Network Working Group
Request for Comments: 2136
Updates: 1035
Category: Standards Track

P. Vixie, Editor
ISC
S. Thomson
Bellcore
Y. Rekhter
Cisco
J. Bound
DEC

April 1997

Dynamic Updates in the Domain Name System (DNS UPDATE)

Динамические обновления в системе доменных имён (DNS UPDATE)

Статус документа

Этот документ задаёт проект стандартного протокола Internet для сообщества Internet и служит приглашением к дискуссии в целях развития и совершенствования протокола. Текущий статус стандартизации протокола можно узнать из документа Internet Official Protocol Standards (STD 1). Распространение документа не ограничивается.

Аннотация

Исходно система доменных имён создавалась для поддержки запросов к статически настроенной базе данных. Хотя изменение данных предусматривалось, частота таких изменений должна быть невысокой и все обновления предполагались в форме внешнего редактирования первичных файлов (Master File) зон.

С помощью заданной здесь операции UPDATE можно добавлять и удалять RR и RRset в заданной зоне. Предварительные условия задаются отдельно от операции и могут включать зависимости от прежнего наличия или отсутствия RRset или наличия отдельной RR.

Операция UPDATE неделима (atomic), т. е. все предварительные условия должны быть соблюдены, иначе операция не выполняется. Когда предварительные условия выполнены, не может возникать зависящих от данных ошибок.

1. Определения

В этом документе намеренно введены дополнительные определения для ролей серверов Master, Slave, Primary Master и их перечислению в NS RR и поле SOA MNAME. В этом смысле приведённые ниже определения типов серверов можно считать дополнением к [RFC1035] и они согласуются с [RFC1996]

Slave - ведомый
Полномочный сервер, использующий AXFR или IXFR для извлечения зоны и указанный в NS RRset зоны.

Master - ведущий
Полномочный сервер, настроенный как источник данных для AXFR/IXFR одного или нескольких ведомых серверов.

Primary Master - основной ведущий
Первичный сервер в корне графа зависимости AXFR/IXFR. Этот сервер указывается в поле SOA MNAME для зоны и может присутствовать в NS RR. По определению в зоне имеется лишь 1 первичный ведущий сервер.

Доменное имя указывает узел в структуре дерева пространства доменных имён. Каждый узел имеет набор (возможно, пустой) записей о ресурсах (Resource Record или RR). RR с совпадающими NAME, CLASS, TYPE называют набором записей о ресурсах (Resource Record Set или RRset).

Псевдокод в этом документе служит лишь примером. В случае несоответствия псевдокода тексту предпочтение отдаётся тексту. При неоднозначности текста можно использовать псевдокод для устранения неоднозначности.

1.1. Правила сравнения

1.1.1. RR считаются одинаковыми, если их поля NAME, CLASS, TYPE, RDLENGTH и RDATA совпадают. Сжатые имена в RDATA должны быть восстановлены до сравнения. Отметим, что поле срока действия (time-to-live или TTL) явно исключается из сравнения.

1.1.2. Правила сравнения символьных строк в именах заданы в параграфе 2.3.3 [RFC1035].

1.1.3. Шаблоны запрещены, т. е. символ * в обновлении соответствует лишь символу * в зоне и наоборот.

1.1.4. Псевдонимы запрещены, т. е. CNAME в зоне совпадает с CNAME в обновлении, иначе не будет выполняться. Все операции UPDATE выполняются на основе канонических имён.

1.1.5. Показанные ниже типы RR не могут быть добавлены в RRset. Если указанные правила сравнения выполняются, попытка добавить RR приведёт к замене прежней RR:

SOA
Сравниваются NAME, CLASS, TYPE. В зоне может быть лишь 1 запись SOA, даже если поля данных разные.

WKS
Сравниваются NAME, CLASS, TYPE, ADDRESS, PROTOCOL. Для этого кортежа возможно лишь 1 запись WKS RR, даже если маски служб различаются.

CNAME
Сравниваются NAME, CLASS, TYPE. Не может быть более 1 записи CNAME RR, даже если поля данных разные.

1.2. Склеивающие записи

При определении наличия доменного имени, использованного в протоколе UPDATE, в конкретной зоне считается, что имя находится в зоне, если оно принадлежит доменному имени этой зоны. Детали приведены в параграфе 7.18.

1.3. Новые выделенные значения

```
CLASS = NONE (254)
RCODE = YXDOMAIN (6)
RCODE = YXRRSET (7)
RCODE = NXRRSET (8)
RCODE = NOTAUTH (9)
RCODE = NOTZONE (10)
Opcode = UPDATE (5)
```

2. Формат сообщения Update

Формат сообщений DNS определён в параграфе 4.1 [RFC1035]. Нужны некоторые расширения формата (например, дополнительные коды ошибок, внесённые UPDATE), а некоторые поля требуется заменить (см. CLASS ниже). Формат сообщения UPDATE показан на рисунке.

```
+----------------------+
|        Header        |
+----------------------+
|         Zone         | Указывает зону для обновления
+----------------------+
|     Prerequisite     | RR или RRset, которые должны быть или отсутствовать
+----------------------+
|        Update        | RR или RRset для добавления или удаления
+----------------------+
|   Additional Data    | Дополнительные данные
+----------------------+
```

Раздел Header указывает, что это сообщение является UPDATE, и описывает размеры других разделов. В разделе Zone указана зона, обновляемая этим сообщением. Раздел Prerequisite указывает начальные инварианты (в смысле содержимого зоны), требуемые для этого обновления. В разделе Update указаны изменения, которые нужно внести, а раздел Additional Data Section содержит дополнительные сведения, которые могут потребоваться для обновления, но не являются его частью.

2.1. Вопросы транспорта

Транзакция обновления может быть передана в дейтаграмме UDP (если запрос помещается в неё) или через соединение TCP (на усмотрение запрашивающего). Для случая TCP формат сообщения описан в параграфе 4.2.2 [RFC1035].

