dnsrecord — Typeset DNS Records in LaTeX

Version 1.0.0

A comprehensive package for DNS documentation

Vahid Shaik[*]

March 9, 2026

Abstract

The `dnsrecord` package provides a comprehensive set of LaTeX macros and environments for typesetting DNS (Domain Name System) records, zone files, propagation status tables, DNSSEC chain-of-trust diagrams, email authentication summaries, health check reports, latency comparisons, and DNS provider tables. It is designed for network engineers, system administrators, security researchers, and computer science academics who need to include DNS data in papers, theses, technical reports, and operational documentation. The package supports 22 DNS record types with color-coded badges, configurable display options, and both inline and tabular output formats.

Contents

1 Introduction **2**

2 Installation **2**

2.1 Quick install 2

2.2 Dependencies 2

3 Package Options **2**

4 Inline Record Commands **3**

4.1 Generic command 3

4.2 Standard record types 3

4.3 DNSSEC record types 4

4.4 Modern record types 4

4.5 Email authentication records 4

4.6 Command reference 4

5 Zone File Tables **5**

6 Propagation Status Tables **6**

7 DNSSEC Chain of Trust **7**

8 DNS Health Check Reports **7**

9 Email Authentication Summary **8**

---

[*] Homepage: `https://dnsrobot.net` — Repository: `https://github.com/dnsrobot/latex-dnsrecord`

10 DNS Latency Comparison 8

11 DNS Provider Comparison 9

12 TTL Visualization 9

13 Utility Commands 9
  13.1 Domain and IP formatting 9
  13.2 TTL with human-readable conversion 9
  13.3 DNS response codes 10
  13.4 DNS flags 10
  13.5 DNS query/response header 10

14 Real-World Examples 10
  14.1 Example 1: Documenting a DNS migration 10
  14.2 Example 2: Security audit report 10
  14.3 Example 3: Email deliverability investigation 11

15 Compatibility 11

16 Known Limitations 11

17 Changelog 11

18 License 11

19 Links and Contact 11

1 Introduction

The Domain Name System is one of the most critical components of internet infrastructure, translating human-readable domain names to IP addresses and providing essential services like email routing, service discovery, and security validation. Despite its importance, there has been no standard LaTeX package for consistently typesetting DNS data in technical documents.

Authors of networking papers, system administration guides, and security audit reports typically resort to ad-hoc `verbatim` environments, manual table formatting, or inconsistent text markup when presenting DNS records. This leads to:

- Inconsistent visual presentation across documents and authors
- Difficulty distinguishing between record types at a glance
- No semantic markup for DNS-specific data
- Time wasted on formatting instead of content

The `dnsrecord` package addresses these issues by providing:

- **22 record type commands** with color-coded badges (A, AAAA, CNAME, MX, NS, TXT, SOA, PTR, SRV, CAA, DNSKEY, DS, RRSIG, NSEC, NSEC3, TLSA, HTTPS, SVCB, SPF, DKIM, DMARC)
- **7 environments** for zone tables, propagation checks, DNSSEC chains, health reports, email auth, latency, and provider comparisons
- **Utility commands** for domains, IPs, TTLs, RCODEs, flags, and headers
- **6 package options** for customizing output appearance

2 Installation

2.1 Quick install

Copy `dnsrecord.sty` to the same directory as your document, or to your local `texmf` tree:

```
cp dnsrecord.sty ~/texmf/tex/latex/dnsrecord/
texhash ~/texmf
```

2.2 Dependencies

The package requires the following (all included in TeX Live and MiKTeX):

- `booktabs`
- `xcolor`
- `xparse`
- `etoolbox`
- `amsmath`

3 Package Options

```
\usepackage[color,monospace,ttl]{dnsrecord}  % defaults
```

`color / nocolor`	Enable or disable color-coded record type badges. Default: `color`. Use `nocolor` for black-and-white printing.
`monospace / nomonospace`	Render DNS names, values, and IPs in monospace (`typewriter`) font. Default: `monospace`.
`ttl / nottl`	Show or hide TTL (Time To Live) values in records and tables. Default: `ttl`.

