NAME

autoreverse - a minimalist-configuration reverse DNS name server

SYNOPSIS

autoreverse -h | **--help** | **--manpage** | **-v** | **--version**
autoreverse --forward *zone-name* | **--local-forward** *zone-name*
--reverse *CIDR*<?> | **--local-reverse** *CIDR*<?>
[**--listen** *listen-address*]<?> [**--PTR-deduce** *URL*]<?> [**--passthru** *auth-server*]
[**--synthesize**=*true*] [**--CHAOS**=*true*] [**--NSID** *hostid*] [**--TTL** *time.Duration=1h*]
[**--user** *user-name*] [**--group** *group-name*] [**--chroot** *path*]
[**--log-debug**] [**--log-major**=*true*] [**--log-minor**] [**--log-queries**=*true*]
[**--report** *time.Duration=1h*]

The ‘<?>’ symbol indicates options which can be specified multiple times.

DESCRIPTION

autoreverse is a specialized authoritative DNS server whose goal is to make it as easy as possible to auto-answer reverse queries without ever requiring reverse zone files. **autoreverse** synthesizes reverse answers and automatically derives **PTR** answers from specified forward zones. Importantly, **autoreverse** automatically answers *forward* queries corresponding to the synthetic reverse answers, meeting the requirements of many remote services which insist on matching forward and reverse names.

autoreverse is designed to run on residential gateway routers and servers behind NATs which acquire ISP-assigned addresses via DHCP or SLAAC, but it also runs on publicly accessible servers with static network configurations.

On start-up, **autoreverse** extracts forward and reverse delegation details from the DNS to synthesize its ‘Zones of Authority’. This approach to gleaning information from the DNS represents an over-arching philosophy of **autoreverse** in that it *never* requires configuration material which duplicates that already present in the DNS. This approach is suggested in rfc8501, Section 2.5 **[RFC8501]**.

While **autoreverse** normally runs with pre-configured forward and reverse delegations in the global DNS, it also supports non-delegated rfc1918 **[RFC1918]** and rfc4193 **[RFC4193]** addresses, otherwise known as private addresses or User Local addresses (**ULAs**).

See *GETTING STARTED* for details on how to run **autoreverse**, but a typical invocation is:

```
# autoreverse --forward autoreverse.example.net --reverse 2001:db8::/64
```

Where ‘autoreverse.example.net’ and ‘0.0.0.0.0.0.0.0.8.b.d.0.1.0.0.2.ip6.arpa’ (the reverse zone for

2001:db8::/64) are delegated to the **autoreverse** listening addresses(s).

That’s it! That’s all that’s needed to serve your reverse and matching forward queries.

OPTIONS

Documentation Options are:

-h | **--help**

print command-line usage

--manpage

print the raw mandoc text suitable for piping into mandoc(1).

-v | **--version**

print **autoreverse** version and origin URL

Run-time Options are:

--CHAOS [=false]

Answer **CHAOS TXT** queries for version.bind, version.server, authors.bind, hostname.bind and id.server. The default is **true**.

--NSID *hostid*

Respond to **EDNS NSID** sub-option requests with the specified *hostid*. **autoreverse** manages the hexadecimal conversion implicitly so this option should be set as a regular string, e.g. ‘--NSID a.ns.example.net’.

--PTR-deduce *URL*

The zone is loaded from the URL and scanned for address records to deduce **PTR** answers in preference to synthetic answers. This is a way of overlaying synthetic answers with zone data.

Any **PTR RRs** found in the zone are also loaded in preference to synthetic answers. Supported URL schemes are: ‘file’, ‘axfr’, ‘http’ and ‘https’.

In all cases, address and **PTR** records are only considered if they are in-bailiwick of **--reverse** or **--local-reverse** zones.

CNAME RRs are resolved while loading zones, thus if a CNAME resolves to an in-bailiwick reverse address it’s included as a preferred response to PTR queries. A classic use of this is when you have a CNAME to a **RIPE** Atlas probe such as:

```
myprobe.example.net. 600 IN CNAME pxxxx.probes.atlas.ripe.net.
```

in this case a reverse query of the probe address returns 'myprobe.example.net'.

The **--PTR-deduce** URLs are loaded after **--chroot** processing which means paths in 'file' scheme URLs must be relative to the chroot directory.

