DEMYSTIFYING ENS

...with sbt-ethereum
ENS Architecture Summary

- **TLD**: "eth"
- **Registrar**: 0xfac7bea255a6990f749363002136af6556b31e04
- **Controller**: 0xb22c1c159d12461ea124b0deb45b93020e6ad16
- **Resolver**: 0x314159265dd8dbb310642f98f50c066173c1259b, 0x97683a370239817cf33ec2c2ad3b3a1884571f69
- **Interface Implementer (Controller)**: 0x018fac06, 0x6ccb2df4
- **Interface Implementer (Resolver)**: 0xb22c1c159d12461ea124b0deb45b93020e6ad16

**Permanent**:
- **Owner (Registrar)**: 0x314159265dd8dbb310642f98f50c066173c1259b
- **Resolver (Registrar)**: 0x97683a370239817cf33ec2c2ad3b3a1884571f69

**Temporary**:
- **Controller**:
  - **Interface ID** — **Controller**: 0x018fac06
  - **Interface ID** — **ERC721 (NFT)**: 0x6ccb2df4

---

- **Namehash ("eth")**: 0x93cdeb708b7545dc668eb9280176169d1c33cfd8ed6f04690a0bcc88a93fc4ae
- **Controller**: 0x018fac06
- **ERC721 (NFT)**: 0x6ccb2df4
ENS operates on hashes, not names

In order to distinguish between hashes of new names (which must be registered) and subnodes (which owners can define and alter at will, hashing is hierarchical.

Given a name like 'happy.birthday.eth', 'happy' is the label and 'birthday.eth' is the parent

hash('') is defined as 0x0000000000000000000000000000000000000000000000000000000000000000

let's call this ZEROHASH

namehash( childpath ) = keccak256( namehash( parent ) ~concat~ keccak256( normalize( label ) ) )

labels are normalized by lowercasing then using IDN (international domain name, "punycode") conventions to convert to ascii-only bytes

let's define labelhash( label ) = keccak256( normalize( label ) )

So, the top-level-domain 'eth' gets EthNameHash = namehash( ZEROHASH ~concat~ labelhash( "eth" ) )

'happy.eth' gets HappyEthNameHash = namehash( EthNameHash ~concat~ labelhash( "happy" ) ), etc.
interface ENS {

    // Logged when the owner of a node assigns a new owner to a subnode.
    event NewOwner(bytes32 indexed node, bytes32 indexed label, address owner);

    // Logged when the owner of a node transfers ownership to a new account.
    event Transfer(bytes32 indexed node, address owner);

    // Logged when the resolver for a node changes.
    event NewResolver(bytes32 indexed node, address resolver);

    // Logged when the TTL of a node changes
    event NewTTL(bytes32 indexed node, uint64 ttl);

    function setSubnodeOwner(bytes32 node, bytes32 label, address owner) external;
    function setResolver(bytes32 node, address resolver) external;
    function setOwner(bytes32 node, address owner) external;
    function setTTL(bytes32 node, uint64 ttl) external;
    function owner(bytes32 node) external view returns (address);
    function resolver(bytes32 node) external view returns (address);
    function ttl(bytes32 node) external view returns (uint64);
}
contract BaseRegistrar is IERC721, Ownable {
    uint constant public GRACE_PERIOD = 90 days;

    event ControllerAdded(address indexed controller);
    event ControllerRemoved(address indexed controller);
    event NameMigrated(uint256 indexed id, address indexed owner, uint expires);
    event NameRegistered(uint256 indexed id, address indexed owner, uint expires);
    event NameRenewed(uint256 indexed id, uint expires);

    // The ENS registry
    ENS public ens;

    // The namehash of the TLD this registrar owns (eg .eth)
    bytes32 public baseNode;