`class / noclass`	Show or hide DNS class (IN) in records and zone tables. Default: `noclass`.
`compact`	Use compact table layout. Default: off.
`boxed`	Render inline records in bordered boxes. Default: off.

4 Inline Record Commands

4.1 Generic command

The base command for any DNS record type:

```
\dnsrecord{type}{name}{value}[ttl]
```

The `ttl` parameter is optional and controlled by the `ttl/nottl` package option.

Example:

```
\dnsrecord{A}{example.com}{93.184.216.34}[300]
```

Result: **`A`** `example.com` → `93.184.216.34` TTL 300

4.2 Standard record types

Address records:

`A` `example.com` → `93.184.216.34` TTL 300

`AAAA` `example.com` → `2606:2800:220:1::1946` TTL 300

Name resolution:

`CNAME` `www.example.com` → `example.com` TTL 3600

`NS` `example.com` → `ns1.example.com` TTL 86400

`PTR` `34.216.184.93.in-addr.arpa` → `example.com` TTL 3600

Mail and services:

`MX` `example.com` → `mail.example.com (pri 10)` TTL 3600

`SRV` `_sip._tcp.example.com` → `sip.example.com:5060 (pri 10)` TTL 3600

Text and policy:

`TXT` `example.com` → `"v=spf1 mx -all"` TTL 300

`CAA` `example.com` → `0 issue letsencrypt.org` TTL 3600

Authority:

`SOA` `example.com` → `ns1.example.com (admin.example.com)` TTL 86400

4.3 DNSSEC record types

DNSKEY example.com → 257 3 13 (base64key...) TTL 86400

DS example.com → 12345 13 2 (digest...) TTL 86400

RRSIG example.com → A 13 2 300 (sig...) TTL 300

NSEC example.com → www.example.com A AAAA RRSIG TTL 300

NSEC 3 → example.com1 0 10 ABCDEF (hash...)[300]

4.4 Modern record types

TLSA _443._tcp.example.com → 3 1 1 (hash...) TTL 3600

HTTPS example.com → 1 . alpn=h2,h3 ipv4hint=93.184.216.34 TTL 300

SVCB example.com → 1 . alpn=h2 TTL 300

4.5 Email authentication records

SPF example.com → v=spf1 include:mail.example.com -all TTL 300

DKIM s1._domainkey.example.com → v=DKIM1; k=rsa; p=MIGf... TTL 3600

DMARC _dmarc.example.com → v=DMARC1; p=reject; rua=mailto:d@example.com TTL 3600

4.6 Command reference

```
% Standard records
\dnsA{name}{ip}[ttl]
\dnsAAAA{name}{ipv6}[ttl]
\dnsCNAME{alias}{canonical}[ttl]
\dnsMX{name}{priority}{mailserver}[ttl]
\dnsNS{name}{nameserver}[ttl]
\dnsTXT{name}{text}[ttl]
\dnsSOA{name}{primary-ns}{admin-email}[ttl]
\dnsPTR{reverse-ip}{hostname}[ttl]
\dnsSRV{name}{priority}{target}{port}[ttl]
\dnsCAA{name}{flags-tag}{value}[ttl]

% DNSSEC records
\dnsDNSKEY{name}{flags-proto-algo-key}[ttl]
\dnsDS{name}{keytag-algo-digest-type-digest}[ttl]
\dnsRRSIG{name}{type-algo-labels-ttl-sig}[ttl]
\dnsNSEC{name}{next-domain-types}[ttl]
\dnsNSEC3{name}{algo-flags-iter-salt-hash}[ttl]

% Modern records
\dnsTLSA{name}{usage-selector-matching-data}[ttl]
\dnsHTTPS{name}{priority-target-params}[ttl]
\dnsSVCB{name}{priority-target-params}[ttl]

% Email authentication
\dnsSPF{name}{policy}[ttl]
\dnsDKIM{selector._domainkey.name}{policy}[ttl]
\dnsDMARC{_dmarc.name}{policy}[ttl]
```