The reload strategy varies with the URL scheme: 'file' periodically detects Date-Time-Modified changes while the other schemes rely on the **SOA** 'Refresh' value expiring.

If any of the **--PTR-deduce** URLs fail to load, the previous data for **all** zones is retained and the partially loaded new values are discarded. If the initial load of any zone fails, **autoreverse** exits. In other words, **autoreverse** continues to run with stale data, but does not start with missing data.

Examples of syntactically valid **--PTR-deduce** URLs:

```
axfr://a.ns.example.org/example.net
file:///etc/nsd/data/example.net.zone
https://www.example.com/example.org.txt
```

The **--PTR-deduce** option can be specified multiple times.

--TTL *time.Duration*

'Time To Live' for synthetic responses expressed in 'go' **time.Duration** syntax. The minimum value allowed is '1s' and the default is '1h'.

--chroot *Path*

Reduce process privileges by issuing chroot(2) after **--listen** sockets have been established.

In conjunction with **--user** and **--group**, this option restricts access to a subset of the file system. **autoreverse** must start as root for this option to succeed. The specified path must be an absolute path otherwise chroot(2) fails.

There are caveats with **--chroot**. Once the chroot directory is set, **all** file access is relative to that directory. This obviously affects 'file' URLs given to **--PTR-deduce** but it also potentially impacts name resolution which often relies on files such as **/etc/resolv.conf**. In other words, if not properly established, **--chroot** can cause name resolution to fail.

autoreverse defers all zone loading and discovery until after process privileges are reduced so any problems with chroot and friends are exposed at start up.

--forward *Domain*

The forward zone to discover and serve. Also used as the suffix domain in **PTR** responses. This zone *must* be delegated to an **autoreverse** listening address.

autoreverse queries the DNS to extract delegation details for this zone from the parent zone. If one of the zone name servers self-identifies, as determined by DNS Probing, the zone is accepted as a ‘Zone of Authority’. If the zone does not self-identify, **autoreverse** exits. Only one of **--forward** or **--local-forward** can be specified.

autoreverse processes **--forward** before **--reverse** which means reverse zones can refer to forward zone name servers and discovery will ‘just work’ as **autoreverse** is in a position to answer forward zone queries.

autoreverse synthesizes zone information from the delegation details.

--group *group-name*

Reduce privileges by issuing a setgid(2) after **--listen** sockets have been established.

In conjunction with **--user** and **--chroot**, this option removes root privileges and restricts access to other system components. **autoreverse** must start as root for this option to succeed.

--listen *listen-address*

Address to listen on for DNS queries. If just an IP address or host name is specified, **autoreverse** assumes the ‘domain’ service (aka port 53). A specific port can be provided with the usual ‘host:port’, ‘v4address:port’ or ‘[v6address]:port’ syntax.

However the port is determined, on most Unix-like systems, **autoreverse** normally needs to be started as root to listen on ‘privileged ports’ such as port 53. If started as root, it is highly recommended that the **autoreverse** invocation include the **--user**, **--group** and **--chroot** options to reduce process privileges once the **-listen** sockets have been established.

autoreverse listens on both **UDP** and **TCP** networks for **DNS** queries. The default is ‘:domain’.

The **--listen** option can be specified multiple times.

--local-forward *Domain*

A local forward zone to serve as a ‘Zone of Authority’. Unlike **--forward**, no attempt is made to discover the delegation and self-identify the name server. A skeletal SOA is created and **autoreverse** arbitrarily serves the domain and uses it as a suffix for synthetic **PTR** generation.

Domain represents a zone which is not expected to be visible in the public DNS and is thus only visible locally where local resolvers are configured to direct such queries to **autoreverse**.

Only one of **--forward** or **--local-forward** can be specified.

--local-reverse *CIDR*

CIDR of a local reverse zone to serve as a ‘Zone of Authority’. Intended for rfc1918 and rfc4193 addresses otherwise known as private addresses or Unique Local Addresses in **ipv6** parlance. Unlike **--reverse** no attempt is made to discover the delegation and self-identify the name server. A skeletal SOA is created and **autoreverse** arbitrarily serves the reverse domain.

The **CIDR** represents a zone which is not expected to be visible in the public DNS and is thus only visible locally where local resolvers are configured to direct reverse queries to **autoreverse**. How this redirection is achieved varies greatly depending on the local resolver.