2.2. Заголовок сообщения

Заголовок сообщений DNS определён в параграфе 4.1 [RFC 1035]. Не все коды операций (opcode) задают одинаковые наборы флагов, хотя на практике большинство битов, заданных для QUERY, идентичны флагам других операций. В UPDATE используется лишь 1 флаг (QR).

Формат сообщений DNS задаёт число записей для своий 4 разделов (Question, Answer, Authority, Additional). В UPDATE используются такие же поля и тот же формат разделов, но именование и назначение разделов отличается, как показано на рисунке, от заголовка из параграфа 4.1.1 в [RFC1035].

```
                                    1  1  1  1  1  1
  0  1  2  3  4  5  6  7  8  9  0  1  2  3  4  5
+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
|                      ID                       |
+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
|QR|   Opcode  |           Z          |   RCODE   |
+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
|                    ZOCOUNT                    |
+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
|                    PRCOUNT                    |
+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
|                    UPCOUNT                    |
+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
|                    ADCOUNT                    |
+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
```

ID

16-битовый идентификатор, назначаемый организацией, генерирующей запрос любого типа. Идентификатор копируется в соответствующий отклик и может применяться запрашивающим для сопоставления откликов с остающимися невыполненными запросами или сервером для обнаружения дубликатов запросов.

QR

Однобитовое поле, указывающее, является сообщение запросом (0) или откликом (1).

Opcode

4-битовое поле, задающее тип запроса в сообщении. Устанавливается инициатором запроса и копируется в отклик. Для сообщения UPDATE поле Opcode имеет значение 5.

Z

Резерв на будущее. Во всех запросах и откликах следует устанавливать значение 0. Ненулевые значения реализациям этой спецификации следует игнорировать.

RCODE

4-битовый код отклика, не применяемый в запросах. Значения и их смысл показаны в таблице ниже.

Имя	*Значение*	*Описание*
NOERROR	0	Нет ошибок.
FORMERR	1	Сервер имён не способен интерпретировать запрос из-за ошибки формата.
SERVFAIL	2	На сервере имён возник внутренний отказ при обработке запроса (например, ошибка ОС или тайм-аут пересылки).
NXDOMAIN	3	Не существует имени, которое должно быть.
NOTIMP	4	Сервер имён не поддерживает заданную Opcode операцию.
REFUSED	5	Сервер имён отказался выполнять указанную операцию по соображениям политики или безопасности.
YXDOMAIN	6	Существует имя, которого не должно быть.
YXRRSET	7	Существует RRset, которого не должно быть.
NXRRSET	8	Не существует набора RRset, который должен быть.
NOTAUTH	9	Сервер не является полномочным для зоны из раздела Zone.
NOTZONE	10	Имя из раздела Prerequisite или Update не относится к зоне, указанной в разделе Zone.

ZOCOUNT

Число RR в разделе Zone.

PRCOUNT

Число RR в разделе Prerequisite.

UPCOUNT

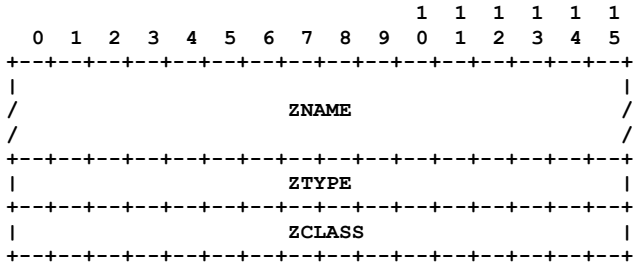
Число RR в разделе Update.

ADCOUNT

Число RR в разделе Additional Data.

2.3. Раздел Zone

Раздел Zone имеет формат, заданный в параграфе 4.1.2 [RFC1035], но поля переопределены, как показано на рисунке.



В UPDATE этот раздел служит для указания зоны, где будут обновляться записи. Все обновляемые записи должны быть в одной зоне, поэтому в разделе Zone разрешено указывать лишь 1 запись. Поле ZNAME указывает имя зоны, в ZTYPE должно быть указано SOA, а ZCLASS задаёт класс зоны.

2.4 Раздел Prerequisite

В этом разделе указывается набор условий RRset, которые должны быть выполнены на момент приёма сообщения UPDATE первичным ведущим сервером. Формат раздела соответствует параграфу 4.1.3 в [RFC1035]. В разделе может быть указано 5 вариантов семантики, описанных в последующих параграфах.

(1) RRset существует (независимо от значения). Должна быть хотя бы одна RR с указанными NAME и TYPE (в зоне и классе, указанных в разделе Zone).

(2) RRset существует (в зависимости от значения). Должен быть набор RR с указанными NAME и TYPE, имеющих те же элементы с такими же RDATA, как в RRset из этого раздела.

(3) RRset не существует. Нет RR с указанными NAME и TYPE (в зоне и классе, указанных в разделе Zone).

(4) Имя используется. Должна быть хотя бы одна запись RR с заданным NAME (в зоне и классе, указанных в разделе Zone). Отметим, что условие **не** выполняется для пустых нетерминальных имён.

(5) Имя не используется. Нет RR какого-либо типа, принадлежащий указанному NAME. Отметим, что условие **выполняется** для пустых нетерминальных имён.

2.4.1. RRset существует (независимо от значения)

Должна быть хотя бы одна RR с указанными NAME и TYPE (в зоне и классе, указанных в разделе Zone).

Для этого условия запрашивающий добавляет в раздел одну RR, где NAME и TYPE совпадают со значениями требуемого в зоне RRset. RDLENGTH = 0, поэтому поле RDATA пусто. Для CLASS должно указываться значение ANY, чтобы отличить это условие от наличия RR с RDLENGTH = 0 (например, NULL). TTL= 0.

2.4.2. RRset существует (в зависимости от значения)

Должен быть набор RR с указанными NAME и TYPE, имеющих те же элементы с такими же RDATA, как в RRset из этого раздела. Хотя порядок RRset не задан и не учитывается при сравнении, наборы должны быть идентичны по масштабу.