5 Zone File Tables

The `dnszone` environment creates a formatted table for an entire DNS zone, automatically counting records.

```
\begin{dnszone}{example.com}
  \dnsentry{A}{@}{93.184.216.34}[300]
  \dnsentry{AAAA}{@}{2606:2800:220:1::1946}[300]
  \dnsentry{CNAME}{www}{example.com}[3600]
  \dnsentry{MX}{@}{mail.example.com (pri 10)}[3600]
  \dnsentry{NS}{@}{ns1.example.com}[86400]
  \dnsentry{NS}{@}{ns2.example.com}[86400]
  \dnsentry{TXT}{@}{v=spf1 mx -all}[300]
  \dnsentry{CAA}{@}{0 issue letsencrypt.org}[3600]
\end{dnszone}
```

Type TTL	**Name**	**Value**	**Class**
`A` 300	@	`93.184.216.34`	IN
`AAAA` 300	@	`2606:2800:220:1::1946`	IN
`CNAME` 3600	www	`example.com`	IN
`MX` 3600	@	`mail.example.com (pri 10)`	IN
`NS` 86400	@	`ns1.example.com`	IN
`NS` 86400	@	`ns2.example.com`	IN
`TXT` 300	@	`v=spf1 mx -all`	IN
`CAA` 3600	@	`0 issue letsencrypt.org`	IN

DNS zone: `example.com` — 8 records

A more complex example with DNSSEC records:

Type	Name	Value	Class	TTL
`A`	@	`198.51.100.1`	IN	300
`AAAA`	@	`2001:db8::1`	IN	300
`NS`	@	`ns1.secure.example.com`	IN	86400
`NS`	@	`ns2.secure.example.com`	IN	86400
`DNSKEY`	@	`257 3 13 (KSK base64...)`	IN	86400
`DNSKEY`	@	`256 3 13 (ZSK base64...)`	IN	86400
`DS`	@	`12345 13 2 (digest...)`	IN	86400
`RRSIG`	@	`A 13 3 300 (signature...)`	IN	300
`NSEC`	@	`www.secure.example.com A AAAA NS`	IN	300

DNS zone: `secure.example.com` *— 9 records*

6 Propagation Status Tables

The `dnspropagation` environment displays DNS record propagation across multiple global resolver locations. This is useful for documenting DNS migration events, TTL behavior analysis, or troubleshooting reports.

```
\begin{dnspropagation}{example.com}{A}
  \dnsserver{Google}{8.8.8.8}{93.184.216.34}{5ms}{propagated}
  \dnsserver{Cloudflare}{1.1.1.1}{93.184.216.34}{3ms}{propagated}
  \dnsserver{Quad9}{9.9.9.9}{93.184.216.34}{8ms}{propagated}
  \dnsserver{OpenDNS}{208.67.222.222}{93.184.216.34}{12ms}{propagated}
  \dnsserver{ISP Germany}{194.25.0.60}{93.184.216.33}{45ms}{pending}
  \dnsserver{ISP France}{80.10.246.2}{---}{---}{failed}
  \dnsserver{ISP Japan}{210.171.224.1}{---}{120ms}{timeout}
\end{dnspropagation}
```

Server	IP	Result	RTT	Status
Google	8.8.8.8	93.184.216.34	5ms	● Propagated
Cloudflare	1.1.1.1	93.184.216.34	3ms	● Propagated
Quad9	9.9.9.9	93.184.216.34	8ms	● Propagated
OpenDNS	208.67.222.222	93.184.216.34	12ms	● Propagated
ISP Germany	194.25.0.60	93.184.216.33	45ms	● Pending
ISP France	80.10.246.2	--	--	● Failed
ISP Japan	210.171.224.1	--	120ms	● Timeout

Propagation check: `A` *record for* `example.com`

The status indicators use color-coded bullets: ● Propagated, ● Pending, ● Failed, ● Timeout.