As one example, in the case of unbound(8), the normal approach is to use a ‘stub-zone’ directive such as:

```
stub-zone:
    name: "0.0.0.0.0.0.0.0.e.d.2.d.f.ip6.arpa."
    stub-host: autoreverse.example.net.
    stub-prime: yes
```

Which directs **unbound** to resolve all addresses within the **ULA CIDR** of fd2d:e000::/48 by querying ‘autoreverse.example.net’.

The **--local-reverse** option can be specified multiple times.

--log-major [=false]

Log major events to Stdout. Major events are rare events which are something you normally want to know about. The default is **true**.

Most major events are start-up related, although there are some on-going major events such as periodic statistics report. There is no good reason to set **--log-major** to false unless you absolute cannot tolerate *any* logging information at all.

--log-minor [=true]

Log minor events to Stdout. Minor events are an elaboration of major events logged by **--log-major** which provide additional insights behind the event. Generally minor event logging is useful when you're trying to diagnose an unexpected major event. Setting **--log-minor** implies setting **--log-major**. The default is **false**.

-log-debug [=true]

Log extensive diagnostic material - mostly discovery related. Most likely of use to developers or sysadmins who are prepared to correlate log details with source code to evaluate the behaviour of **autoreverse**. Setting **--log-debug** implies setting **--log-major** and **--log-minor**.

--log-queries

Write a one line summary of each query to Stdout. The output is intended to be amenable to programmatic post-processing and statistics gathering, but still somewhat human-friendly. On busy systems this option should probably be set to **false** unless you wish to generate voluminous log files. This setting can be toggled at run-time with **SIGUSR2** if you wish to gather a snapshot of activity. The default is **true**.

--max-answers *Integer*

Maximum **PTRs** to allow in a response. This further limits response sizes below the maximum allowed by the query and system defaults.

This limit only applies to potential multiple **PTRs** extracted from **--PTR-deduce** zones. Regardless of this setting, responses are **always** limited to the maximum size allowed by the query including any EDNS0 values. If set to zero, all available **PTRs** are placed in the response within size limits.

The default is **5**.

--passthru *auth-server*

Proxy out-of-bailiwick queries to the *auth-server*.

THIS IS AN EXPERIMENTAL FEATURE - USE WITH CAUTION.

Normally out-of-bailiwick queries generated a **REFUSED** DNS response. However, with this option set, out-of-bailiwick queries are proxied unmodified to the *auth-server* using the same network type the query came in, i.e. **UDP** or **TCP**. Any response from the *auth-server* is similarly proxied unmodified back to the querying client. No retries are attempted, nor are truncated UDP responses re-queried in **TCP**. In effect, **autoreverse** acts as a transparent DNS proxy.

This option is most likely of use in NAT/port-forwarding scenarios where a local authority server is already running on port 53 on a single routable **IPv4** address.

Be aware that DNS Cookies returned by the *auth-server* will not match those sent by **autoreverse** which means clients will see **two** DNS Cookies from the same server IP address. Since clients only retain the most recent DNS Cookie they are likely to send back the wrong one when sending queries which are sometimes answered by **autoreverse** and other times answered by the *auth-server*. There is no impact when **autoreverse** received bad server cookies (at this stage), but there may be some if the *auth-server* de-prioritizes bad server cookies.

--report *time.Duration*

Interval between printing statistics reports expressed in ‘go’ **time.Duration** syntax. The minimum value is 1s and the default is **1h**.

--reverse *CIDR*

Defines the starting point within the reverse zone to discover and serve.

autoreverse ascends the reverse DNS tree from the starting point to discover the zone delegated to **autoreverse** as determined by DNS Probing. If the zone cannot be verified by probing, **autoreverse** exits.

autoreverse processes **--reverse** *after* **--forward** which means reverse delegations can refer to in-bailiwick forward name servers and **autoreverse** correctly responds to SOA related queries as part of the reverse discovery.

autoreverse synthesizes zone information from the discovered delegation details.

The **--reverse** option can be specified multiple times.

--synthesize [=false]

Synthesize missing **PTRs**.

If a **PTR** query cannot be satisfied from **-PTR-deduce** zones, a synthetic response is generated based on the domain name of the forward zone. If set false **NXDomain** is returned for missing **PTRs**. The default is **true**.

--user *user-name*

Reduce privileges by issuing a setuid(2) after **--listen** sockets have been established. In conjunction with **--group** and **--chroot**, this option removes root privileges and restricts access to other system components. **autoreverse** must start as root for this option to succeed.

