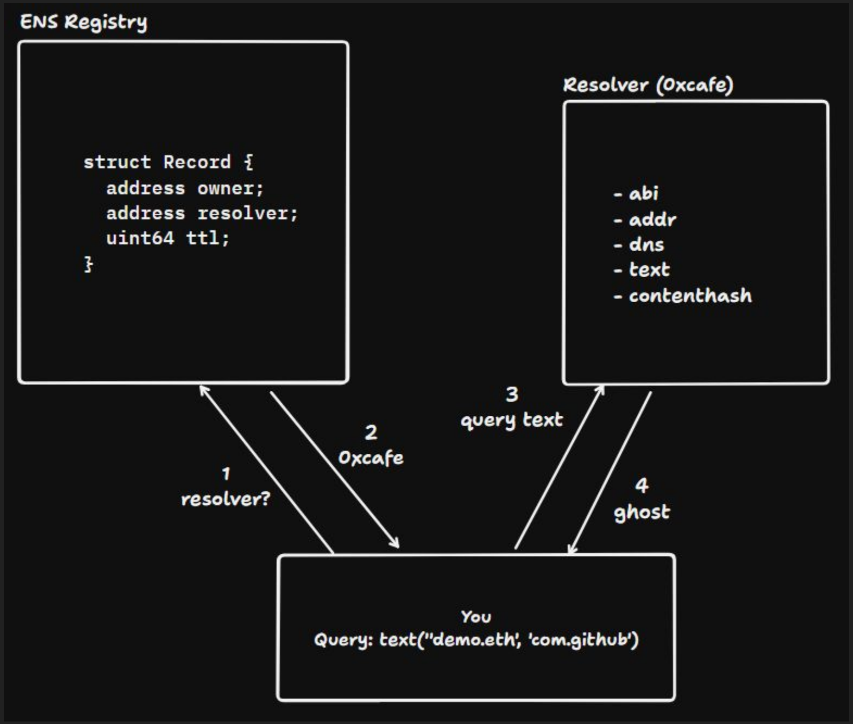
- DNS on chain

DNS vs. ENS



Server 1
1.1.1.1
(.)
Server 2
2.2.2.2
(com)
Server 3
3.3.3.3
(github)
1
com?
2
2.2.2.2
3
github?
4
3.3.3.3
5
A?
6
20.27.177.113
You
Query: github.com A

DNS vs. ENS



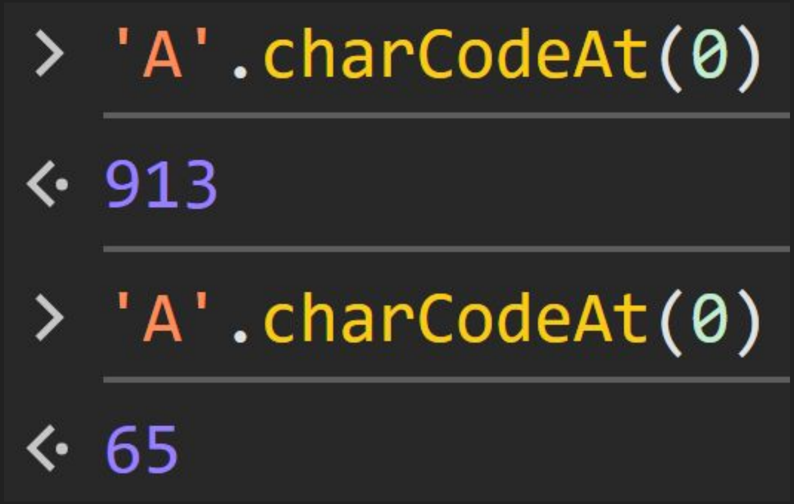
```
struct Record {
  address owner;
  address resolver;
  uint64 ttl;
}
```