Для этого условия запрашивающий добавляет в раздел RRset, чьё наличие требуется. NAME и TYPE соответствуют означенному RRset, CLASS - это класс зоны. TTL должно быть 0 и игнорируется при проверке идентичности RRset.

2.4.3. RRset не существует

Нет RR с указанными NAME и TYPE (в зоне и классе, указанных в разделе Zone).

Для этого условия запрашивающий добавляет в раздел одну RR, где NAME и TYPE совпадают со значениями недопустимого в зоне RRset. RDLENGTH = 0, поэтому поле RDATA пусто. Для CLASS должно указываться значение NONE, чтобы отличить это условие от наличия RR с RDLENGTH = 0 (например, NULL). TTL= 0.

2.4.4. Имя используется

Должна быть хотя бы одна запись RR с заданным NAME (в зоне и классе, указанных в разделе Zone). Отметим, что условие **не** выполняется для пустых нетерминальных имён.

Для этого условия запрашивающий добавляет в раздел одну RR, где NAME совпадает с именем, для которого требуется владение RR. RDLENGTH = 0, поэтому поле RDATA пусто. Для CLASS должно указываться значение ANY, чтобы отличить это условие от проверки наличия RRset. TTL= 0.

2.4.5. Имя не используется

Имя не используется. Нет RR какого-либо типа, принадлежащий указанному NAME. Отметим, что условие **выполняется** для пустых нетерминальных имён.

Для этого условия запрашивающий добавляет в раздел одну RR, где NAME совпадает с именем, для которого требуется отсутствие владения RR. RDLENGTH = 0, поэтому поле RDATA пусто. Для CLASS должно указываться значение, для TYPE - ANY. TTL= 0.

2.5. Раздел Update

В этом разделе содержатся RR для добавления или удаления из зоны. Формат раздела соответствует параграфу 4.1.3 в [RFC1035]. Имеется 4 варианта семантики содержимого радела, описанные ниже.

(1) Добавить записи RR в RRset.

(2) Удалить RRset.

(3) Удалить все RRset для имени.

(4) Удалить RR из RRset.

2.5.1. Добавление RRset

В разделе Update указываются RR, чьи NAME, TYPE, TTL, RDLENGTH и RDATA соответствуют добавляемым, а CLASS совпадает с классом зоны. Первичный ведущий сервер имён отбрасывает дубликаты RR без уведомления.

2.5.2. Удаление RRset

В разделе Update указывается одна RR, чьи NAME и TYPE соответствуют удаляемому RRset. В поле TTL должно быть указано значение 0, в противном случае первичный ведущий сервер не использует его. В поле CLASS должно быть значение ANY. Поле RDLENGTH должно иметь значение 0, следовательно поле RDATA должно быть пустым. Если такого RRset нет, первичный ведущий сервер игнорирует RR.

2.5.3. Удаление всех RRset для Name

В разделе Update указывается одна RR, где NAME - имя, которое нужно удалить из RRset. В поле TYPE должно быть значение ANY, поле TTL должно иметь значение 0, в противном случае оно не используется первичным ведущим сервером. В поле CLASS должно быть значение ANY, поле RDLENGTH должно иметь значение 0, а RDATA быть пустым. Если указанных RRset не существует, Update RR просто игнорируется первичным ведущим сервером.

2.5.4. Удаление RR из RRset

В разделе Update указываются RR, чьи NAME, TYPE, TTL, RDLENGTH и RDATA соответствуют удаляемым. В поле TYPE должно быть значение ANY, поле TTL должно иметь значение 0, в противном случае оно не используется первичным ведущим сервером. В поле CLASS должно быть значение NONE, чтобы отличить от случая добавления RR. Если указанных RR не существует, Update RR просто игнорируется первичным ведущим сервером.

2.6. Раздел Additional Data

В этом разделе указываются RR, связанные с самим обновлением или добавляемыми им RR. Например, здесь следует указывать склеивающие записи, находящиеся вне зоны (RR, указанные новыми NS RR). Сервер может использовать или игнорировать внешние склеивающие записи по своему усмотрению. Формат раздела задан в параграфе 4.1.3 [RFC1035].

3. Поведение сервера

При получении запроса UPDATE сервер передаёт запрашивающему сигнал NOTIMP, если код операции UPDATE не распознан или не реализован. В иных случаях обработка выполняется, как описано ниже.

3.1. Обработка раздела Zone

3.1.1. Проверяется раздел Zone на предмет наличия в точности одной RR с ZTYPE = SOA и при невыполнении условия запрашивающему передаётся сигнал FORMERR. Затем просматриваются ZNAME и ZCLASS на предмет полномочности сервера для доны и при невыполнении условия запрашивающему передаётся сигнал NOTAUTH. Если сервер является ведомым для зоны, запрос пересылается первичному ведущему.

3.1.2. Псевдокод обработки раздела Zone

```
if (zcount != 1 || ztype != SOA)
     return (FORMERR)
if (zone_type(zname, zclass) == SLAVE)
     return forward()
if (zone_type(zname, zclass) == MASTER)
     return update()
return (NOTAUTH)
```

В параграфах 3.2 - 3.8 описано поведение первичного ведущего сервера, в разделе 6 - поведение пересылающего.

3.2. Обработка раздела Prerequisite

После раздела Zone проверяется раздел Prerequisite на предмет соответствия текущего состояния зоны заданным условиям. Если NAME в какой-либо RR не относится к зоне, заданной в разделе Zone (см. определение в параграфе 1.2), запрашивающему передаётся сигнал NOTZONE.

3.2.1. Для RR из этого раздела с CLASS = ANY проверяется, что поля TTL и RDLENGTH имеют значение 0, в противном случае запрашивающему передаётся сигнал FORMERR. Если TYPE = ANY, проверяется наличие хотя бы одной RR в зоне, для которой NAME совпадает с именем в Prerequisite RR, иначе запрашивающему передаётся сигнал NXDOMAIN. При TYPE, отличном от ANY, проверяется наличие хотя бы одной RR в зоне, для которой NAME и TYPE совпадают с указанными в Prerequisite RR, иначе запрашивающему передаётся сигнал NXRRSET.