SIGNALS

autoreverse responds to the following signals:

SIGHUP	Reload all zones specified with **--PTR-deduce**
SIGQUIT	Produce a stack dump and exit
SIGINT	Initiate shutdown
SIGTERM	Initiate shutdown
SIGUSR1	Generates an immediate statistics report
SIGUSR2	Toggles **--log-queries**

GETTING STARTED

Since **autoreverse** relies on the forward and reverse delegation details to deduce its own zone information, the first step is to add those delegation details into the global DNS. Here is an example of the recommended snippet for your forward zone:

```
$ORIGIN yourdomain.
;;
autoreverse IN NS   autoreverse
            IN AAAA 2001:db8:aa:bb::53
            IN A    192.0.2.53
;;
```

Reverse delegation is typically managed by your ISP or address assignment provider so normally you arrange with them to configure the reverse name server as: ‘autoreverse.yourdomain’ to match the ‘NS’ entry in the above snippet.

That completes the setup for **autoreverse**. It is now ready to run!

INVOCATION

With forward and reverse delegations in place, the simplest invocation is to run **autoreverse** with a single **--forward** and **--reverse** option:

```
# autoreverse --forward autoreverse.yourdomain --reverse 2001:db8:23::/64
```

With that information **autoreverse** walks and probes the global DNS to glean delegation details to create its **Zones of Authority** to serve.

IMPLEMENTATION NOTES

autoreverse starts at one label up from the **-forward** and **-reverse** zones and directly queries the parent name servers for delegation details of the specified zone to populate its ‘Zones of Authority’. **autoreverse** continues ‘walking’ up the DNS until it finds responding parents or reaches the upper reaches of the DNS. This ‘walking’ process is important because there are (uncommonly) gaps between child and parent zones in the forward direction, while such gaps are very common in the reverse direction. ‘Walking’ skips over those gaps to discover the delegation material.

Once the parents are discovered, **autoreverse** directly queries them for name servers of the delegated **--forward** and **--reverse** zones. These purported delegated name servers are *DNS Probed* to determine if any of them refer back to the **autoreverse** instance. If at least one does, **autoreverse** accepts the domain as a ‘Zone of Authority’ which it will server answers for.

This is a convoluted way of saying that **autoreverse** determines if it is one of the delegated name servers. You might think that **autoreverse** could simply compare interface addresses against the delegation details and accept a match as ‘proof’, but that doesn’t work in a proxy or port forwarding or NAT environment. Thus **autoreverse** relies on the stronger proof of a *DNS Probe*.

PTR AND FORWARD SYNTHESIS

autoreverse answers **PTR** queries for in-bailiwick zones with synthetic and matching forward names. For example a **PTR** query might produce the following response:

```
    f.7.1.f.0.d. ... 8.b.d.0.1.0.0.2.ip6.arpa. 60
    IN PTR 2001-db8-0-0-0-0-d0-d17f.autoreverse.yourdomain.
or
    54.2.0.192.in-addr.arpa. 60 IN PTR 192-0-2-54.autoreverse.yourdomain.
```

and **autoreverse** answers forward queries for these synthetic **PTR** values with matching address records, i.e.:

```
    2001-db8-0-0-0-0-d0-d17f.autoreverse.yourdomain. 60 IN AAAA 2001:db8::d0:d17f
and
    192-0-2-54.autoreverse.yourdomain. 60 IN A 192.0.2.54
```

This automatic forward and reverse matching is perhaps the main reason for deploying **autoreverse** as it helps meet the requirements of many logging and checking systems which insist on matching entries; sshd(8) and dovecot(1) IMAP and POP3 servers being prominent examples.

