

Package hns

```
import "github.com/harmony-domains/coredns-hns"
```

[Overview](#)[Index](#)[Subdirectories](#)

Overview ▾

Package hns implements a plugin that returns information held in the Ethereum Name Service.

Index ▾

type HNS

func (e HNS) HasRecords(domain string, name string) (bool, error)

func (e HNS) IsAuthoritative(domain string) bool

func (e HNS) Name() string

func (e HNS) Query(domain string, name string, qtype uint16, do bool) ([]dns.RR, error)

func (e HNS) Ready() bool

func (e HNS) ServeDNS(ctx context.Context, w dns.ResponseWriter, r *dns.Msg) (int, error)

type Result

func Lookup(server Server, state request.Request) ([]dns.RR, []dns.RR, []dns.RR, Result)

type Server

Package files

[dname.go](#) [hns.go](#) [server.go](#) [setup.go](#) [wildcard.go](#)

type HNS

HNS is a plugin that returns information held in the Ethereum Name Service.

```
type HNS struct {
    Next          plugin.Handler
    Client        *ethclient.Client
    Registry      *hns.Registry
    EthLinkNameServers []string
    IPFSGatewayAs  []string
    IPFSGatewayAAAAAs []string
}
```

func (HNS) HasRecords

```
func (e HNS) HasRecords(domain string, name string) (bool, error)
```

HasRecords checks if there are any records for a specific domain and name. This is used for wildcard eligibility

func (HNS) IsAuthoritative

```
func (e HNS) IsAuthoritative(domain string) bool
```

IsAuthoritative checks if the HNS plugin is authoritative for a given domain

func (HNS) Name

```
func (e HNS) Name() string
```

Name implements the Handler interface.

func (HNS) Query

```
func (e HNS) Query(domain string, name string, qtype uint16, do bool) ([]dns.RR, error)
```

Query queries a given domain/name/resource combination

func (HNS) Ready

```
func (e HNS) Ready() bool
```

Ready returns true if we're ready to serve DNS records i.e. our chain is synced

func (HNS) ServeDNS

```
func (e HNS) ServeDNS(ctx context.Context, w dns.ResponseWriter, r *dns.Msg) (int, error)
```

ServeDNS implements the plugin.Handler interface.

type Result

Result of a lookup

```
type Result int
```

```
const (
    // Success is a successful lookup.
    Success Result = iota
    // NameError indicates a nameerror
    NameError
    // Delegation indicates the lookup resulted in a delegation.
    Delegation
    // NoData indicates the lookup resulted in a NODATA.
    NoData
    // ServerFailure indicates a server failure during the lookup.
    ServerFailure
)
```

func Lookup

```
func Lookup(server Server, state request.Request) ([]dns.RR, []dns.RR, []dns.RR, Result)
```

Lookup contains the logic required to move through A DNS hierarchy and gather the appropriate records

type Server

Server is an interface defined by any plugin that wishes to serve authoritative records

```
type Server interface {
    // Query returns records for a specific domain, name, and resource type
    Query(domain string, qname string, qtype uint16, do bool) ([]dns.RR, error)

    // HasRecords checks if there are any records for a specific domain and name
    // This is used to check for wildcard eligibility
    HasRecords(domain string, qname string) (bool, error)

    // IsAuthoritative returns true if this server is authoritative for the
    // supplied domain
    IsAuthoritative(qdomain string) bool
}
```

Subdirectories

Name

..
build

- coredns
 - core
 - dnsserver
 - plugin
 - coremain
 - pb
 - plugin
 - acl
 - any
 - auto
 - autopath
 - azure
 - bind
 - bufsize
 - cache
 - freq
 - cancel
 - chaos
 - clouddns
 - debug
 - deprecated
 - dns64
 - dnssec
 - dnstap
 - msg
 - erratic
 - errors
 - etcd
 - msg
 - file
 - rrutil
 - tree
 - forward
 - grpc
 - health
 - hosts
 - k8s_external
 - kubernetes
 - object
 - loadbalance
 - local
 - log
 - loop
 - metadata
 - metrics

vars
nsid
pkg
cache
dnstest
dnsutil
doh
edns
fall
fuzz
log
nonwriter
parse
rcode
replacer
response
reuseport
singleflight
tls
trace
transport
uniq
up
upstream
pprof
ready
reload
rewrite
root
route53
secondary
sign
template
test
tls
trace
transfer
whoami
request
test

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](#)