    // A map of addresses that are authorised to register and renew names.
    mapping(address=>bool) public controllers;

    function addController(address controller) external;                                // onlyOwner (in current implementation)
    function removeController(address controller) external;                             // onlyOwner
    function setResolver(address resolver) external;                                    // onlyOwner
    function nameExpires(uint256 id) external view returns(uint);                      // onlyOwner
    function available(uint256 id) public view returns(bool);                          // onlyOwner
    function register(uint256 id, address owner, uint duration) external returns(uint); // onlyController (in current implementation)
    function renew(uint256 id, uint duration) external returns(uint);                  // onlyController
}
contract IERC721 is IERC165 {
    event Transfer(address indexed from, address indexed to, uint256 indexed tokenId);
    event Approval(address indexed owner, address indexed approved, uint256 indexed tokenId);
    event ApprovalForAll(address indexed owner, address indexed operator, bool approved);

    function balanceOf(address owner) public view returns (uint256 balance);

    function ownerOf(uint256 tokenId) public view returns (address owner);

    function safeTransferFrom(address from, address to, uint256 tokenId) public;
    function transferFrom(address from, address to, uint256 tokenId) public;
    function approve(address to, uint256 tokenId) public;
    function getApproved(uint256 tokenId) public view returns (address operator);
    function setApprovalForAll(address operator, bool _approved) public;
    function isApprovedForAll(address owner, address operator) public view returns (bool);

    function safeTransferFrom(address from, address to, uint256 tokenId, bytes memory data) public;
}
Note: Not all resolvers support all functionality!

When in doubt call

supportsInterface(bytes4 interfaceID) returns (bool)
interface Resolver{
    event AddrChanged(bytes32 indexed node, address a);
    event AddressChanged(bytes32 indexed node, uint coinType, bytes newAddress);
    event NameChanged(bytes32 indexed node, string name);
    event ABIChanged(bytes32 indexed node, uint256 indexed contentType);
    event PubkeyChanged(bytes32 indexed node, bytes32 x, bytes32 y);
    event TextChanged(bytes32 indexed node, string indexed indexedKey, string key);
    event ContenthashChanged(bytes32 indexed node, bytes hash);

    function ABI(bytes32 node, uint256 contentTypes) external view returns (uint256, bytes memory);
    function addr(bytes32 node) external view returns (address);
    function addr(bytes32 node, uint coinType) external view returns(bytes memory);
    function contenthash(bytes32 node) external view returns (bytes memory);
    function dnsrr(bytes32 node) external view returns (bytes memory);
    function name(bytes32 node) external view returns (string memory);
    function pubkey(bytes32 node) external view returns (bytes32 x, bytes32 y);
    function text(bytes32 node, string calldata key) external view returns (string memory);
    function interfaceImplementer(bytes32 node, bytes4 interfaceID) external view returns (address);

    function setABI(bytes32 node, uint256 contentType, bytes calldata data) external;
    function setAddr(bytes32 node, address addr) external;
    function setAddr(bytes32 node, uint coinType, bytes calldata data) external;
    function setContenthash(bytes32 node, bytes calldata hash) external;
    function setDnsrr(bytes32 node, bytes calldata data) external;
    function setName(bytes32 node, string calldata _name) external;
    function setPubkey(bytes32 node, bytes32 x, bytes32 y) external;
    function setText(bytes32 node, string calldata key, string calldata value) external;
    function setInterface(bytes32 node, bytes4 interfaceID, address implementer) external;

    function supportsInterface(bytes4 interfaceID) external pure returns (bool);
}

{deprecated items removed}
// use simple names, e.g. "puppy" NOT "puppy.eth"

contract ETHRegistrarController is Ownable {

    uint constant public MIN_REGISTRATION_DURATION = 28 days;

    mapping(bytes32=>uint) public commitments;

    event NameRegistered(string name, bytes32 indexed label, address indexed owner, uint cost, uint expires);
    event NameRenewed(string name, bytes32 indexed label, uint cost, uint expires);
    event NewPriceOracle(address indexed oracle);

    function rentPrice(string memory name, uint duration) view public returns(uint);
    function valid(string memory name) public view returns(bool);
    function available(string memory name) public view returns(bool);
    function makeCommitment(string memory name, address owner, bytes32 secret) pure public returns(bytes32);
    function commit(bytes32 commitment) public;
    function register(string calldata name, address owner, uint duration, bytes32 secret) external payable;
    function renew(string calldata name, uint duration) external payable;
}
SBT-ETHEREUM ENS COMMANDS