3.2.2. Для RR из этого раздела с CLASS = NONE проверяется, что поля TTL и RDLENGTH имеют значение 0, в противном случае запрашивающему передаётся сигнал FORMERR. Если TYPE = ANY, проверяется отсутствие RR в зоне, для которой NAME совпадает с именем в Prerequisite RR, иначе запрашивающему передаётся сигнал YXDOMAIN. При TYPE, отличном от ANY, проверяется отсутствие RR в зоне, для которой NAME и TYPE совпадают с указанными в Prerequisite RR, иначе запрашивающему передаётся сигнал YXRRSET.

3.2.3. Для RR из этого раздела с CLASS = ZCLASS проверяется, что TTL = 0, иначе запрашивающему передаётся сигнал FORMERR. Затем создаётся RRset для каждой уникальной пары <NAME,TYPE> и каждый полученный набор сравнивается с наборами RRset в зоне на предмет точного совпадения. Если какой-либо из Prerequisite RRset не совпадает точно с RRset в зоне, запрашивающему передаётся сигнал NXRRSET. Если в какой-либо RR в этом разделе CLASS отличается от ZCLASS, NONE, ANY, запрашивающему передаётся сигнал FORMERR.

3.2.4. Таблица метазначений, применяемых в разделе Prerequisite

CLASS	*TYPE*	*RDATA*	*Значение*
ANY	ANY	пусто	Имя используется
ANY	rrset	пусто	RRset существует (независимо от значения)
NONE	ANY	пусто	Имя не используется
NONE	rrset	пусто	RRset существует
zone	rrset	rr	RRset существует (в зависимости от значения)

3.2.5. Псевдокод обработки раздела Prerequisite

```
for rr in prerequisites
     if (rr.ttl != 0)
          return (FORMERR)
     if (zone_of(rr.name) != ZNAME)
          return (NOTZONE);
     if (rr.class == ANY)
          if (rr.rdlength != 0)
               return (FORMERR)
          if (rr.type == ANY)
               if (!zone_name<rr.name>)
                    return (NXDOMAIN)
          else
               if (!zone_rrset<rr.name, rr.type>)
                    return (NXRRSET)
     if (rr.class == NONE)
          if (rr.rdlength != 0)
               return (FORMERR)
          if (rr.type == ANY)
               if (zone_name<rr.name>)
                    return (YXDOMAIN)
          else
               if (zone_rrset<rr.name, rr.type>)
                    return (YXRRSET)
     if (rr.class == zclass)
          temp<rr.name, rr.type> += rr
     else
          return (FORMERR)

for rrset in temp
     if (zone_rrset<rrset.name, rrset.type> != rrset)
          return (NXRRSET)
```

3.3. Проверка полномочий запрашивающего

3.3.1. Далее могут проверяться полномочия запрашивающего на изменение RR, указанных в разделе Update, в зависимости от реализации и использования механизмов, указанных в последующем протоколе Secure DNS Update. Если у запрашивающего нет полномочий вносить изменения, сервер может записать предупреждения в свой журнал операций, передать запрашивающему сигнал REFUSED или игнорировать проблему с полномочиями и выполнить обновление.

3.3.2. Хотя детали обработки определяются реализацией, при выполнении проверочных действий их следует помещать перед действиями по обновлению, поскольку возникновение условия REFUSED после частичного обновления потребует отмены внесённых изменений и возврата к исходному состоянию до отправки ответа запрашивающему.

3.3.3. Псевдокод проверки полномочий

```
if (security policy exists)
     if (this update is not permitted)
          if (local option)
               log a message about permission problem
          if (local option)
               return (REFUSED)
```

3.4. Обработка раздела Update

Затем обрабатывается раздел Update, как описано ниже.

3.4.1. Предварительное сканирование

Раздел Update разбирается по RR и в каждой записи проверяется значение CLASS. Если это не ANY, NONE или значение Zone Class, запрашивающему передаётся сигнал FORMERR. В соответствии с определением параграфа 1.2, поле NAME каждой RR должно относиться к зоне из раздела Zone, иначе запрашивающему передаётся сигнал NOTZONE.

3.4.1.1. Для RR, где CLASS отличается от ANY, проверяется значение TYPE и если это ANY, AXFR, MAILA, MAILB, иной метатип QUERY или тип не распознан, запрашивающему передаётся сигнал FORMERR.

3.4.1.2. Для RR с CLASS ANY или NONE проверяется условие TTL = 0 и при ином значении запрашивающему передаётся сигнал FORMERR. Для любой RR с CLASS = ANY проверяется условие RDLENGTH = 0 (т. е. поле RDATA пусто) и отличие TYPE от AXFR, MAILA, MAILB иных метатипов QUERY (кроме ANY) и нераспознанных типов, иначе запрашивающему передаётся сигнал FORMERR.

3.4.1.3. Псевдокод предварительного сканирования раздела Update

```
[rr] for rr in updates
     if (zone_of(rr.name) != ZNAME)
          return (NOTZONE);
     if (rr.class == zclass)
          if (rr.type & ANY|AXFR|MAILA|MAILB)
               return (FORMERR)
     elsif (rr.class == ANY)
          if (rr.ttl != 0 || rr.rdlength != 0
              || rr.type & AXFR|MAILA|MAILB)
               return (FORMERR)
     elsif (rr.class == NONE)
          if (rr.ttl != 0 || rr.type & ANY|AXFR|MAILA|MAILB)
               return (FORMERR)
     else
          return (FORMERR)
```

3.4.2. Раздел Update

Найденные в разделе Update записи RR обрабатываются по порядку, как описано ниже.

3.4.2.1. При возникновении в процессе обработки раздела системной ошибки (например, проблем с памятью или оборудованием в постоянном хранилище), запрашивающему передаётся сигнал SERVFAIL и отменяются все изменения, внесённые в зону данной транзакцией.