7 DNSSEC Chain of Trust

The `dnssecchain` environment visualizes the DNSSEC delegation chain from root to target domain:

```
\begin{dnssecchain}{example.com}
  \dnssecentry{. (root)}{DNSKEY}{20326}{RSASHA256}
  \dnssecentry{. (root)}{DS}{20326}{RSASHA256}
  \dnssecentry{com.}{DNSKEY}{30909}{ECDSAP256SHA256}
  \dnssecentry{com.}{DS}{30909}{ECDSAP256SHA256}
  \dnssecentry{example.com.}{DNSKEY}{12345}{ECDSAP256SHA256}
  \dnssecentry{example.com.}{RRSIG}{12345}{ECDSAP256SHA256}
\end{dnssecchain}
```

Zone	**Record**	**Key Tag**	**Algorithm**
`. (root)`	`DNSKEY`	`20326`	`RSASHA256`
`. (root)`	`DS`	`20326`	`RSASHA256`
`com.`	`DNSKEY`	`30909`	`ECDSAP256SHA256`
`com.`	`DS`	`30909`	`ECDSAP256SHA256`
`example.com.`	`DNSKEY`	`12345`	`ECDSAP256SHA256`
`example.com.`	`RRSIG`	`12345`	`ECDSAP256SHA256`

DNSSEC chain of trust for `example.com`

DNSSEC validation status indicators:

DNSSEC: Validated **DNSSEC:** Validation Failed **DNSSEC:** Not Signed **DNSSEC:** Insecure Delegation

8 DNS Health Check Reports

The `dnshealthcheck` environment creates a diagnostic report format commonly used in DNS audits:

```
\begin{dnshealthcheck}{example.com}
  \dnshealthitem{healthy}{All nameservers responding}
  \dnshealthitem{healthy}{SOA serial numbers consistent}
  \dnshealthitem{healthy}{MX records resolve to valid IPs}
  \dnshealthitem{warning}{TTL for A record is below 300s}
  \dnshealthitem{warning}{No IPv6 (AAAA) record found}
  \dnshealthitem{critical}{DNSSEC signatures expired}
  \dnshealthitem{critical}{Open resolver detected on ns2}
  \dnshealthitem{info}{Domain registered until 2028-01-15}
\end{dnshealthcheck}
```

DNS Health Check: `example.com`

- **PASS:** All nameservers responding
- **PASS:** SOA serial numbers consistent across all NS
- **PASS:** MX records resolve to valid IP addresses

- **WARN:** TTL for A record is below recommended 300s minimum
- **WARN:** No IPv6 (AAAA) record found for root domain
- **FAIL:** DNSSEC RRSIG signatures have expired
- **FAIL:** Open resolver detected on ns2.example.com
- **INFO:** Domain registration expires 2028-01-15

9 Email Authentication Summary

The `dnsemailauth` environment displays SPF, DKIM, and DMARC configuration status for a domain:

```
\begin{dnsemailauth}{example.com}
  \dnsemailentry{SPF}{v=spf1 mx -all}{Strict}{propagated}
  \dnsemailentry{DKIM}{v=DKIM1; k=rsa; p=MIGf...}{2048-bit RSA}{propagated}
  \dnsemailentry{DMARC}{v=DMARC1; p=reject}{Reject}{propagated}
  \dnsemailentry{MX}{mail.example.com (pri 10)}{Primary}{propagated}
  \dnsemailentry{TLSA}{3 1 1 (cert hash...)}{DANE}{pending}
\end{dnsemailauth}
```

