Go Documentation Server

Package onens

```
import "github.com/jw-1ns/go-1ns"
```


Overview

Index

func (c *RegistrarController) CommitmentHash(opts *bind.CallOpts, domain string, owner common.Address, duration *big.Int, secret [32]byte, resolver common.Address, data [][]byte, reverseRecord bool, fuses uint32, wrapperExpiry uint64) (common.Hash, error)
func (c *RegistrarController) Commitments(opts *bind.CallOpts, commitment [32]byte) (*big.Int, error)
func (c *RegistrarController) IsAvailable(domain string) (bool, error)
func (c *RegistrarController) IsValid(domain string) (bool, error)
func (c *RegistrarController) MaxCommitmentInterval(opts *bind.CallOpts) (*big.Int, error)
func (c *RegistrarController) MinCommitmentInterval() (*big.Int, error)
func (c *RegistrarController) MinRegistrationDuration() (time.Duration, error)
func (c *RegistrarController) NameWrapper() (common.Address, error)
func (c *RegistrarController) Owner() (common.Address, error)
func (c *RegistrarController) Prices() (common.Address, error)
func (c *RegistrarController) RecoverFunds(opts *bind.TransactOpts, token common.Address, to common.Address, amount *big.Int) (*types.Transaction, error)
func (c *RegistrarController) Register(opts *bind.TransactOpts, name string, owner common.Address, duration *big.Int, secret [32]byte, resolver common.Address, data [][]byte, reverseRecord bool, fuses uint32, wrapperExpiry uint64) (*types.Transaction, error)
func (c *RegistrarController) Renew(opts *bind.TransactOpts, name string, duration *big.Int) (*types.Transaction, error)
func (c *RegistrarController) RenewWithFuses(opts *bind.TransactOpts, name string, duration *big.Int, fuses uint32, wrapperExpiry uint64) (*types.Transaction, error)
func (c *RegistrarController) RenounceOwnership(opts *bind.TransactOpts) (*types.Transaction, error)
func (c *RegistrarController) RentPrice(opts *bind.CallOpts, name string, duration *big.Int) (registrarcontroller.IPriceOraclePrice, error)
func (c *RegistrarController) ReverseRegistrar() (common.Address, error)
func (c *RegistrarController) SupportsInterface(opts *bind.CallOpts, interfaceID [4]byte) (bool, error)
func (c *RegistrarController) TransferOwnership(opts *bind.TransactOpts, newOwner common.Address) (*types.Transaction, error)
func (c *RegistrarController) Withdraw(opts *bind.TransactOpts) (*types.Transaction, error)

type Registry

func NewRegistry(backend bind.ContractBackend) (*Registry, error)
func NewRegistryAt(backend bind.ContractBackend, address common.Address) (*Registry, error)
func (r *Registry) Owner(name string) (common.Address, error)
func (r *Registry) Resolver(name string) (*Resolver, error)
func (r *Registry) ResolverAddress(name string) (common.Address, error)
func (r *Registry) SetOwner(opts *bind.TransactOpts, name string, address common.Address) (*types.Transaction, error)
func (r *Registry) SetResolver(opts *bind.TransactOpts, name string, address common.Address) (*types.Transaction, error)
func (r *Registry) SetSubdomainOwner(opts *bind.TransactOpts, name string, subname string, address common.Address) (*types.Transaction, error)

type Resolver

func NewResolver(backend bind.ContractBackend, domain string) (*Resolver, error)
func NewResolverAt(backend bind.ContractBackend, domain string, address common.Address) (*Resolver, error)
func (r *Resolver) ABI(name string) (string, error)
func (r *Resolver) Address() (common.Address, error)
func (r *Resolver) Contenthash() ([]byte, error)
func (r *Resolver) InterfaceImplementer(interfaceID [4]byte) (common.Address, error)
func (r *Resolver) MultiAddress(coinType uint64) ([]byte, error)
func (r *Resolver) PubKey() ([32]byte, [32]byte, error)
func (r *Resolver) SetABI(opts *bind.TransactOpts, name string, abi string, contentType *big.Int) (*types.Transaction, error)

func (r *Resolver) SetContenthash(opts *bind.TransactOpts, contenthash []byte) (*types.Transaction, error)
func (r *Resolver) SetPubKey(opts *bind.TransactOpts, x [32]byte, y [32]byte) (*types.Transaction, error)
func (r *Resolver) SetText(opts *bind.TransactOpts, name string, value string) (*types.Transaction, error)
func (r *Resolver) Text(name string) (string, error)