3.4.2.2. В зону добавляются все RR из раздела Update, где CLASS совпадает с ZCLASS. В случае дублирования RDATA (что для SOA RR происходит всегда, а для WKS RR - при совпадении полей ADDRESS и PROTOCOL) RR из Zone заменяется RR из Update. Если TYPE = SOA и в Zone нет SOA RR или новое значение SOA.SERIAL не больше (в соответствии с [RFC1982]) имеющегося в Zone SOA RR, Update RR игнорируется. Если Update RR - это CNAME, а Zone RRset не CNAME или наоборот, CNAME Update RR, игнорируется, иначе CNAME RR в Zone заменяется CNAME RR из Update.

3.4.2.3. Для любых Update RR с CLASS = ANY и TYPE = ANY все Zone RR с тем же NAME удаляются, если NAME отличается от ZNAME, а при совпадении имён удаляются лишь RR, где TYPE не является SOA или NS. Для любых Update RR с CLASS = ANY и TYPE, отличным от ANY, удаляются все Zone RR с тем же NAME и TYPE, если NAME не совпадает с ZNAME, а при совпадении записи SOA и NS не удаляются.

3.4.2.4. Для любых Update RR с классом NONE удаляются все Zone RR, где NAME, TYPE, RDATA и RDLENGTH совпадают с Update RR, за исключением случаев, когда NAME совпадает с ZNAME, TYPE = SOA или TYPE =NS и соответствующая Zone RR является единственной NS, оставшейся в RRset (тогда Update RR игнорируется).

3.4.2.5. Сигнализация запрашивающему отсутствия ошибок.

3.4.2.6. Таблица метазначений, применяемых в разделе Update

CLASS	*TYPE*	*RDATA*	*Значение*
ANY	ANY	пусто	Удалить для имени все RRset
ANY	rrset	пусто	Удалить RRset
NONE	rrset	rr	Удалить RR из RRset
zone	rrset	rr	Добавить RRset

3.4.2.7. Псевдокод обработки раздела Update

```
[rr] for rr in updates
     if (rr.class == zclass)
          if (rr.type == CNAME)
               if (zone_rrset<rr.name, ~CNAME>)
                    next [rr]
          elsif (zone_rrset<rr.name, CNAME>)
```

```
                    next [rr]
               if (rr.type == SOA)
                    if (!zone_rrset<rr.name, SOA> ||
                        zone_rr<rr.name, SOA>.serial > rr.soa.serial)
                         next [rr]
               for zrr in zone_rrset<rr.name, rr.type>
                    if (rr.type == CNAME || rr.type == SOA ||
                        (rr.type == WKS && rr.proto == zrr.proto &&
                         rr.address == zrr.address) ||
                        rr.rdata == zrr.rdata)
                         zrr = rr
                         next [rr]
               zone_rrset<rr.name, rr.type> += rr
          elsif (rr.class == ANY)
               if (rr.type == ANY)
                    if (rr.name == zname)
                         zone_rrset<rr.name, ~(SOA|NS)> = Nil
                    else
                         zone_rrset<rr.name, *> = Nil
               elsif (rr.name == zname &&
                      (rr.type == SOA || rr.type == NS))
                    next [rr]
               else
                    zone_rrset<rr.name, rr.type> = Nil
          elsif (rr.class == NONE)
               if (rr.type == SOA)
                    next [rr]
               if (rr.type == NS && zone_rrset<rr.name, NS> == rr)
                    next [rr]
               zone_rr<rr.name, rr.type, rr.data> = Nil
     return (NOERROR)
```

3.5. Стабильность

При изменении зоны операцией UPDATE сервер должен зафиксировать изменения в энергонезависимом хранилище до отправки отклика запрашивающему и ответов на любые запросы или переноса изменённой зоны. Серверу разумно сохранять обновлённые записи, пока перезагрузка системы или отказ питания не вызовут включение этих записей в зону при следующем запуске сервера. Имеет смысл также копировать изменённую зону целиком в энергонезависимое хранилище после каждого обновления, хотя это будет снижать производительность для больших зон.

3.6. Отождествление зоны

Если операция обновления меняет значение SOA SERIAL, изменение должно быть положительным (рост) в соответствии с арифметикой по модулю 2^{32}, описанной в [RFC1982]. Попытки установить в SOA значение SERIAL меньше текущего должны просто игнорироваться первичным ведущим сервером. Если операция обновления не меняет SOA SERIAL, серверу нужно автоматически инкрементировать это значение до того, как SOA, любое изменённое имя, RR или RRset будут включены в отклик или перенос зоны. Реализация первичного ведущего сервера может автоматически инкрементировать SOA SERIAL по любому из указанных ниже событий.

(1) Каждая операция обновления.

(2) Имя, RR или RRset в зоне были изменены и стали видимыми для клиента DNS, поскольку он видел SOA до инкрементирования и вскоре сможет увидеть новую запись SOA.

(3) Прошло настраиваемое в конфигурации время с момента последней операции обновления. Этот интервал следует делать не более трети от времени обновления зоны (refresh), а по умолчанию следует использовать меньшее из значений этого периода и 300 сек.

(4) Применено настраиваемое число обновлений с момента последнего изменения SOA. По умолчанию для этого параметра следует использовать значение 100.

Необходимо строго синхронизировать содержимое зоны и SOA SERIAL. Если зона изменяется, нужно изменять и SOA.

3.7. Неделимость транзакций

В процессе обработки транзакции UPDATE сервер должен обеспечивать её неделимость по отношению к другим (одновременным) транзакциям UPDATE и QUERY. Длв транзакции не могут обрабатываться одновременно, если любая из них зависит от результата другой. В частности, транзакции QUERY не могут извлекать RRset, которые были частично изменены одновременной операцией UPDATE, а выполнение UPDATE не может начинаться с предварительных условий, которые могут быть изменены по завершении другой одновременной транзакции UPDATE. Если две транзакции UPDATE меняют одни имена, RR или RRset, они должны выполняться последовательно.

