NAME

Net::DNS::Update – DNS dynamic update packet

SYNOPSIS

```
use Net::DNS;

$update = new Net::DNS::Update( 'example.com', 'IN' );

$update->push( prereq => nxrrset('foo.example.com. AAAA') );
$update->push( update => rr_add('foo.example.com. 86400 AAAA 2001::DB8::1') );
```

DESCRIPTION

Net::DNS::Update is a subclass of Net::DNS::Packet, to be used for making DNS dynamic updates.

Programmers should refer to RFC2136 for dynamic update semantics.

METHODS

new

```
$update = new Net::DNS::Update;
$update = new Net::DNS::Update( 'example.com' );
$update = new Net::DNS::Update( 'example.com', 'HS' );
```

Returns a Net::DNS::Update object suitable for performing a DNS dynamic update. Specifically, it creates a packet with the header opcode set to UPDATE and the zone record type to SOA (per RFC 2136, Section 2.3).

Programs must use the **push()** method to add RRs to the prerequisite and update sections before performing the update.

Arguments are the zone name and the class. The zone and class may be undefined or omitted and default to the default domain from the resolver configuration and IN respectively.

push

```
$ancount = $update->push( prereq => $rr );
$nscount = $update->push( update => $rr );
$arcount = $update->push( additional => $rr );

$nscount = $update->push( update => $rr1, $rr2, $rr3 );
$nscount = $update->push( update => @rr );
```

Adds RRs to the specified section of the update packet.

Returns the number of resource records in the specified section.

Section names may be abbreviated to the first three characters.

unique_push

```
$ancount = $update->unique_push( prereq => $rr );
$nscount = $update->unique_push( update => $rr );
$arcount = $update->unique_push( additional => $rr );

$nscount = $update->unique_push( update => $rr1, $rr2, $rr3 );
$nscount = $update->unique_push( update => @rr );
```

Adds RRs to the specified section of the update packet provided that the RRs are not already present in the same section.

Returns the number of resource records in the specified section.

Section names may be abbreviated to the first three characters.

EXAMPLES

The first example below shows a complete program. Subsequent examples show only the creation of the update packet.

Although the examples are presented using the string form of RRs, the corresponding (name => value) form may also be used.

Add a new host

```
#!/usr/bin/perl

use Net::DNS;

# Create the update packet.
my $update = new Net::DNS::Update('example.com');

# Prerequisite is that no address records exist for the name.
$update->push( pre => nxrrset('foo.example.com. A') );
$update->push( pre => nxrrset('foo.example.com. AAAA') );

# Add two address records for the name.
$update->push( update => rr_add('foo.example.com. 86400 A 192.0.2.1') );
$update->push( update => rr_add('foo.example.com. 86400 AAAA 2001:DB8::1') );

# Send the update to the zone's primary master.
my $resolver = new Net::DNS::Resolver;
$resolver->nameservers('primary-master.example.com');

my $reply = $resolver->send($update);

# Did it work?
if ($reply) {
        if ( $reply->header->rcode eq 'NOERROR' ) {
                print "Update succeeded\n";
        } else {
                print 'Update failed: ', $reply->header->rcode, "\n";
        }
} else {
        print 'Update failed: ', $resolver->errorstring, "\n";
}
```

Add an MX record for a name that already exists

```
my $update = new Net::DNS::Update('example.com');
$update->push( prereq => yxdomain('example.com') );
$update->push( update => rr_add('example.com MX 10 mailhost.example.com') );
```

Add a TXT record for a name that does not exist

```
my $update = new Net::DNS::Update('example.com');
$update->push( prereq => nxdomain('info.example.com') );
$update->push( update => rr_add('info.example.com TXT "yabba dabba doo"') );
```

Delete all A records for a name

```
my $update = new Net::DNS::Update('example.com');
$update->push( prereq => yxrrset('foo.example.com A') );
$update->push( update => rr_del('foo.example.com A') );
```

Delete all RRs for a name

```
my $update = new Net::DNS::Update('example.com');
$update->push( prereq => yxdomain('byebye.example.com') );
$update->push( update => rr_del('byebye.example.com') );
```

Perform a DNS update signed using a BIND private key file

```
my $update = new Net::DNS::Update('example.com');
$update->push( update => rr_add('foo.example.com AAAA 2001:DB8::1') );
$update->sign_tsig( "$dir/Khmac-sha512.example.com.+165+01018.private" );
my $reply = $resolver->send( $update );
$reply->verify( $update ) || die $reply->verifyerr;
```

Signing the DNS update using a BIND public key file

```
$update->sign_tsig( "$dir/Khmac-sha512.example.com.+165+01018.key" );
```

Signing the DNS update using a customised TSIG record

```
$update->sign_tsig( "$dir/Khmac-sha512.example.com.+165+01018.private",
                    fudge => 60
                    );
```

Another way to sign a DNS update

```
my $key_name = 'tsig-key';
my $key      = 'awwLOtRfpGE+rRKF2+DEiw==';

my $tsig = new Net::DNS::RR("$key_name TSIG $key");
$tsig->fudge(60);

my $update = new Net::DNS::Update('example.com');
$update->push( update     => rr_add('foo.example.com AAAA 2001:DB8::1') );
$update->push( additional => $tsig );
```

COPYRIGHT

LICENSE

SEE ALSO

perl, Net::DNS, Net::DNS::Packet, Net::DNS::Header, Net::DNS::RR, Net::DNS::Resolver, RFC 2136, RFC 2845