Available Resolve Types

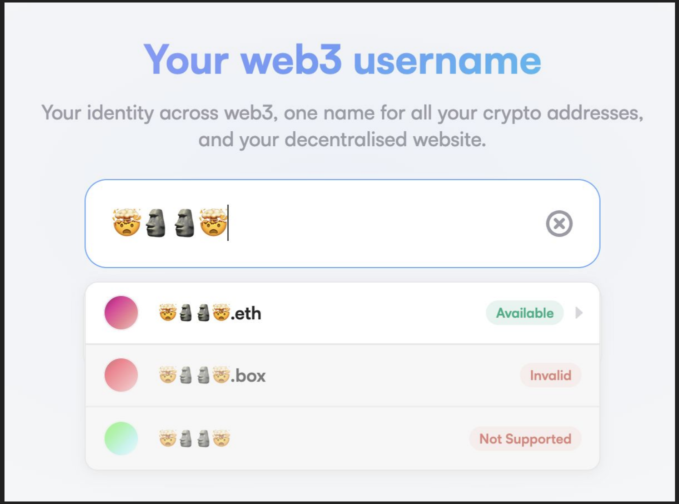
Type	ENSIP	Type	ENSIP
Addr	ENSIP-1	Contenthash	ENSIP-7
Name	ENSIP-3	Interface Implementer	ENSIP-3
ABI	ENSIP-4	ABI	ENSIP-4
Text	ENSIP-5	DNS-in-ENS	ENSIP-6
Avatar	ENSIP-12		

Name Normalization

- Define a normalization process to prevent issue with confusables
- Service indicators on unusual characters ⚠️
- Emojis are available 🤯🤯🤯



```
> 'Α'.charCodeAt(0)
<· 913
> 'A'.charCodeAt(0)
<· 65
```



Your web3 username

Your identity across web3, one name for all your crypto addresses, and your decentralised website.

🤯🗿🗿🤯

🤯🗿🗿🤯.eth	Available
🤯🗿🗿🤯.box	Invalid
🤯🗿🗿🤯	Not Supported

My deployed version

PublicResolver.sol - My Version

- Only supports
  - addr
  - name
  - text

ReverseRegistrar.sol - ENS

- Owns `addr.reverse` record
- Registers `<address>.addr.reverse` and resolve it to `name(<address>)`

