Протокол SMTP

Протокол SMTP

1. Команды SMTP
    1. EHLO
    2. HELO
    3. MAIL
    4. RCPT
    5. DATA
    6. QUIT
    7. HELP
    8. VRFY, EXPN
    9. NOOP
    10. RSET
    11. SEND, SOML, SAML
    12. TURN
2. Ответы сервера SMTP
3. Пример диалога SMTP
4. Расширения ESMTP
    1. 8BITMIME
    2. CHECKPOINT
    3. SIZE
    4. ETRN
    5. ENHANCEDSTATUSCODES
    6. AUTH
    7. STARTTLS
    8. ATRN
    9. DSN
    10. DELIVERBY
    11. PIPELINING
    12. CHUNKING, BINARYMIME
5. Тестирование сервера SMTP
6. Протокол LMTP
7. Контрольные вопросы

8. Практическое задание

Простой протокол передачи почты – Simple Mail Transfer Protocol (**SMTP**) обычно используется на участке от **MUA** отправителя до ближайшего к получателю **MTA**.

Протокол разрабатывался в начале восьмидесятых годов прошлого века. Окончательная версия была закреплена в RFC 821 1 августа 1982 года. Все годы, прошедшие с того времени, протокол **SMTP** оставался одним из наиболее часто используемых протоколов семейства **TCP/IP** .

За это время принципиально изменились многие требования, касающиеся достоверности и защищенности передаваемых сообщений, значительно увеличились средний размер сообщений, и их количество, разнообразнее стала передаваемая информация: это уже не только текстовые сообщения на английском языке – сейчас электронные письма пишутся на многих языках и могут содержать вложения самых разных типов.

Однако протокол **SMTP** получил за время своего существования такое широкое распространение, что просто заменить его другим протоколом уже не представляется возможным. Вместо этого для него разрабатываются различные расширения (extensions), дополняющие возможности базового протокола. Дополненный расширениями протокол **SMTP** часто называют **ESMTP** (Extended **SMTP**).

Сам протокол изменился незначительно. На смену команде HELO , использовавшейся для начала диалога, пришла команда EHLO , позволяющая работать с расширениями **ESMTP** . Команды, применяемые для настройки почтовых систем и для получения справочной информации о пользователях, теперь используются значительно осторожнее, чем в восьмидесятые годы. Эти команды, к сожалению, создают удобства не только для сетевых администраторов, но и для злоумышленников. Поэтому такие команды обычно используют только на этапе настройки почтовой системы. В работающей системе их, как правило, отключают.

В апреле 2001 г . RFC 821 , который на сегодняшний день является основным стандартом, описывающим протокол **SMTP** . Новый стандарт учитывает изменения, произошедшие в Internet ‘е за восемнадцать с половиной лет.

SMTP может работать с различными протоколами транспортного уровня (см. RFC 821 и RFC 1090), но обычно используется **TCP** . За **SMTP** закреплен порт **TCP**25.

Почта по протоколу **SMTP** посылается от клиента к серверу. Клиент запрашивает соединение с сервером. После успешного установления соединения сервер сообщает клиенту свое доменное имя. Он также может сообщить тип и версию установленного программного обеспечения. Однако, из соображений безопасности, чтобы не дать потенциальному взломщику воспользоваться известными ошибками данной версии сервера **SMTP** , передача этой информации часто блокируется системными администраторами.

Ответ сервера, свидетельствующий о готовности к приему команд клиента, служит сигналом к началу диалога, в котором клиент последовательно посылает серверу команды и ожидает ответы, либо подтверждающие исполнение команд, либо сообщающих о невозможности исполнения, либо содержащих информацию, запрошенную клиентом.

3.1 Команды SMTP

Каждая команда **SMTP** начинается с ключевого слова – названия команды. За ним могут следовать параметры, отделенные пробелом.

Регистр символов, используемых во всех названиях команд и, за редким исключением, в параметрах базового протокола **SMTP** , не имеет значения. Однако в некоторых элементах расширений строчные и прописные символы могут различаться. Необходимо также учитывать, что левая часть почтового адреса, до символа @, может быть регистрозависимой.

В командах допускается использование только кодировки us – ascii , то есть символов, кодируемых семью битами. Это цифры, латинские буквы, и знаки препинания. Если информация передается восьмибитными блоками (октетами), старший бит должен быть равен нулю. Корректная интерпретация символов, старший, восьмой бит которых равен единице, например, русских букв, не гарантируется, использовать такие символы не следует.

Конец строк в протоколе **SMTP** обозначается последовательностью символов “возврат каретки” (шестнадцатеричный код 0 D) и “перевод строки” (шестнадцатеричный код 0A). Эта последовательность обозначается **CRLF** . Сервер начинает выполнение команды только получив от клиента строку, завершающуюся последовательностью **CRLF** .

Сервера **SMTP** должны принимать командные строки длинной до 512 символов. Это значение может быть увеличено по желанию разработчиков. Для серверов, поддерживающих расширения **ESMTP** , требующие дополнительных параметров, максимально допустимая длина командной строки увеличивается. Соответствующие требования приведены в **RFC** , описывающих эти расширения.