> ensAddressLookup <ens-name>.eth
> ensAddressSet <ens-name>.eth <address-as-hex-or-ens-or-alias>

> ensAddressMultichainLookup <BTC|ETH|slip44-index> <ens-name>.eth
> ensAddressMultichainSet <BTC|ETH|slip44-index> <ens-name>.eth <address-as-hex-or-ens-or-alias>

> ensMigrateRegistrar <ens-name>.eth

> ensNameExtend <ens-name>.eth
> ensNameHashes <ens-name>.eth
> ensNamePrice <ens-name>.eth
> ensNameRegister <ens-name>.eth [optional-registrant-address] [optional-secret-from-prior-commitment]
> ensNameStatus <ens-name>.eth

> ensOwnerLookup <ens-name>.eth
> ensOwnerSet <ens-name>.eth <owner-address-as-hex-or-ens-or-alias>

> ensResolverLookup <ens-name>.eth
> ensResolverSet <ens-name>.eth [optional-resolver-address-as-hex-or-ens-or-alias]

> ensSubnodeCreate <full-subnode-ens-name>.eth
> ensSubnodeOwnerSet <full-subnode-ens-name>.eth <subnode-owner-as-hex-or-ens-or-alias>
Some Key Understandings

» An ENS name has an "owner" (which is an Ethereum address), and it may also have an "address"

» These are two very different things!

» The "owner" can set the "address" and many other things

» The "address" is where the money goes if you send ETH or transfer tokens to the ENS name.

» Registering a name sets the owner.

» For everything else, the owner needs to define a resolver.
REGISTERING A NAME

» It's easy, and interactive.

» But it requires two transactions and so is fragile!

» Just pay attention to the recovery command, which you can copy and paste if the second transaction fails.

> ensNameRegister shiningmonkey.eth
For how long would you like to rent the name (ex: "3 years")? 1 month
...
...
> ensNameStatus shiningmonkey.eth
[info] ENS name 'shiningmonkey.eth' is currently owned by '0x465e79b940bc2157e4259ff6b2d92f454497f1e4'.
[info] This registration will expire at 'Tue, 10 Dec 2019 14:33:54 -0800'.
[success] Total time: 2 s, completed Nov 10, 2019 4:58:55 AM
SETTING THE RESOLVER FOR A NAME