Package files

config.go contenthash.go ensregistry.go misc.go name.go namehash.go publicresolver.go registrarcontroller.go

Variables

UnknownAddress is the address to which unknown entries resolve

```
var UnknownAddress = common.HexToAddress("00")
```

func **ContenthashToString**

```
func ContenthashToString(bytes []byte) (string, error)
```

ContenthashToString turns EIP-1577 binary format in to EIP-1577 text format

func **CreateRegistrySession**

```
func CreateRegistrySession(chainID *big.Int, wallet *accounts.Wallet, account
*accounts.Account, passphrase string, contract *ensregistry.Contract, gasPrice
*big.Int) *ensregistry.ContractSession
```

CreateRegistrySession creates a session suitable for multiple calls

func **Domain**

```
func Domain(domain string) string
```

Domain obtains the domain of an ENS name, including subdomains. It does this by removing everything up to and including the first period. For example, 'country' will return ”

```
'foo.country' will return 'country'
'bar.foo.country' will return 'foo.country'
```

func **DomainLevel**

```
func DomainLevel(name string) (level int)
```

DomainLevel calculates the level of the domain presented. A top-level domain (e.g. 'country') will be 0, a domain (e.g. 'foo.country') will be 1, a subdomain (e.g. 'bar.foo.country' will be 2, etc.

func **DomainPart**

```
func DomainPart(domain string, part int) (string, error)
```

DomainPart obtains a part of a name Positive parts start at the lowest-level of the domain and work towards the top-level domain. Negative parts start at the top-level domain and work towards the lowest-level domain. For example, with a domain bar.foo.com the following parts will be returned: Number | part

```
 1 |  bar
 2 |  foo
 3 |  com
-1 |  com
-2 |  foo
-3 |  bar
```

func **LabelHash**

```
func LabelHash(label string) (hash [32]byte, err error)
```

LabelHash generates a simple hash for a piece of a name.

func **NameHash**

```
func NameHash(name string) (hash [32]byte, err error)
```

NameHash generates a hash from a name that can be used to look up the name in ENS

func **NormaliseDomain**

```
func NormaliseDomain(domain string) (string, error)
```

NormaliseDomain turns ENS domain in to normal form

func **NormaliseDomainStrict**

```
func NormaliseDomainStrict(domain string) (string, error)
```

NormaliseDomainStrict turns ENS domain in to normal form, using strict DNS rules (e.g. no underscores)

func **Normalize**

```
func Normalize(input string) (output string, err error)
```

Normalize normalizes a name according to the ENS rules

func **PublicResolverAddress**

```
func PublicResolverAddress(backend bind.ContractBackend) (common.Address, error)
```

PublicResolverAddress obtains the address of the public resolver for a chain

func **RegistryContractAddress**

```
func RegistryContractAddress(backend bind.ContractBackend) (common.Address,
error)
```

RegistryContractAddress obtains the address of the registry contract for a chain. This is (currently) the same for all chains.

func **Resolve**

```
func Resolve(backend bind.ContractBackend, input string) (address common.Address,
err error)
```

Resolve resolves an ENS name in to an Etheruem address This will return an error if the name is not found or otherwise 0

func **SetResolver**

```
func SetResolver(session *ensregistry.ContractSession, name string, resolverAddr
*common.Address) (*types.Transaction, error)
```

SetResolver sets the resolver for a name

func **SetSubdomainOwner**

```
func SetSubdomainOwner(session *ensregistry.ContractSession, name string,
subdomain string, ownerAddr *common.Address) (*types.Transaction, error)
```