3.8. Отклик

В конце операции UPDATE будет известен код отклика. Сообщение с откликом создаётся путём копирования полей ID и Opcode из запроса и копирования ZOCOUNT, PRCOUNT, UPCOUNT, ADCOUNT и соответствующих разделов или установки в 0 этих счётных полей без включения каких-либо частей исходного обновления. Устанавливается (1) бит QR и отклик передаётся запрашивающему. Если запрашивающий использовал протокол UDP, отклик передаётся в порт отправителя UDP у запрашивающего. При использовании TCP отклик передаётся через созданное запрашивающим соединение TCP.

4. Поведение запрашивающего

4.1. С точки зрения запрашивающего любой полномочный для зоны сервер может казаться способным обрабатывать запросы на обновление, хотя фактически основной файл зоны может менять только её

первичный ведущий сервер. Предполагается, что запрашивающий знает имя зоны, которую он намерен обновить, а также знает или может определить серверы имён этой зоны.

4.2. Если желательно упорядочить обновления, запрашивающему нужно знать значение имеющейся SOA RR. Запрашивающий, обновляющий SOA RR, должен увеличить поле SOA SERIAL (в соответствии с [RFC1982]), сохраняя другие поля SOA, если он явно не меняет их. В полу SOA SERIAL недопустимо значение 0.

4.3. Если у запрашивающего есть веские основания полагать, что все серверы зоны будут одинаково доступны, ему следует сначала попытаться обратиться к первичному ведущему серверу зоны (как указано полем SOA MNAME, если оно совпадает с неким NS NSDNAME), чтобы избавиться от ненужных пересылок между ведомыми серверами. Отметим, что первичный ведущий сервер может быть доступен не всем запрашивающим из-за межсетевых экранов или разделения сети.

4.4. Когда серверы имён для зоны найдены и, возможно, отсортированы, чтобы первыми в списке были те, которые с большей вероятностью доступны и/или поддерживают операцию UPDATE, запрашивающий создаёт сообщение UPDATE показанной ниже формы и отправляет его первому серверу имён из списка.

ID	***(новые значения)***
Opcode	UPDATE
Zone zcount	1
Zone zname	(имя зоны)
Zone zclass	(класс зоны)
Zone ztype	T_SOA
Раздел Prerequisite	(см. выше)
Раздел Update	(см. выше)
РазделAdditional Data	(пусто)

4.5. Если запрашивающий получает отклик с RCODE, отличным от SERVFAIL и NOTIMP, он возвращает вызвавшему походящий отклик.

4.6. Если получен отклик с RCODE SERVFAIL или NOTIMP, а также при отсутствии отклика в зависящем от реализации интервале времени или получении сообщения ICMP о недоступности сервера, запрашивающий удаляет непригодный сервер из внутреннего списка серверов имён и пытается обратиться к следующему (пока список серверов не пуст). Если в списке не осталось серверов, запрашивающий возвращается вызвавшему его соответствующее сообщение об ошибке.

5. Обнаружение дубликатов, упорядочение и взаимоисключение

5.1. Для корректной работы могут потребоваться механизмы обеспечения идемпотентности, упорядочения запросов UPDATE и обеспечения взаимных исключений. Сообщение или отклик UPDATE могут быть недоставлены, а также доставлен 1 или несколько раз. Дублирование дейтаграмм представляет особый интерес, поскольку с ним связаны так называемые replay-атаки, где корректные запросы злонамеренно дублируются атакующим.

5.2. Несколько запросов или откликов UPDATE, находящихся в процессе передачи, могут быть доставлены в любом порядке в результате изменения топологии сети, распределения нагрузки или пересылки по нескольким путям, где несколько ведомых серверов будет пересылать их одному первичному ведущему. В некоторых случаях может потребоваться, чтобы более раннее обновление не применялось, когда уже применено последующее (раньше или позже определяется по внешним часам, видимым некому множеству запрашивающих, а не по порядку получения запросов первичным ведущим сервером).

5.3. Запрашивающий может обеспечить идемпотентность транзакций, явно удаляя некую «маркерную RR» (вместо удаления RRset, в который эта запись входит), а затем добавляя новую «маркерную RR» с другим полем RDATA. В разделе Prerequisite следует указывать, что исходная «маркерная RR» должна присутствовать, чтобы это сообщение UPDATE было воспринято сервером.

5.4. Если запрос дублируется из-за ошибки в сети, все дубликаты будут приводить к отказу, поскольку лишь первый запрос будет содержать исходную «маркерную RR» и знать предыдущее значение. Решение об использовании «маркерной RR» и выборе для неё конкретной записи остаётся за разработчиками приложения, хотя одним из очевидных вариантов является SOA RR для зоны, как описано ниже.

5.5. Запрашивающий может упорядочить обновления с помощью внешней синхронизации использования в них последовательных значений «маркерной RR». Взаимное исключение можно считать вырожденным случаем с использованием одной «маркерной RR».

5.6. Особый случай пересечения порядка обновлений и дублирования дейтаграмм возникает при изменении RR на новое значение с последующим возвратом к прежнему. Без «маркерной RR» такая последовательность обновлений может оставить зону в неопределённом состоянии, если дейтаграммы дублируются.

5.7. Для обеспечения неделимости циклов из нескольких транзакций чтение-обновление-запись, запрашивающий может сначала извлечь SOA RR и создать сообщение UPDATE, где одним из условий является старая SOA RR. Затем задаются обновления, которые удалят эту SOA RR и добавят новую с большим SOA SERIAL вместе с фактическими условиями и обновлениями, являющимися целью транзакции. Если транзакция пройдёт успешно, запрашивающий будет знать, что RR не были изменены другой стороной.

6. Пересылка

Когда ведомый сервер зоны пересылает сообщение UPDATE в направлении первичного ведущего сервер зоны, он должен выделить новый идентификатор и приготовиться к выполнению роли «пересылающего сервера», который является запрашивающим по отношению к forward-серверу, куда пересылается сообщение.