sbt-ethereum will help you through this if you forget this and jump to setting an address. But let's do it explicitly

```bash
> ensResolverSet shiningmonkey.eth
[warn] No resolver specified. Using default public resolver '0x226159d592e2b063810a10ebf6dcbada94ed8bb8'.

== TRANSACTION SIGNATURE REQUEST ==

> The transaction would be a message with...
  To: 0x314159265dd8dbb310642f98f50c066173c1259b (with aliases ['ens'] on chain with ID 1)
  From: 0x465e79b940bc2157e4259ff6b2d92f454497f1e4 (with aliases ['default-sender','testing0'] on chain with ID 1)
  Data: 0x1896f70a49763e65c2efcc46b84722d1358e19f41f5932f6db32400e399002828f451d500000000000000000226159d592e2b063810a10ebf6dcbada94ed8bb8
  Value: 0 ether

> According to the ABI currently associated with the 'to' address, this message would amount to the following method call...
  Function called: setResolver(bytes32,address)
    Arg 1 [name=node, type=bytes32]: 0x49763e65c2efcc46b84722d1358e19f41f5932f6db32400e399002828f451d5
    Arg 2 [name=resolver, type=address]: 0x226159d592e2b063810a10ebf6dcbada94ed8bb8

> The nonce of the transaction would be 524.

== TRANSACTION SIGNATURE REQUEST ==

> Would you like to sign this transaction? [y/n] y

[info] Unlocking address '0x465e79b940bc2157e4259ff6b2d92f454497f1e4' (on chain with ID 1, aliases ['default-sender','testing0'])
Enter passphrase or hex private key for address '0x465e79b940bc2157e4259ff6b2d92f454497f1e4': **************

[info] A transaction with hash '0x35b7d30ecc1ca53c002b4476d88657a7d0854f1a0724f9d17b9c2263fb842' has been submitted.
[info] Waiting up to 5 minutes for the transaction to be mined.
[info] The name 'shiningmonkey.eth' is now set to be resolved by a contract at '0x226159d592e2b063810a10ebf6dcbada94ed8bb8' (with aliases ['ens-public-resolver-2019-10-24'] on chain with ID 1).
[success] Total time: 100 s, completed Nov 10, 2019 4:42:41 AM
Setting the address for a name

> ensAddressSet shiningmonkey.eth default-sender
[warn] Gas price override set, default gas price plus a markup of 0.50 (50.00%), not subject to any cap or floor

=> TRANSACTION SIGNATURE REQUEST
=>
=> The transaction would be a message with...
=> To: 0x226159d592e2b06381a0ebf6dcbada94ed68b8 (with aliases ['ens-public-resolver-2019-10-24'] on chain with ID 1)
=> From: 0x465e79b940bc2157e4259ff6b2d92f454497f1e4 (with aliases ['default-sender','testing0'] on chain with ID 1)
=> Data: 0xd5fa2b049763e65c2efcc46b84722d1358e19f41fd5932f6db324800e39902828f451d50000000000000000000485e79b940bc2157e4259ff6b2d92f454497f1e4
=> Value: 0 ether

=> According to the ABI currently associated with the 'to' address, this message would amount to the following method call...
=> Function called: setAddr(bytes32,address)
=> Arg 1 [name=node, type=bytes32]: 0x49763e65c2efcc46b84722d1358e19f41fd5932f6db324800e39902828f451d5
=> Arg 2 [name=a, type=address]: 0x465e79b940bc2157e4259ff6b2d92f454497f1e4

=> The nonce of the transaction would be 525.

=> $$$ The transaction you have requested could use up to 66172 units of gas.