SetSubdomainOwner sets the owner for a subdomain of a name

func **StringToContenthash**

```
func StringToContenthash(text string) ([]byte, error)
```

StringToContenthash turns EIP-1577 text format in to EIP-1577 binary format

func **Tld**

```
func Tld(domain string) string
```

Tld obtains the top-level domain of an ENS name

func **UnqualifiedName**

```
func UnqualifiedName(domain string, root string) (string, error)
```

UnqualifiedName strips the root from the domain and ensures the result is suitable as a name

type **Name**

Name represents an ENS name, for example 'foo.bar.eth'.

```
type Name struct {

    // Name is the fully-qualified name of an ENS domain e.g. foo.bar.eth
    Name string
    // Domain is the domain of an ENS domain e.g. bar.eth
    Domain string
    // Label is the name part of an ENS domain e.g. foo
    Label string
    // contains filtered or unexported fields
}
```

func NewName

```
func NewName(backend bind.ContractBackend, name string) (*Name, error)
```

NewName creates an ENS name structure. Note that this does not create the name on-chain.

func (*Name) Controller

```
func (n *Name) Controller() (common.Address, error)
```

Controller obtains the controller for this name. The controller can carry out operations on the name such as setting records, but cannot transfer ultimate ownership of the name.

func (*Name) RegistrationInterval

```
func (n *Name) RegistrationInterval() (time.Duration, error)
```

RegistrationInterval obtains the minimum interval between commit and reveal when registering this name.

type RegistrarController

RegistrarController is the structure for the registrar controller contract

```
type RegistrarController struct {
    Contract     *registrarcontroller.Contract
    ContractAddr common.Address
    // contains filtered or unexported fields
}
```

func NewRegistrarControllerAt

```
func NewRegistrarControllerAt(backend bind.ContractBackend, domain string,
address common.Address) (*RegistrarController, error)
```

NewRegistrarControllerAt creates a registrar controller at a given address

func (*RegistrarController) BaseNode

```
func (c *RegistrarController) BaseNode(opts *bind.CallOpts) ([32]byte, error)
```

"function baseNode() view returns (bytes32)", BaseNode retreives the base node

func (*RegistrarController) Basextension

```
func (c *RegistrarController) Basextension() (string, error)
```

"function baseExtension() view returns (string)", BaseExtension retrieves the baseExtension

func (*RegistrarController) Commit

```
func (c *RegistrarController) Commit(opts *bind.TransactOpts, commitment
[32]byte) (*types.Transaction, error)
```

"function commit(bytes32)", Commit sends a commitment to register a domain.

func (*RegistrarController) CommitmentHash

```
func (c *RegistrarController) CommitmentHash(opts *bind.CallOpts, domain string,
owner common.Address, duration *big.Int, secret [32]byte, resolver
common.Address, data [][]byte, reverseRecord bool, fuses uint32, wrapperExpiry
uint64) (common.Hash, error)
```

"function makeCommitment(string,address,uint256,bytes32,address,bytes[],bool,uint32,uint64) pure returns (bytes32)", CommitmentHash returns the commitment hash for a label/owner/secret tuple

func (*RegistrarController) Commitments

```
func (c *RegistrarController) Commitments(opts *bind.CallOpts, commitment
[32]byte) (*big.Int, error)
```

"function commitments(bytes32) view returns (uint256)" Commitments returns the block timestamp of the commmitment

func (*RegistrarController) IsAvailable

```
func (c *RegistrarController) IsAvailable(domain string) (bool, error)
```

"function available(string) view returns (bool)", IsAvailable returns true if the domain is available for registration.

func (*RegistrarController) IsValid

```
func (c *RegistrarController) IsValid(domain string) (bool, error)
```

IsValid returns true if the domain is considered valid by the controller.

func (*RegistrarController) MaxCommitmentInterval

```
func (c *RegistrarController) MaxCommitmentInterval(opts *bind.CallOpts)
(*big.Int, error)
```

"function maxCommitmentAge() view returns (uint256)",

func (*RegistrarController) MinCommitmentInterval

```
func (c *RegistrarController) MinCommitmentInterval() (*big.Int, error)
```