6.1. Набор серверов пересылки будет совпадать с набором серверов, которые ведомый сервер зоны будет использовать в качестве источников данных AXFR или IXFR. Хотя исходный запрашивающий может применить

NS RRset зоны для обнаружения сервера обновлений, пересылающий всегда будет пересылать данные в направлении назначенного первичного ведущего сервера зоны.

6.2. Если исходный запрашивающий использовал TCP, созданное им соединение TCP остаётся открытым и пересылающий сервер должен использовать TCP для пересылки сообщения. При использовании исходным запрашивающим протокола UDP пересылающий сервер может использовать для пересылки сообщения UDP или TCP по усмотрению его создателя.

6.3. Если граф зависимостей AXFR является глубоким и включает межсетевые экраны и множество областей подключения, разумно делать пересылающими сами серверы пересылки. В большинстве случаев граф зависимостей AXFR будет неглубоким и сервером пересылки будет первичный ведущий сервер.

6.4. Пересылающий не будет отвечать запрашивающему, пока не получит отклик от сервера, куда сообщение было переслано. Транзакции UPDATE с пересылающими синхронизируются по времени относительно исходного запрашивающего и первичного ведущего сервера.

6.5. При наличии нескольких возможных источников данных AXFR (следовательно, нескольких возможных forward-серверов) пересылающий будет использовать при тайм-аутах и ошибках связности ту же стратегию отката, которая применяется при AXFR (это зависит от реализации).

6.6. При получении пересылающим отклика от forward-сервера он копирует отклик в новое сообщение отклика, назначает этому сообщению идентификатор своего запрашивающего и передаёт сообщением тому.

7. Устройство, реализация, операции и протокол

Ниже приведены некоторые принципы, которыми руководствовались при разработке данной спецификации UPDATE. Они не являются частью формальной спецификации и при наличии расхождений между этим разделом и другими частями документа преимущество отдаётся формальной спецификации.

7.1. Использование метазначений для CLASS возможно лишь потому, что предполагается нахождение всех RR в одной зоне, и CLASS является атрибутом зоны, а не RRset. Поэтому раздел Zone является обязательным.

7.2. Поскольку при обработке раздела Update не может быть ошибок, связанных с присутствием или отсутствием данных, все связанные с этим зависимости следует указывать в разделе Prerequisite.

7.3. Раздел Additional Data может служить для предоставления серверу внешней склеивающей записи, которая потребуется в рефералах. Например, при добавлении в HOME.VIX.COM новой NS RR, задающей сервер имен NS.AU.OZ, запись A RR для NS.AU.OZ можно включить в раздел Additional Data. Серверы могут использовать эти сведения или игнорировать их по усмотрению исполнителя. Не рекомендуется кэшировать эти данные для использования в последующих откликах DNS.

7.4. Раздел Additional Data можно использовать, если некоторые RR, которые затем потребуются для Secure DNS Update, на деле не являются обновлениями зоны, а содержат вспомогательные ключи или подписи, не предназначенные для сохранения в зоне (как было бы при обновлении), но требуемые для проверки операции обновления.

7.5. Предполагается, что в отсутствие Secure DNS Update сервер будет воспринимать лишь обновления, поступающие от источника, адрес которого статически задан в описании серверов из первичного файла зоны. Вероятно, в этот статический список будут включаться серверы DHCP.

7.6. С помощью этого протокола невозможно создать зону, поскольку нет возможности указать ведомому серверу, кто является его ведущим. Предполагается, что в будущем протокол может быть расширен для охвата этого случая. Поэтому в настоящее время добавление SOA RR не поддерживается. По тем же причинам не поддерживается удаление SOA RR.

7.7. Условие указания наличия хотя бы одной RR семантически отличается от QUERY, поскольку QUERY при запросе RRset для этого имени будет возвращать <NOERROR,ANCOUNT=0>, а не NXDOMAIN, тогда как предварительное условие из UPDATE [параграф 2.4.4] не будет выполняться.

7.8. Возможна потеря отклика UDP в пути и повторный запрос по тайм-ауту. В этом случае операция UPDATE, выполненная в первый раз при получении запроса первичным ведущим сервером, может в конечном итоге казаться неудачной при получении запрашивающим отклика на дубликат запроса. Это обусловлено тем, что предварительные условия уже не будут выполняться после выполненного обновления. Поэтому запрашивающие, которым нужен точный код отклика, должны использовать TCP.

7.9. Поскольку запрашивающий, которому нужен точный отклик, будет инициировать свою транзакцию UPDATE по протоколу TCP, пересылающий, принявший запрос по TCP, должен и перслать его по протоколу TCP.

7.10. Отсрочка автоинкремента SOA SERIAL позволяет сделать переход серийного номера через границу 2^{32} более редким. Видимые клиентам DNS значения SOA SERIAL должны различаться, если содержимое зоны различно. Отметим, что SOA раздела Authority в отклике QUERY является формой видимости для этого предварительного условия.

7.11. В поле SOA SERIAL для зоны никогда не следует указывать значение 0 из-за проблем совместимости с некоторыми старыми, но широко распространёнными реализациями DNS. Если инкрементирование SOA SERIAL ведёт к значению 0 (переход через границу 2^{32}), необходимо повторить инкрементирование или установить в поле значение 1. Более подробно это рассмотрено в [RFC1982].

7.12. Из-за минимизации TTL, требуемой при кэшировании RRset, рекомендуется устанавливать одно значение TTL для всего RRset. Хотя формат сообщений DNS разрешает устанавливать разные значения TTL в одном RRset и различия могут возникать в зоне, это может приводить к противоречивым результатам и не рекомендуется.

7.13. В управлении срезами зон имеются неясные моменты для операций добавления и удаления раздела Update. Можно удалить NS RR, если она не является последней NS RR в корне зоны. При удалении всех RR для имени, записи SOA и NS RR в корне не удаляются. При удалении RRset невозможно удалить наборы SOA и

NS на вершине зоны. Попытка добавить SOA будет считаться операцией замены, если SOA уже существует, или отсутствием операции (no-op), если SOA будет новой.