Если не используется расширение, позволяющее серверу принимать несколько команд подряд, клиент передает серверу следующую команду только после получения ответа на предыдущую.

Рассмотрим команды **SMTP** , необходимые для отправки сообщения.

1.1 EHLO (Расширенное HELO)

Формат команды:

EHLO полное_доменное_имя_клиента **CRLF**

или

EHLO адрес_отправителя **CRLF**

Диалог клиента и сервера, как правило, начинается с приветствия. В RFC 821 в качестве приветствия предлагалась

команда HELO . Однако с введением расширений **ESMTP** , эта команда была заменена на EHLO . Использование расширений **ESMTP** возможно только после выполнения команды EHLO .

Передача почты возможна только после выполнения одной из двух названых команд. Другие команды, не связанные с передачей почты (NOOP, HELP, EXPN, VRFY, RSET и QUIT), в принципе могут быть исполнены и без приветствия.

В качестве аргумента клиент передает серверу свое полное доменное имя, если таковое имеется. Если клиент не имеет доменного имени, например, если в качестве клиента выступает **MUA** , установленный на компьютере, получающем адрес динамически, то в качестве аргумента передается адрес электронной почты отправителя. Желательно, чтоб полученная от клиента информация была исчерпывающей для его идентификации.

Сервер проверяет соответствие указанного клиентом в приветствии доменного имени его адресу **IP** . Результат проверки добавляется к заголовку письма, но диалог продолжается независимо от достоверности полученного сервером идентификатора.

В ответ на команду EHLO сервер присылает список, каждая строка которого содержит ключевое слово, соответствующее расширению, поддерживаемому вызываемым сервером, и, при необходимости, уточняющие параметры. Это единственный предусмотренный базовым протоколом **SMTP** ответ сервера, в котором клиентская программа должна проанализировать не только числовой код ответа, но и его текст. Из ответа на команду EHLO клиент узнает, какие дополнительные функции он может использовать при отправке сообщения.

Если устаревшее программное обеспечение сервера не поддерживает команду EHLO , то выдается сообщение об ошибке. В этом случае клиент должен попытаться повторить приветствие, используя команду HELO . Естественно, расширениями **ESMTP** уже

не удастся воспользоваться.

1.2 HELO (Приветствие)

Формат команды:

HELO полное_доменное_имя_клиента **CRLF**

или

HELO адрес_отправителя **CRLF**

RFC 2821 рекомендует использовать команду HELO , только если программное обеспечение не поддерживает команду EHLO . Отличие этой команды только в том, что она делает невозможным использование расширений **ESMTP** .

В ответ на эту команду сервер сообщает, готов ли он к продолжению диалога.

1.3 MAIL (Отправитель)

Формат команды:

MAIL FROM : адрес_отправителя дополнительные_параметры **CRLF**

Отправка электронного сообщения невозможна без успешного выполнения этой команды, для каждого письма команда MAIL должна быть выполнена только один раз.

Команда MAIL может быть выполнена только после успешного выполнения команды EHLO или H E LO .

С помощью этой команды серверу сообщается адрес отправителя письма. На этот адрес письмо должно вернуться в случае невозможности доставки. Если возврат не желателен, адрес может быть оставлен пустым: <>:

MAIL FROM : <> **CRLF**

Обычно это бывает, если отправляемое сообщение уже и есть возвращаемое письмо. Это делается для того, чтобы избежать

петли: письмо не может быть доставлено адресату и возвращается отправителю, но, если оно не может быть доставлено отправителю, то посылается обратно и так далее. Если же адрес отправителя не известен, то попытки вернуть его предприниматься не будут: в случае невозможности доставки получателю, письмо будет удалено.

Поскольку базовый протокол **SMTP** не предусматривает никакой авторизации отправителя, адрес отправителя может быть указан произвольно и не может считаться достоверным. В последние годы большинство **MTA** проверяют существование почтового домена отправителя и отвергают сообщения с адресами отправителей в несуществующих доменах. К сожалению, более детальная проверка возможна только при использовании дополнительных средств авторизации и обычно не применяется.

Базовый протокол **SMTP** не предусматривает дополнительных параметров для команды MAIL , но такие параметры использует ряд расширений **ESMTP** .

1.4 RCPT (Получатель)

Формат команды:

RCPT TO: адрес_получателя дополнительные_параметры **CRLF**

Доставка сообщения возможна, только если указан хотя бы один доступный адрес получателя. Команда RCPT принимает в качестве аргумента только один адрес. Если нужно послать письмо большему числу адресатов, то команду RCPT следует повторять для каждого. Согласно RFC 2821 , сервера **SMTP** должны быть готовы принять до ста команд RCPT на одно сообщение. Если письмо адресовано большему числу получателей, то для оставшихся клиент должен передать сообщение повторно. Максимальное число получателей может быть изменено администратором.

Команда RCPT может быть выполнена только после успешного выполнения команды MAIL .

Сервер анализирует каждый адрес и после каждой команды RCPT выдает сообщение, свидетельствующее о возможности или невозможности доставки письма по указанному адресу.