"function minCommitmentAge() view returns (uint256)", MinCommitmentInterval returns the minimum time that has to pass between a commit and reveal

func (*RegistrarController) MinRegistrationDuration

```
func (c *RegistrarController) MinRegistrationDuration() (time.Duration, error)
```

```
"function MIN_REGISTRATION_DURATION() view returns (uint256)",
```

MinRegistrationDuration returns the minimum duration for which a name can be registered

func (*RegistrarController) NameWrapper

```
func (c *RegistrarController) NameWrapper() (common.Address, error)
```

"function nameWrapper() view returns (address)",

func (*RegistrarController) Owner

```
func (c *RegistrarController) Owner() (common.Address, error)
```

"function owner() view returns (address)",

func (*RegistrarController) Prices

```
func (c *RegistrarController) Prices() (common.Address, error)
```

"function prices() view returns (address)",

func (*RegistrarController) RecoverFunds

```
func (c *RegistrarController) RecoverFunds(opts *bind.TransactOpts, token
common.Address, to common.Address, amount *big.Int) (*types.Transaction, error)
```

"function recoverFunds(address,address,uint256)",

func (*RegistrarController) Register

```
func (c *RegistrarController) Register(opts *bind.TransactOpts, name string,
owner common.Address, duration *big.Int, secret [32]byte, resolver
common.Address, data [][]byte, reverseRecord bool, fuses uint32, wrapperExpiry
uint64) (*types.Transaction, error)
```

"function register(string,address,uint256,bytes32,address,bytes[],bool,uint32,uint64) payable",

func (*RegistrarController) Renew

```
func (c *RegistrarController) Renew(opts *bind.TransactOpts, name string,
duration *big.Int) (*types.Transaction, error)
```

"function renew(string,uint256) payable",

func (*RegistrarController) RenewWithFuses

```
func (c *RegistrarController) RenewWithFuses(opts *bind.TransactOpts, name
string, duration *big.Int, fuses uint32, wrapperExpiry uint64)
(*types.Transaction, error)
```

"function renewWithFuses(string,uint256,uint32,uint64) payable",

func (*RegistrarController) RenounceOwnership

```
func (c *RegistrarController) RenounceOwnership(opts *bind.TransactOpts)
(*types.Transaction, error)
```

"function renounceOwnership()",

func (*RegistrarController) RentPrice

```
func (c *RegistrarController) RentPrice(opts *bind.CallOpts, name string,
duration *big.Int) (registrarcontroller.IPriceOraclePrice, error)
```

"function rentPrice(string,uint256) view returns (tuple(uint256,uint256))",

func (*RegistrarController) ReverseRegistrar

```
func (c *RegistrarController) ReverseRegistrar() (common.Address, error)
```

"function reverseRegistrar() view returns (address)",

func (*RegistrarController) SupportsInterface

```
func (c *RegistrarController) SupportsInterface(opts *bind.CallOpts, interfaceID
[4]byte) (bool, error)
```

"function supportsInterface(bytes4) pure returns (bool)",

func (*RegistrarController) TransferOwnership

```
func (c *RegistrarController) TransferOwnership(opts *bind.TransactOpts, newOwner
common.Address) (*types.Transaction, error)
```

"function transferOwnership(address)",

func (*RegistrarController) Withdraw

```
func (c *RegistrarController) Withdraw(opts *bind.TransactOpts)
(*types.Transaction, error)
```

"function withdraw()"

type Registry

Registry is the structure for the ensregistry contract

```
type Registry struct {
    Contract     *ensregistry.Contract
    ContractAddr common.Address
    // contains filtered or unexported fields
}
```

func NewRegistry

```
func NewRegistry(backend bind.ContractBackend) (*Registry, error)
```

NewRegistry obtains the ENS registry

func NewRegistryAt

```
func NewRegistryAt(backend bind.ContractBackend, address common.Address)
(*Registry, error)
```