Protocol	Record	Policy	Status
`SPF`	`v=spf1 mx -all`	`Strict`	• Propagated
`DKIM`	`v=DKIM1; k=rsa; p=MIGf...`	`2048-bit RSA`	• Propagated
`DMARC`	`v=DMARC1; p=reject`	`Reject`	• Propagated
`MX`	`mail.example.com (pri 10)`	`Primary`	• Propagated
`TLSA`	`3 1 1 (cert hash...)`	`DANE`	• Pending

Email authentication for `example.com`

10 DNS Latency Comparison

The `dnslatency` environment compares response times across DNS resolvers:

```
\begin{dnslatency}{example.com}
  \dnslatencyentry{Cloudflare}{1.1.1.1}{2}{4}{8}
  \dnslatencyentry{Google}{8.8.8.8}{4}{7}{15}
  \dnslatencyentry{Quad9}{9.9.9.9}{5}{9}{18}
  \dnslatencyentry{OpenDNS}{208.67.222.222}{8}{14}{25}
  \dnslatencyentry{ISP Default}{192.168.1.1}{12}{35}{120}
\end{dnslatency}
```

Resolver	IP	Min (ms)	Avg (ms)	Max (ms)
`Cloudflare`	`1.1.1.1`	2	4	8
`Google`	`8.8.8.8`	4	7	15
`Quad9`	`9.9.9.9`	5	9	18
`OpenDNS`	`208.67.222.222`	8	14	25
`ISP Default`	`192.168.1.1`	12	35	120

DNS latency for `example.com`

11 DNS Provider Comparison

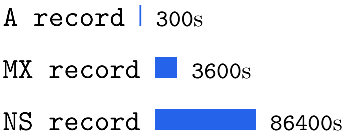
```
\begin{dnsproviders}
  \dnsprovider{Cloudflare}{1.1.1.1}{1.0.0.1}{Yes}{DoH + DoT}
  \dnsprovider{Google}{8.8.8.8}{8.8.4.4}{Yes}{DoH + DoT}
  \dnsprovider{Quad9}{9.9.9.9}{149.112.112.112}{Yes}{DoH + DoT}
  \dnsprovider{OpenDNS}{208.67.222.222}{208.67.220.220}{No}{DoH}
  \dnsprovider{AdGuard}{94.140.14.14}{94.140.15.15}{Yes}{DoH + DoT}
\end{dnsproviders}
```

Provider	**Primary**	**Secondary**	**DNSSEC**	**DoH/DoT**
Cloudflare	`1.1.1.1`	`1.0.0.1`	Yes	DoH + DoT
Google	`8.8.8.8`	`8.8.4.4`	Yes	DoH + DoT
Quad9	`9.9.9.9`	`149.112.112.112`	Yes	DoH + DoT
OpenDNS	`208.67.222.222`	`208.67.220.220`	No	DoH
AdGuard	`94.140.14.14`	`94.140.15.15`	Yes	DoH + DoT

12 TTL Visualization

The `\dnsttlbar` command creates a simple horizontal bar visualization for comparing TTL values:

```
\dnsttlbar{A record}{300}{86400}
\dnsttlbar{MX record}{3600}{86400}
\dnsttlbar{NS record}{86400}{86400}
```



13 Utility Commands

13.1 Domain and IP formatting

```
\dnsdomain{example.com}      % formatted domain name
\dnsip{93.184.216.34}         % formatted IP address
```

Result: `example.com` `93.184.216.34`

13.2 TTL with human-readable conversion

```
\dnsttl{300}     \dnsttl{3600}     \dnsttl{86400}
```

Result: `300` (5m) `3600` (1h) `86400` (1d)