INTERMIXING

A common scenario is where you want to intermix configured names with synthetic names in **PTR** responses. This is the purpose of **-PTR-deduce**. **autoreverse** loads the nominated zones and deduces **PTR RRs** for every **A**, **AAAA** and **CNAME** resource found. It also directly loads any **PTR RRs** in the zone. These deduced and direct **PTRs** have preference over synthetic **PTRs**. For example, if you supply a forward zone which contains:

```
     $ORIGIN otherdomain.
```

```
router IN AAAA 2001:db8::1
s1     IN AAAA 2001:db8::2
mail   IN AAAA 2001:db8::5
```

autoreverse replies to the following **PTR** queries with:

```
1.0.0. ... 8.b.d.0.1.0.0.2.ip6.arpa. 60 IN PTR router.otherdomain.
2.0.0. ... 8.b.d.0.1.0.0.2.ip6.arpa. 60 IN PTR s1.otherdomain.
3.0.0. ... 8.b.d.0.1.0.0.2.ip6.arpa. 60 IN PTR 2001-db8-0-0-0-0-0-3.autoreverse.yourdomain.
4.0.0. ... 8.b.d.0.1.0.0.2.ip6.arpa. 60 IN PTR 2001-db8-0-0-0-0-0-4.autoreverse.yourdomain.
5.0.0. ... 8.b.d.0.1.0.0.2.ip6.arpa. 60 IN PTR mail.otherdomain.
6.0.0. ... 8.b.d.0.1.0.0.2.ip6.arpa. 60 IN PTR 2001-db8-0-0-0-0-0-6.autoreverse.yourdomain.
...
```

EXAMPLES

Few use-cases require such a complicated invocation, but this example demonstrates less common features:

```
# autoreverse --forward autoreverse.mydomain --reverse 2001:db8::/64
          --log-query=false --NSID Host:‘hostname‘ --CHAOS=false
          --user nobody --group nobody --chroot /tmp
          --listen 192.0.2.1 --listen [2001:db8::1]:53
          --passthru 127.0.0.1
          --PTR-deduce file:///etc/nsd/data/example.net.zone
          --PTR-deduce file:///etc/nsd/data/8.b.d.0.1.0.0.2.ip6.arpa
          --PTR-deduce axfr://a.ns.example.org/example.net
```

which causes **autoreverse** to listen on multiple addresses, deduces reverse **PTR** names from multiple zone URLs and relinquishes root permissions to run as a less-privileged daemon.

This invocation also supplies the hostname in response to any query containing the NSID option. Finally, out-of-bailiwick queries are passed thru to a name server presumed to be listening on 127.0.0.1 which allows **autoreverse** to proxy or answer all inbound queries.

PERFORMANCE

While **autoreverse** is not particularly designed with performance in mind, it is interesting to note the performance and concurrency impact of a **go [golang]** implementation. A number of tests were run with dnsperf(1) **[dns-oarc]** to inject 4,000,000 random queries while simulating 20 concurrent clients. This table shows the average results across multiple runs:

Platform	**OS**	**Queries/s**	**Memory**
Pi4 (arm64)	FreeBSD 13.0	18,112	17MB
i5-6260U (x64)	Debian 5.10	82,211	18MB

Since dnsperf(1) was run on the *same* system, exchanging queries via loopback, these results should be viewed as indicative rather than definitive. **autoreverse** was run with **--log-queries=false** during these tests but it was otherwise a normal invocation.

SEE ALSO

autoreverse	Project Home Page	https://github.com/markdingo/autoreverse
[dns-oarc]	dnsperf	https://www.dns-oarc.net/tools/dnsperf
[golang]	The go language	https://go.dev
[RFC1918]	ipv4 Private Addresses	https://datatracker.ietf.org/doc/html/rfc1918
[RFC4193]	ipv6 Private Addresses	https://datatracker.ietf.org/doc/html/rfc4193
[RFC7873]	DNS Cookies	https://datatracker.ietf.org/doc/html/rfc7873
[RFC8501]	Reverse DNS in IPv6	https://datatracker.ietf.org/doc/html/rfc8501#section-2.5

FUTURE

--cache-directory *path*

Relying on access to the global DNS during start-up may cause difficulties in some environments (though if connectivity is a problem, inbound queries are unlikely to arrive anyway...).

In such cases it may be sensible for **autoreverse** to cache delegation material and **--PTR-deduce** zones to use as a fallback during start-up if current information is inaccessible. Cache information could be refreshed when fetched and deleted if the source authoritatively says it no longer exists.

HISTORY

First released in late 2021, **autoreverse** development was triggered by a local ISP (ABB) offering free *static* /48 **ipv6** allocations. Importantly, they also supported free reverse delegation to home and small business accounts which is where **autoreverse** comes in handy.

AUTHORS

autoreverse and this manual page were written by Mark Delany.

BUGS

autoreverse has no clue about **DNSSEC**.

As always, any bugs or feedback should be directed to the project page at https://github.com/markdingo/autoreverse