NAME

Net::DNS::RR – DNS resource record base class

SYNOPSIS

```
use Net::DNS;

$rr = new Net::DNS::RR('example.com IN AAAA 2001:DB8::1');

$rr = new Net::DNS::RR(
        owner   => 'example.com',
        type    => 'AAAA',
        address => '2001:DB8::1'
        );
```

DESCRIPTION

Net::DNS::RR is the base class for DNS Resource Record (RR) objects. See also the manual pages for each specific RR type.

METHODS

WARNING!!! Do not assume the RR objects you receive from a query are of a particular type. You must always check the object type before calling any of its methods. If you call an unknown method, you will get an error message and execution will be terminated.

new (from string)

```
$aaaa  = new Net::DNS::RR('host.example.com. 86400 AAAA 2001:DB8::1');
$mx    = new Net::DNS::RR('example.com. 7200 MX 10 mailhost.example.com.');
$cname = new Net::DNS::RR('www.example.com 300 IN CNAME host.example.com');
$txt   = new Net::DNS::RR('txt.example.com 3600 HS TXT "text data"');
```

Returns an object of the appropriate RR type, or a Net::DNS::RR object if the type is not implemented. The attribute values are extracted from the string passed by the user. The syntax of the argument string follows the RFC1035 specification for zone files, and is compatible with the result returned by the string method.

The owner and RR type are required; all other information is optional. Omitting the optional fields is useful for creating the empty RDATA sections required for certain dynamic update operations. See the Net::DNS::Update manual page for additional examples.

All names are interpreted as fully qualified domain names. The trailing dot (.) is optional.

new (from hash)

```
$rr = new Net::DNS::RR(%hash);

$rr = new Net::DNS::RR(
        owner   => 'host.example.com',
        ttl     => 86400,
        class   => 'IN',
        type    => 'AAAA',
        address => '2001:DB8::1'
        );

$rr = new Net::DNS::RR(
        owner   => 'txt.example.com',
        type    => 'TXT',
        txtdata => [ 'one', 'two' ]
        );
```

Returns an object of the appropriate RR type, or a Net::DNS::RR object if the type is not implemented. Consult the relevant manual pages for the usage of type specific attributes.

The owner and RR type are required; all other information is optional. Omitting optional attributes is useful

for creating the empty RDATA sections required for certain dynamic update operations.

decode

```
( $rr, $next ) = decode Net::DNS::RR( \$data, $offset, @opaque );
```

Decodes a DNS resource record at the specified location within a DNS packet.

The argument list consists of a reference to the buffer containing the packet data and offset indicating where resource record begins. Remaining arguments, if any, are passed as opaque data to subordinate decoders.

Returns a `Net::DNS::RR` object and the offset of the next record in the packet.

An exception is raised if the data buffer contains insufficient or corrupt data.

Any remaining arguments are passed as opaque data to subordinate decoders and do not form part of the published interface.

encode

```
$data = $rr->encode( $offset, @opaque );
```

Returns the `Net::DNS::RR` in binary format suitable for inclusion in a DNS packet buffer.

The offset indicates the intended location within the packet data where the `Net::DNS::RR` is to be stored.

Any remaining arguments are opaque data which are passed intact to subordinate encoders.

canonical

```
$data = $rr->canonical;
```

Returns the `Net::DNS::RR` in canonical binary format suitable for DNSSEC signature validation.

The absence of the associative array argument signals to subordinate encoders that the canonical uncompressed lower case form of embedded domain names is to be used.

print

```
$rr->print;
```

Prints the resource record to the currently selected output filehandle. Calls the string method to get the formatted RR representation.

string

```
print $rr->string, "\n";
```

Returns a string representation of the RR using the master file format mandated by RFC1035. All domain names are fully qualified with trailing dot. This differs from RR attribute methods, which omit the trailing dot.

plain

```
$plain = $rr->plain;
```

Returns a simplified single-line representation of the RR. This facilitates interaction with programs like nsupdate which have rudimentary parsers.

token

```
@token = $rr->token;
```

Returns a token list representation of the RR zone file string.

generic

```
$generic = $rr->generic;
```

Returns the generic RR representation defined in RFC3597. This facilitates creation of zone files containing RRs unrecognised by outdated nameservers and provisioning software.

owner name

```
$name = $rr->owner;
```

Returns the owner name of the record.

type

```
$type = $rr->type;
```

Returns the record type.

class

```
$class = $rr->class;
```

Resource record class.

ttl

```
$ttl = $rr->ttl;
$ttl = $rr->ttl(3600);
```

Resource record time to live in seconds.

rdata

```
$rr = new Net::DNS::RR( type => NULL, rdata => 'arbitrary' );
```

Resource record data section when viewed as opaque octets.

rdstring

```
$rdstring = $rr->rdstring;
```

Returns a string representation of the RR-specific data.

rdlength

```
$rdlength = $rr->rdlength;
```

Returns the uncompressed length of the encoded RR-specific data.

Sorting of RR arrays

Sorting of RR arrays is done by **Net::DNS::rrsort()**, see documentation for Net::DNS. This package provides class methods to set the comparator function used for a particular RR based on its attributes.

set_rrsort_func

```
my $function = sub {                ## numerically ascending order
    $Net::DNS::a->{'preference'} <=> $Net::DNS::b->{'preference'};
};

Net::DNS::RR::MX->set_rrsort_func( 'preference', $function );

Net::DNS::RR::MX->set_rrsort_func( 'default_sort', $function );
```

set_rrsort_func() must be called as a class method. The first argument is the attribute name on which the sorting is to take place. If you specify "default_sort" then that is the sort algorithm that will be used when **get_rrsort_func()** is called without an RR attribute as argument.

The second argument is a reference to a comparator function that uses the global variables `$a` and `$b` in the Net::DNS package. During sorting, the variables `$a` and `$b` will contain references to objects of the class whose **set_rrsort_func()** was called. The above sorting function will only be applied to Net::DNS::RR::MX objects.

The above example is the sorting function implemented in MX.

get_rrsort_func

```
$function = Net::DNS::RR::MX->get_rrsort_func('preference');
$function = Net::DNS::RR::MX->get_rrsort_func();
```

get_rrsort_func() returns a reference to the comparator function.

COPYRIGHT

Copyright (c)1997−2001 Michael Fuhr.

Portions Copyright (c)2002,2003 Chris Reinhardt.

Portions Copyright (c)2005−2007 Olaf Kolkman.

Portions Copyright (c)2007,2012 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::Question, Net::DNS::Packet, Net::DNS::Update, RFC1035 Section 4.1.3, RFC1123, RFC3597