13.3 DNS response codes

```
\dnsrcode{NOERROR}  \dnsrcode{NXDOMAIN}  \dnsrcode{SERVFAIL}  \dnsrcode{REFUSED}
```

Result: `NOERROR` `NXDOMAIN` `SERVFAIL` `REFUSED`

13.4 DNS flags

```
\dnsflag{QR} \dnsflag{AA} \dnsflag{RD} \dnsflag{RA} \dnsflag{AD} \dnsflag{CD}
```

Result: `QR` `AA` `RD` `RA` `AD` `CD`

13.5 DNS query/response header

```
\dnsheader{12345}{QUERY}{NOERROR}{QR RD RA AD}
```

```
;; HEADER: id=12345, opcode=QUERY, rcode=NOERROR, flags=QR RD RA AD
```

14 Real-World Examples

14.1 Example 1: Documenting a DNS migration

When migrating `company.com` from Provider A to Provider B, the following propagation results were observed 30 minutes after the NS record change (3600 (1h) TTL):

Server	IP	Result	RTT	Status
`Google`	`8.8.8.8`	`ns1.providerb.com`	5ms	• Propagated
`Cloudflare`	`1.1.1.1`	`ns1.providerb.com`	3ms	• Propagated
`Quad9`	`9.9.9.9`	`ns1.providera.com`	8ms	• Pending
`ISP Germany`	`194.25.0.60`	`ns1.providera.com`	45ms	• Pending
`ISP Brazil`	`200.221.11.101`	`ns1.providera.com`	180ms	• Pending

Propagation check: `NS` record for `company.com`

14.2 Example 2: Security audit report

DNS security assessment for `bank.example.com`:

DNS Health Check: `bank.example.com`

- **PASS:** DNSSEC fully deployed with ECDSAP256SHA256
- **PASS:** CAA record restricts issuance to approved CAs
- **PASS:** SPF, DKIM, and DMARC all configured with strict policies
- **PASS:** DANE/TLSA record published for SMTP
- **WARN:** SOA refresh interval (3600s) below recommended 7200s
- **FAIL:** Zone transfer (AXFR) permitted from any source
- **INFO:** 4 nameservers across 2 autonomous systems

14.3 Example 3: Email deliverability investigation

Email authentication configuration for `newsletter.example.com`:

Protocol	Record	Policy	Status
`SPF`	`v=spf1 include:sendgrid.net -all`	`Strict (-all)`	• Propagated
`DKIM`	`v=DKIM1; k=rsa; p=MIGf...`	`2048-bit RSA`	• Propagated
`DMARC`	`v=DMARC1; p=quarantine; pct=100`	`Quarantine`	• Propagated
`MX`	`mx.sendgrid.net (pri 10)`	`SendGrid`	• Propagated
`PTR`	`mail.newsletter.example.com`	`Reverse DNS`	• Failed

Email authentication for `newsletter.example.com`

The missing PTR record (`NXDOMAIN`) for the sending IP is likely causing deliverability issues with strict mail servers.

15 Compatibility

The `dnsrecord` package has been tested with:

- **TeX Live** 2022–2026
- **MiKTeX** 22.1+
- **Overleaf** (online)
- `pdflatex`, `xelatex`, and `lualatex` engines
- `article`, `report`, `book`, and `beamer` classes

16 Known Limitations

- Long TXT record values may overflow table columns; consider using abbreviated values or the `compact` option
- The TTL bar visualization (`\dnsttlbar`) uses fixed-width rendering and may not scale well for very large TTL differences
- Color-coded badges require a PDF-capable output driver

17 Changelog

v1.0.0 (2026-03-09) Initial release. 22 record types, 7 environments, utility commands.

18 License

This work is released under the LaTeX Project Public License v1.3c or later. The full text is available at:
`https://www.latex-project.org/lppl.txt`

19 Links and Contact

Homepage `https://dnsrobot.net`
Repository `https://github.com/dnsrobot/latex-dnsrecord`
Bug reports `https://github.com/dnsrobot/latex-dnsrecord/issues`
CTAN `https://ctan.org/pkg/dnsrecord`
Author Vahid Shaik — `https://dnsrobot.net`