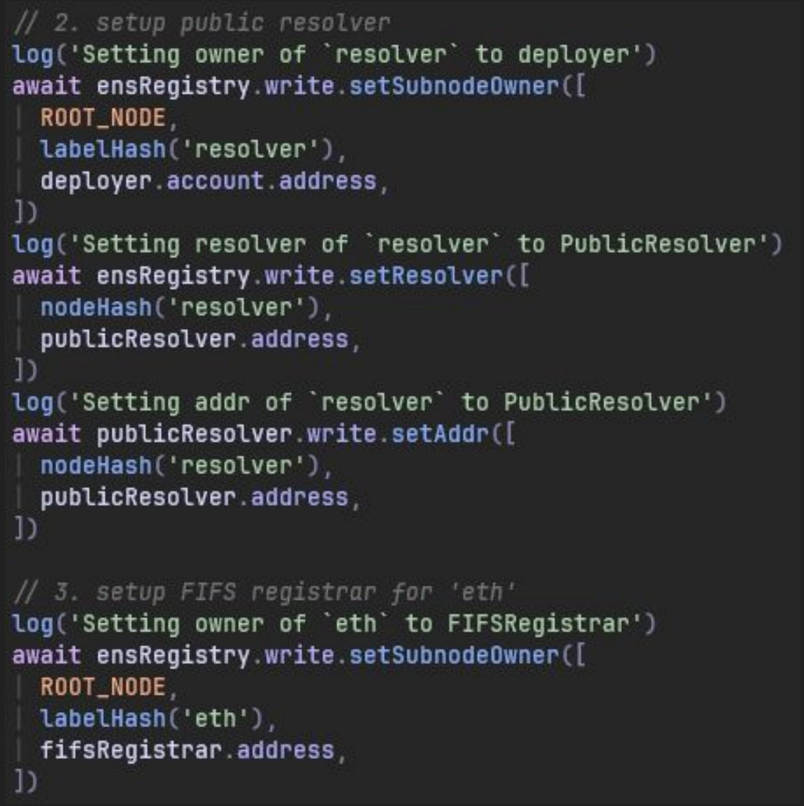
FIFSRegistrar.sol - ENS

- First-In-First-Serve Registrar

Root.sol - ENS

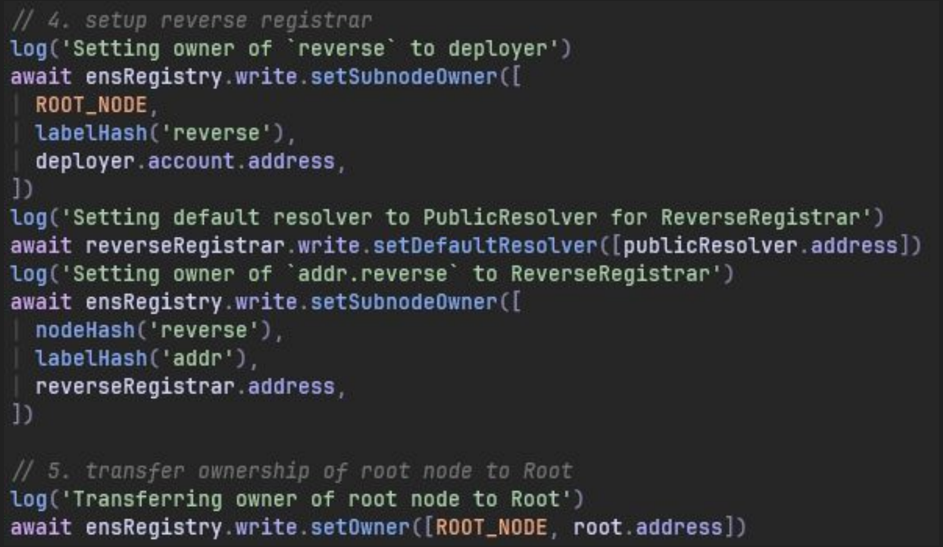
- Hold the root node (0x00…00)
- Has ability to “lock” TLD’s owner

Setting Up Contracts



```
// 2. setup public resolver
log('Setting owner of `resolver` to deployer')
await ensRegistry.write.setSubnodeOwner([
  ROOT_NODE,
  labelHash('resolver'),
  deployer.account.address,
])
log('Setting resolver of `resolver` to PublicResolver')
await ensRegistry.write.setResolver([
  nodeHash('resolver'),
  publicResolver.address,
])
log('Setting addr of `resolver` to PublicResolver')
await publicResolver.write.setAddr([
  nodeHash('resolver'),
  publicResolver.address,
])

// 3. setup FIFS registrar for 'eth'
log('Setting owner of `eth` to FIFSRegistrar')
await ensRegistry.write.setSubnodeOwner([
  ROOT_NODE,
  labelHash('eth'),
  fifsRegistrar.address,
])
```



```
// 4. setup reverse registrar
log('Setting owner of `reverse` to deployer')
await ensRegistry.write.setSubnodeOwner([
  ROOT_NODE,
  labelHash('reverse'),
  deployer.account.address,
])
log('Setting default resolver to PublicResolver for ReverseRegistrar')
await reverseRegistrar.write.setDefaultResolver([publicResolver.address])
log('Setting owner of `addr.reverse` to ReverseRegistrar')
await ensRegistry.write.setSubnodeOwner([
  nodeHash('reverse'),
  labelHash('addr'),
  reverseRegistrar.address,
])

// 5. transfer ownership of root node to Root
log('Transferring owner of root node to Root')
await ensRegistry.write.setOwner([ROOT_NODE, root.address])
```

TypeScript btw

Deployed Addresses - Sepolia

- ENSRegistry: `0x9e86C080275f531A1c2bca31303797d634702E38`
- FIFSRegistrar: `0xe43D572B326Fe31683c11f8F7a9ca99970367e7a`
- ReverseRegistrar: `0x317c788644EC63f2aCd4aD0e68CF106Ea1897d16`
- PublicResolver: `0xef7CE222921f0024F8A05130411826DA81a72783`
- Root: `0xaF9Ed5d73029896E01195eC37b05c00470316B50`

Deployed Addresses - Holesky

- ENSRegistry: `0x291513d6b987b055F1756FF8c9b9C4a7b5B5fA40`
- FIFSRegistrar: `0xde1D8A0Db97F7184f61c5A1B5d54228334D1f8AC`
- ReverseRegistrar: `0x63220518a48BcC3289F2e3BCb662FE6dA2Dc0f97`
- PublicResolver: `0xff56667cA50b88acD526eB22db71bbDeEc70A2E5`
- Root: `0x5442e7AF30202FFc28b41be0F2C27D7283b31a02`

Demo

Misc

Authorization of a Node

- Make other accounts available to manage the node you owned

Offchain Registrar

- Registrars could be offchain
- gskril/ens-offchain-registrar

DNSSEC

- Should be able to host a local DNS server with DNSSEC
- Since the lookup part is done offchain

Deploy to Zircuit?

- My toolchain is complaining
- Viem and Wagmi doesn’t support Zircuit natively
- L2 like zkSync seems fine, though

Q & A

Thanks