=> $$$ You would pay 3.9 gwei for each unit of gas, for a maximum cost of 0.0002580708 ether.
=> $$$ This is worth 0.05 USD (according to Coinbase at 5:14 AM).

Would you like to sign this transaction? [y/n] y

Enter passphrase or hex private key for address '0x465e79b940bc2157e4259ff6b2d92f454497f1e4': *******************

[info] Unlocking address '0x465e79b940bc2157e4259ff6b2d92f454497f1e4' (on chain with ID 1, aliases ['default-sender','testing0'])
[info] A transaction with hash '0x4a491b5f82d1de32add5f384f520b8c68e895333d416581c025366439a60b8e1' has been submitted.
[info] Waiting up to 5 minutes for the transaction to be mined.
[info] The name 'shiningmonkey.eth' now resolves to '0x465e79b940bc2157e4259ff6b2d92f454497f1e4' (with aliases ['default-sender','testing0'] on chain with ID 1).
[success] Total time: 54 s. completed Nov 10, 2019 5:15:30 AM
MULTICHAIN SUPPORT IS FUN & NEW

> ensAddressMultichainLookup BTC exigent.eth

[info] For coin 'BTC' with SLIP-44 Index 0, the name 'exigent.eth' resolves to address 18cjh41Ljp7CPzFZfrX45sdX9yKtaKXtPd, or binary-format:76a914538b134f052afc31504391632474579f2e62cf9288ac.

[success] Total time: 1 s, completed Nov 10, 2019 5:01:24 AM
REGISTERING BY HAND -- STEP 1
DEFINE AN ALIAS FOR THE BASE ENS

Note: The base ENS is the only guaranteed permanent ENS construct. Be careful about caching / giving aliases to the other constructs we will look up below! It's best to look them up fresh each time.

> ethAddressAliasSet ens 0x314159265dd8dbb310642f98f50c066173c1259b
[info] Alias 'ens' now points to address '0x314159265dd8dbb310642f98f50c066173c1259b' (for chain with ID 1).
[info] Refreshing caches.
[success] Total time: 0 s, completed Nov 10, 2019 2:19:11 AM
> ensNameHashes eth
[info] The ENS namehash of 'eth' is '0x93cdeb708b7545dc668eb9280176169d1c33cfd8ed6f04690a0bcc88a93fc4ae'.
[success] Total time: 0 s, completed Nov 10, 2019 2:22:48 AM
Registering by Hand -- Step 3

Discover the Resolver for the Top-Level ENS

> ethTransactionView ens resolver 0x93cdeb708b7545dc668eb9280176169d1c33cfd8ed6f04690a0bcc88a93fc4ae
[info] The function 'resolver' yields 1 result.
[info] + Result 1 of type 'address' is 0x97683a370239817cf33ec2c2ad3b3a1884571f69
[success] Total time: 1 s, completed Nov 10, 2019 2:24:30 AM

> ethContractAbiImport 0x97683a370239817cf33ec2c2ad3b3a1884571f69
An Etherscan API key has been set. Would you like to try to import the ABI for this address from Etherscan? [y/n] y
Attempting to fetch ABI for address '0x97683a370239817cf33ec2c2ad3b3a1884571f69' from Etherscan.
ABI found:
[...lots and lots of JSON here...]
Use this ABI? [y/n] y
[info] A default ABI is now known for the contract at address 0x97683a370239817cf33ec2c2ad3b3a1884571f69
[info] Refreshing caches.
[success] Total time: 8 s, completed Nov 10, 2019 2:53:20 AM
The Controller interface has an **EIP-165** interface ID of 0x018fac06.

Remember, the namehash of "eth" was

0x93cdeb708b7545dc668eb9280176169d1c33cfd8ed6f04690a0bcc88a93fc4ae
» Remember, our controller address was 0xb22c1c159d12461ea124b0deb4b5b93020e6ad16

» We'll price the shortest registration allowed