NewRegistryAt obtains the ENS registry at a given address

func (*Registry) Owner

```
func (r *Registry) Owner(name string) (common.Address, error)
```

Owner returns the address of the owner of a name

func (*Registry) Resolver

```
func (r *Registry) Resolver(name string) (*Resolver, error)
```

Resolver returns the resolver for a name

func (*Registry) ResolverAddress

```
func (r *Registry) ResolverAddress(name string) (common.Address, error)
```

ResolverAddress returns the address of the resolver for a name

func (*Registry) SetOwner

```
func (r *Registry) SetOwner(opts *bind.TransactOpts, name string, address
common.Address) (*types.Transaction, error)
```

SetOwner sets the ownership of a domain

func (*Registry) SetResolver

```
func (r *Registry) SetResolver(opts *bind.TransactOpts, name string, address
common.Address) (*types.Transaction, error)
```

SetResolver sets the resolver for a name

func (*Registry) SetSubdomainOwner

```
func (r *Registry) SetSubdomainOwner(opts *bind.TransactOpts, name string,
subname string, address common.Address) (*types.Transaction, error)
```

SetSubdomainOwner sets the ownership of a subdomain, potentially creating it in the process

type Resolver

Resolver is the structure for the public resolver contract

```
type Resolver struct {
    Contract     *publicresolver.Contract
    ContractAddr common.Address
    // contains filtered or unexported fields
}
```

func NewResolver

```
func NewResolver(backend bind.ContractBackend, domain string) (*Resolver, error)
```

NewResolver obtains an Public resolver for a given domain

func NewResolverAt

```
func NewResolverAt(backend bind.ContractBackend, domain string, address
common.Address) (*Resolver, error)
```

NewResolverAt obtains an ENS resolver at a given address

func (*Resolver) ABI

```
func (r *Resolver) ABI(name string) (string, error)
```

ABI returns the ABI associated with a name

func (*Resolver) Address

```
func (r *Resolver) Address() (common.Address, error)
```

Address returns the Ethereum address of the domain

func (*Resolver) Contenthash

```
func (r *Resolver) Contenthash() ([]byte, error)
```

Contenthash returns the content hash of the domain

func (*Resolver) InterfaceImplementer

```
func (r *Resolver) InterfaceImplementer(interfaceID [4]byte) (common.Address,
error)
```

InterfaceImplementer returns the address of the contract that implements the given interface for the given domain

func (*Resolver) MultiAddress

```
func (r *Resolver) MultiAddress(coinType uint64) ([]byte, error)
```

MultiAddress returns the address of the domain for a given coin type. The coin type is as per https://github.com/satoshilabs/slips/blob/master/slip-0044.md

func (*Resolver) PubKey

```
func (r *Resolver) PubKey() ([32]byte, [32]byte, error)
```

PubKey returns the public key of the domain

func (*Resolver) SetABI

```
func (r *Resolver) SetABI(opts *bind.TransactOpts, name string, abi string,
contentType *big.Int) (*types.Transaction, error)
```

SetABI sets the ABI associated with a name

func (*Resolver) SetContenthash

```
func (r *Resolver) SetContenthash(opts *bind.TransactOpts, contenthash []byte)
(*types.Transaction, error)
```

SetContenthash sets the content hash of the domain

func (*Resolver) SetPubKey

```
func (r *Resolver) SetPubKey(opts *bind.TransactOpts, x [32]byte, y [32]byte)
(*types.Transaction, error)
```

SetPubKey sets the public key of the domain

func (*Resolver) SetText

```
func (r *Resolver) SetText(opts *bind.TransactOpts, name string, value string)
(*types.Transaction, error)
```

SetText sets the text associated with a name

func (*Resolver) Text

```
func (r *Resolver) Text(name string) (string, error)
```

Text obtains the text associated with a name

Subdirectories

Name
..
contracts
baseregistrar
ensregistry
publicresolver
registrarcontroller
util

Build version go1.18.9.
Except as noted, the content of this page is licensed under the Creative Commons Attribution 3.0 License, and code is licensed under a BSD license.
Terms of Service | Privacy Policy