7.14. При добавлении RR на первичном ведущем сервере не требуется семантической проверки. Поэтому запрашивающий может вызвать добавление CNAME, NS или RR иного типа даже если их целевого имени не существует или оно не имеет соответствующих RRset, чтобы сделать исходную RR полезной. Первичному ведущему серверу, реализующему этот вид проверки, следует тщательно избегать внезоновых зависимостей (чью достоверность нельзя проверить в полномочном источнике), а также реализовать все такие проверки на этапе предварительного сканирования.

7.15. Нетерминальные и шаблонные CNAME недостаточно хорошо определены в [RFC1035] и их использование может приводить кнепредсказуемым результатам, поэтому оно не рекомендуется.

7.16. Пустые нетерминальные узлы (с потомками, но без собственных RR) будут приводить к откликам <NOERROR,ANCOUNT=0> на запрос любого типа для такого имени. Пустые терминальные узлы не предусмотрены, поэтому при удалении всех RR терминального узла имя узла больше не будет использоваться и запросы любого типа для этого имени будут приводить к отклику NXDOMAIN.

7.17. В глубоком графе AXFR исторически не считалась ошибкой зависимость ведомых серверов друг от друга. Такая конфигурация применялась для того, чтобы зону можно было передать от первичного ведущего сервера всем ведомым, даже если не у всех из них есть постоянная связность с первичным ведущим. Применение в UPDATE графа зависимостей AXFR для пересылки запрещает такие циклические зависимости, поскольку в пересылке UPDATE нет обнаружения петель, аналогичного предварительной проверке SOA SERIAL в AXFR.

7.18. Существовавшие ранее имена, скрытые новым срезом зоны, по-прежнему считаются частью родительской зоны для целей переноса, даже если запросы для таких имён будут направляться на серверы новой субзоны. Если срез зоны удаляется, все имена родительской зоны, скрытые ранее им, снова становятся видимыми для запросов (уточнение [RFC1034]).

7.19. Если сервер полномочен для зоны и её потомка, на запросы для имён на срезе между ними полномочными будут только данные из дочерней зоны (уточнение [RFC1034]).

7.20. Упорядочение обновлений с использованием SOA RR проблематично, поскольку нет возможности узнать, какая из NS RR в зоне представляет первичный ведущий сервер, а ведомые серверы зоны могут содержать устаревшие данные, если отсчёт их таймеров SOA.REFRESH не завершился с момента последнего изменения зоны на первичном ведущем сервере. Для зон, которым нужны упорядоченные обновления, рекомендуется использовать лишь серверы, реализующие NOTIFY (см. [RFC1996]) и IXFR (см. [RFC1995]), и чтобы клиент, получающий ошибку предварительных условий при попытке упорядоченного обновления, просто повторял попытку по истечении случайного интервала, чтобы зона могла стабилизироваться.

8. Вопросы безопасности

8.1. В отсутствие [RFC2137] или эквивалентной технологии описанный здесь протокол позволяет любому, кто имеет доступ к полномочному серверу имён, изменить содержимое любой зоны на этом сервере. Это серьёзно повышает уязвимость по сравнению с текущей технологией. Поэтому настоятельно рекомендуется не использовать описанный здесь протокол без [RFC2137] или эквивалентных мер строгой защиты, например, IPsec.

8.2. Возможны атаки на службы путём лавинной организации с пересылающими узлами сессий TCP, содержащих обновления, которые первичный ведущий сервер в конечном итоге отвергнет из-за проблем с правами доступа. Это связано с требованием к пересылающим узлам, получившим запрос по протоколу TCP, использовать синхронную сессию TCP для операции пересылки. Механизма управления соединениями из параграфа 4.2.2 в [RFC1035] достаточно для предотвращения масштабного ущерба от таких атак, но не для того, чтобы предотвратить оставление некоторых запросов безответными во время атаки.

Благодарности

Спасибо членам рабочей группы IETF DNSIND за их вклад и помощь, в частности, спасибо Rob Austein, Randy Bush, Donald Eastlake, Masataka Ohta, Mark Andrews, Robert Elz. Особая благодарность Bill Simpson, Ken Wallich и Bob Halley за рецензирование документа.

Литература

[RFC1035] Mockapetris, P., "Domain Names - Implementation and Specification", STD 13, RFC 1035, USC/Information Sciences Institute, November 1987.

[RFC1982] Elz, R., "Serial Number Arithmetic", RFC 1982, University of Melbourne, August 1996.

[RFC1995] Ohta, M., "Incremental Zone Transfer", RFC 1995, Tokyo Institute of Technology, August 1996.

[RFC1996] Vixie, P., "A Mechanism for Prompt Notification of Zone Changes", RFC 1996, Internet Software Consortium, August 1996.

[RFC2065] Eastlake, D., and C. Kaufman, "Domain Name System Protocol Security Extensions", RFC 2065, January 1997.

[RFC2137] Eastlake, D., "Secure Domain Name System Dynamic Update", RFC 2137, April 1997.

Адреса авторов

Yakov Rekhter
Cisco Systems
170 West Tasman Drive

San Jose, CA 95134-1706
Phone: +1 914 528 0090
EMail: yakov@cisco.com

Susan Thomson
Bellcore
445 South Street
Morristown, NJ 07960
Phone: +1 201 829 4514
EMail: set@thumper.bellcore.com

Jim Bound
Digital Equipment Corp.
110 Spitbrook Rd ZK3-3/U14
Nashua, NH 03062-2698
Phone: +1 603 881 0400
EMail: bound@zk3.dec.com

Paul Vixie
Internet Software Consortium
Star Route Box 159A
Woodside, CA 94062
Phone: +1 415 747 0204
EMail: paul@vix.com

Перевод на русский язык

Николай Малых

nmalykh@protokols.ru