```
> ethTransactionView 0xb22c1c159d12461ea124b0deb4b5b93020e6ad16 available glowingmonkey
[info] The function 'available' yields 1 result.
[info] + Result 1 of type 'bool' is true
[success] Total time: 0 s, completed Nov 10, 2019 3:09:35 AM

> ethTransactionView 0xb22c1c159d12461ea124b0deb4b5b93020e6ad16 MIN_REGISTRATION_DURATION
[info] The function 'MIN_REGISTRATION_DURATION' yields 1 result.
[info] + Result 1 of type 'uint256' is 2419200
[success] Total time: 1 s, completed Nov 10, 2019 3:29:25 AM

> ethTransactionView 0xb22c1c159d12461ea124b0deb4b5b93020e6ad16 rentPrice glowingmonkey 2419200
[info] The function 'rentPrice' yields 1 result.
[info] + Result 1 of type 'uint256' is 2037241502249607
[success] Total time: 1 s, completed Nov 10, 2019 3:31:02 AM
```
REGISTERING BY HAND -- STEP 6
GENERATE A COMMITMENT

» To (somewhat) discourage front-running, the ENS controller requires a two-step registration.

» First we must make a commitment. To generate one, we'll need a "random" 32-byte secret.

» Remember, our controller address was 0xb22c1c159d12461ea124b0deb4b5b93020e6ad16

> ethUtilHashKeccak256 0xab569178
[info] 0x0439b438e7ea7ff3a664832ab05d5c9fd065bcdb5d1ff2b51631ed4b987bd5f1
[success] Total time: 0 s, completed Nov 10, 2019 3:07:00 AM

> ethTransactionView 0xb22c1c159d12461ea124b0deb4b5b93020e6ad16 makeCommitment
glowingmonkey default-sender 0x0439b438e7ea7ff3a664832ab05d5c9fd065bcdb5d1ff2b51631ed4b987bd5f1
[info] The function 'makeCommitment' yields 1 result.
[info]  + Result 1 of type 'bytes32' is 0xe51c895f09e8bd0670b8cd3e26679691b2750713c157ece007290d275972e8d1
[success] Total time: 1 s, completed Nov 10, 2019 3:13:54 AM
> ethTransactionInvoke 0xb22c1c159d12461ea124b8d4b693020e6ad16 commit 0xe51c895f09e8bd07b8cd3e2a6776961b2750713c157ecce087208d275972e8d1

>>> TRANSACTION SIGNATURE REQUEST

>>> The transaction would be a message with...
>>> To: 0xb22c1c159d12461ea124b8d4b693020e6ad16 (with aliases ['ens-controller-2019-11-10'] on chain with ID 1)
>>> From: 0x465e79b940bc2157e4259ff6b2d92f454497f1e4 (with aliases ['default-sender','testing0'] on chain with ID 1)
>>> Data: 0xf14fcbc8ec1c895f08b8b078c8d3e2b6776961b2750713c157ecce087208d275972e8d1
>>> Value: 0 ether

>>> According to the ABI currently associated with the 'to' address, this message would amount to the following method call...
>>> Function called: commit(bytes32)
>>> Arg 1 [name=commitment, type=bytes32]: 0xe51c895f09e8bd070b8cd3e2a6776961b2750713c157ecce087208d275972e8d1

>>> The nonce of the transaction would be 520.

>>> The transaction you have requested could use up to 53206 units of gas.
>>> You would pay 1.32 gwei for each unit of gas, for a maximum cost of 0.00007023192 ether.
>>> This is worth 0.01 USD (according to Coinbase at 3:16 AM).

Would you like to sign this transaction? [y/n] y

[info] Unlocking address '0x465e79b940bc2157e4259ff6b2d92f454497f1e4' (on chain with ID 1, aliases ['default-sender','testing0'])
Enter passphrase or hex private key for address '0x465e79b940bc2157e4259ff6b2d92f454497f1e4': *******************

[info] Called function 'commit', with args 'e51c895f09e8bd070b8cd3e2a6776961b2750713c157ecce087208d275972e8d1', sending 0 wei to address '0xb22c1c159d12461ea124b8d4b693020e6ad16' in transaction with hash '0x219559ad270662c8d022516aff91e8c51b333c77c91d6c5954c7dcb58bc74a'.

[info] Waiting for the transaction to be mined (will wait up to 5 minutes).

[info] Transaction Receipt:
[info]        Transaction Hash:    0x219559ad270662c8d022516aff91e8c51b333c77c91d6c5954c7dcb58bc74a
[info]        Transaction Index:   122
[info]        Transaction Status:  SUCCEEDED
[info]        Block Hash:          0xa4c40823f7cae8d90059fd1af4ce40021411097a5427e81324344ab79525b48e
[info]        Block Number:        8908146
[info]        From:                0x465e79b940bc2157e4259ff6b2d92f454497f1e4
[info]        To:                  0xb22c1c159d12461ea124b8d4b693020e6ad16
[info]        Cumulative Gas Used: 7455246
[info]        Gas Used:            44339
[info]        Contract Address:    None
[info]        Logs:                None
[info]        Events:              None

[success] Total time: 152 s, completed Nov 10, 2019 3:18:25 AM
REGISTERING BY HAND -- STEP 8
CHECK MINIMUM WAIT TIME

» It's currently 60 seconds...

> ethTransactionView 0xb22c1c159d12461ea124b0deb4b5b93020e6ad16 minCommitmentAge
[info] The function 'minCommitmentAge' yields 1 result.
[info] + Result 1 of type 'uint256' is 60
[success] Total time: 2 s, completed Nov 10, 2019 3:22:34 AM
REGISTERING BY HAND -- STEP 9
WAIT MINIMUM WAIT TIME

🎵 La la la la la la 🎵
REGISTERING BY HAND -- STEP 10
REGISTER THE NAME!

» We'll bump up the rent price we computed by a bit. Any "change" will be returned anyway.

