

**NAME**

Net::DNS::Header – DNS packet header

**SYNOPSIS**

```
use Net::DNS;

$packet = new Net::DNS::Packet;
$header = $packet->header;
```

**DESCRIPTION**

Net::DNS::Header represents the header portion of a DNS packet.

**METHODS****\$packet->header**

```
$packet = new Net::DNS::Packet;
$header = $packet->header;
```

Net::DNS::Header objects emanate from the Net::DNS::Packet **header()** method, and contain an opaque reference to the parent Packet object.

Header objects may be assigned to suitably scoped lexical variables. They should never be stored in global variables or persistent data structures.

**string**

```
print $packet->header->string;
```

Returns a string representation of the packet header.

**print**

```
$packet->header->print;
```

Prints the string representation of the packet header.

**id**

```
print "query id = ", $packet->header->id, "\n";
$packet->header->id(1234);
```

Gets or sets the query identification number.

A random value is assigned if the argument value is undefined.

**opcode**

```
print "query opcode = ", $packet->header->opcode, "\n";
$packet->header->opcode("UPDATE");
```

Gets or sets the query opcode (the purpose of the query).

**rcode**

```
print "query response code = ", $packet->header->rcode, "\n";
$packet->header->rcode("SERVFAIL");
```

Gets or sets the query response code (the status of the query).

**qr**

```
print "query response flag = ", $packet->header->qr, "\n";
$packet->header->qr(0);
```

Gets or sets the query response flag.

**aa**

```
print "response is ", $packet->header->aa ? "" : "non-", "authoritative\n";
$packet->header->aa(0);
```

Gets or sets the authoritative answer flag.

**tc**

```
print "packet is ", $packet->header->tc ? "" : "not ", "truncated\n";
$packet->header->tc(0);
```

Gets or sets the truncated packet flag.

**rd**

```
print "recursion was ", $packet->header->rd ? "" : "not ", "desired\n";
$packet->header->rd(0);
```

Gets or sets the recursion desired flag.

**ra**

```
print "recursion is ", $packet->header->ra ? "" : "not ", "available\n";
$packet->header->ra(0);
```

Gets or sets the recursion available flag.

**z**

Unassigned bit, should always be zero.

**ad**

```
print "The response has ", $packet->header->ad ? "" : "not", "been verified\n";
```

Relevant in DNSSEC context.

(The AD bit is only set on a response where signatures have been cryptographically verified or the server is authoritative for the data and is allowed to set the bit by policy.)

**cd**

```
print "checking was ", $packet->header->cd ? "not" : "", "desired\n";
$packet->header->cd(0);
```

Gets or sets the checking disabled flag.

**qdcount, zocount**

```
print "# of question records: ", $packet->header->qdcount, "\n";
```

Returns the number of records in the question section of the packet. In dynamic update packets, this field is known as `zocount` and refers to the number of RRs in the zone section.

**ancount, prcount**

```
print "# of answer records: ", $packet->header->ancount, "\n";
```

Returns the number of records in the answer section of the packet which may, in the case of corrupt packets, differ from the actual number of records. In dynamic update packets, this field is known as `prcount` and refers to the number of RRs in the prerequisite section.

**nscount, upcount**

```
print "# of authority records: ", $packet->header->nscount, "\n";
```

Returns the number of records in the authority section of the packet which may, in the case of corrupt packets, differ from the actual number of records. In dynamic update packets, this field is known as `upcount` and refers to the number of RRs in the update section.

**arcount, adcount**

```
print "# of additional records: ", $packet->header->arcount, "\n";
```

Returns the number of records in the additional section of the packet which may, in the case of corrupt packets, differ from the actual number of records. In dynamic update packets, this field is known as `adcount`.

**EDNS Protocol Extensions****do**

```
print "DNSSEC_OK flag was ", $packet->header->do ? "not" : "", "set\n";
$packet->header->do(1);
```

Gets or sets the EDNS DNSSEC OK flag.

### Extended rcode

EDNS extended rcodes are handled transparently by `$packet->header->rcode()`.

### UDP packet size

```
$udp_max = $packet->header->size;
```

```
$udp_max = $packet->edns->size;
```

EDNS offers a mechanism to advertise the maximum UDP packet size which can be assembled by the local network stack.

UDP size advertisement can be viewed as either a header extension or an EDNS feature. Endless debate is avoided by supporting both views.

### edns

```
$header = $packet->header;
```

```
$version = $header->edns->version;
```

```
@options = $header->edns->options;
```

```
$option = $header->edns->option(n);
```

```
$udp_max = $packet->edns->size;
```

Auxiliary function which provides access to the EDNS protocol extension OPT RR.

## COPYRIGHT

Copyright (c)1997 Michael Fuhr.

Portions Copyright (c)2002,2003 Chris Reinhardt.

Portions Copyright (c)2012 Dick Franks.

All rights reserved.

## LICENSE

Permission to use, copy, modify, and distribute this software and its documentation for any purpose and without fee is hereby granted, provided that the above copyright notice appear in all copies and that both that copyright notice and this permission notice appear in supporting documentation, and that the name of the author not be used in advertising or publicity pertaining to distribution of the software without specific prior written permission.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

## SEE ALSO

perl, Net::DNS, Net::DNS::Packet, Net::DNS::RR::OPT RFC 1035 Section 4.1.1