```bash
> ethTransactionInvoke 0x82c2c1015d12461ae1240bbe645b93f026a610 register glowingmonkey default-sender 2419200
0x0439b438e7aa77ff3b64832a0d5c0f0d65bc0b51ff2b51631e4d09780d5f1 237273215fe2249687 wei

> T R A N S A C T I O N   S I G N A T U R E   R E Q U E S T

> The transaction would be a message with...
> To: 0x82c2c1015d12461ae1240bbe645b93f026a610 (with aliases ['ens-controller-2019-11-10'] on chain with ID 1)
> From: 0x465e79b940bc2157e4255b93564b544971e4 (with aliases ['default-sender','testing0'] on chain with ID 1)
> Data: 0x85f6d1550000000000000000000000000000000000000000000000000000000000000080000000000000000000000000465e79b9...
> Value: 0.002237241502249607 ether

> According to the ABI currently associated with the 'to' address, this message would amount to the following method call...
> Function called: register(string,address,uint256,bytes32)
> Arg 1 [name=string]: "glowingmonkey"
> Arg 2 [name=owner, type=address]: 0x465e79b940bc2157e4255b93564b544971e4
> Arg 3 [name=duration, type=uint256]: 2419200
> Arg 4 [name=secret, type=bytes32]: 0x0439b438e7aa77ff3b64832a0d5c0f0d65bc0b51ff2b51631e4d09780d5f1

> The nonce of the transaction would be 521.

> $$$ The transaction you have requested could use up to 183518 units of gas.
> $$$ You would pay 1.4 gwei for each unit of gas, for a maximum cost of 0.0002569252 ether.
> $$$ This is worth 0.05 USD (according to Coinbase at 3:35 AM).
> $$$ Would you also send 0.002237241502249607 ether (0.42 USD), for a maximum total cost of 0.002494166702249607 ether (0.47 USD).

Would you like to sign this transaction? [y/n] y

Enter passphrase or hex private key for address '0x465e79b940bc2157e4255b93564b544971e4': *******************

[info] Unlocking address '0x82c2c1015d12461ae1240bbe645b93f026a610' (on chain with ID 1, aliases ['ens-controller-2019-11-10'])

[info] Called function 'register', with args 'glowingmonkey'. 0x0439b438e7aa77ff3b64832a0d5c0f0d65bc0b51ff2b51631e4d09780d5f1, sending 237273215fe2249687 wei to address '0xb2c2c1015d12461ae1240bbe645b93f026a610' in transaction with hash '0x82c340b3a9a7ac0f1ab2e4a4b3576b93d610'.

[info] Waiting for the transaction to be mined (will wait up to 5 minutes).```
REGISTERING BY HAND
HOORAY!

> ethTransactionView 0xb22c1c159d12461ea124b0deb4b5b93020e6ad16 available glowingmonkey
[info] The function 'available' yields 1 result.
[info] + Result 1 of type 'bool' is false
[success] Total time: 1 s, completed Nov 10, 2019 3:51:44 AM
Ethereum Name Service Lookup (EWHOIS)

Ethereum Name Service (ENS) is a distributed, extensible naming system based on the Ethereum blockchain that can be used to resolve a wide variety of resources:

glowingmonkey.eth

- Label Hash [glowingmonkey]: 0x15bfa48468b499f33ccc5d41f8d88b42bc899939c155870bd2f52acc9921dd573
- [glowingmonkey.eth]: 0x692fcde2eb49f55f72ce3ef90d0ce0e6e67ed3a08bd13ee90edbe5d4a8777ff5

⚠️ The ENS Name 'glowingmonkey.eth' is Not Available for Reservation
THANKS!