Базовый протокол **SMTP** не предусматривает дополнительных параметров для команды RCPT, но такие параметры использует ряд расширений **ESMTP** .

1.5 DATA (Текст сообщения)

Формат команды:

DATA **CRLF**

С помощью этой команды серверу передается текст сообщения, состоящий из заголовка и отделенного от него пустой строкой тела сообщения.

Команда DATA может быть выполнена только после успешного выполнения хотя бы одной команды RCPT.

Команда DATA не требует никаких параметров и завершается последовательностью **CRLF**.

В ответ на правильно введенную команду DATA сервер сообщает о готовности к приему или об ошибке, если прием сообщения невозможен.

В случае положительного ответа сервера, клиент передает сообщение.

Передача заканчивается строкой, состоящей из одной точки. Эта строка не является частью сообщения и удаляется на приемной стороне. Чтобы исключить ложное срабатывание в случае, если сообщение содержит строку, состоящую из одной точки, на передающей стороне к началу каждой строки, начинающейся с точки, добавляется еще одна точка. На приемной стороне добавленные точки удаляются.

Распознав окончание сообщения, сервер должен принять решение о

возможности или невозможности доставки и послать соответствующий ответ клиенту. Сделать это следует как можно быстрее, если нет явных свидетельств невозможности доставки сообщения, его рекомендуется принять, сообщив клиенту об успешном завершении операции. Если доставка в последствии окажется невозможна, следует просто отправить письмо обратно, сообщив в квитанции причину отказа. Это делается для того, чтобы избежать проблемы, описанной в RFC1047 : не дождавшись ответа сервера, клиент разрывает соединение, считая передачу закончившейся неудачей, хотя сервер, возможно, позже осуществил доставку. Через некоторое время клиент снова пытается послать то же самое сообщение, что может привести к многократной передаче одного и того же письма. На ожидание ответа после завершения отправки сообщения рекомендуется выделять не меньше десяти минут. На ожидание других ответов отводятся от двух до пяти минут.

Необходимо принимать строки сообщения длиной до 1000 символов, включая последовательность **CRLF**, но не включая точку, добавляемую в начало строки во избежание ложного обнаружения конца сообщения.

Минимальный размер письма, которое должен принимать сервер **SMTP** – 64 килобайта, включая как тело сообщения, так и его заголовок.

1.6 QUIT (Выход)

Формат команды:

QUIT **CRLF**

Командой QUIT клиент заканчивает диалог с сервером. Сервер посылает подтверждение и закрывает соединение. Получив это подтверждение, клиент тоже прекращает связь.

Перечисленных команд вполне достаточно для того, чтобы

передать сообщение. Однако RFC 2821 предусматривает еще ряд команд, которые могут быть использованы в основном в процессе отладки сервера.

1.7 HELP (Помощь)

Формат команды:

HELP команда **CRLF**

или

HELP **CRLF**

Если команда HELP вызывается без параметров, сервер посылает клиенту список доступных команд. Если в качестве параметра передано название команды, то клиенту посылается описание этой команды. Серверам **SMTP** рекомендуется поддерживать эту команду без параметров. Описание отдельных команд посылать не обязательно.

1.8 VRFY (Проверить), EXPN (Раскрыть)

Формат команд:

VRFY адрес **CRLF**

EXPN адрес **CRLF**

Команда VRFY используется для проверки наличия указанного в качестве аргумента почтового ящика. В ответ сервер посылает информацию о владельце ящика или сообщение об ошибке, свидетельствующее о том, что указанный ящик не существует.

Если указанный адрес указывает на несколько почтовых ящиков, команда VRFY возвращает список пользователей, внесенных в соответствующую рассылку, либо сообщение об ошибке, свидетельствующее о неоднозначности ответа на полученный запрос.

Для получения адресов, внесенных в список рассылки,

используется команда EXPN .

Используя эти команду, злоумышленники могут получить список адресов пользователей сервера.

Обе команды могут быть полезны в процессе отладки сервера, потому они, как правило, поддерживаются. Но на серверах, используемых в реальном окружении, их обычно отключают. Возможен также вариант, при котором вызов команд возможен, но они проверяют только синтаксическую верность адреса.

Согласно RFC 821 , команды VRFY и EXPN не являются обязательными. Поэтому, если сервер их поддерживает, они должны быть перечислены в ответе сервера на команду EHLO , как расширения **ESMTP** .

1.9 NOOP (Пустая команда)

Формат команды:

NOOP параметры **CRLF**

В ответ на команду NOOP сервер посылает подтверждение выполнения. Никаких действий на сервере не производится, параметры команды игнорируются.

1.10 RSET (Сброс)

Формат команды:

RSET **CRLF**

Команда RSET аннулирует все переданные до нее на сервер данные. Процесс передачи сообщения следует начать заново.

1.11 SEND, SOML, SAML (Передача сообщения на терминал пользователя)

Формат команд:

SEND FROM: адрес_отправителя **CRLF**

SOML FROM: адрес_отправителя **CRLF**

SAML FROM: адрес_отправителя **CRLF**

Перечисленные команды используются вместо команды MAIL для передачи сообщения на терминал получателя (SEND) или в его почтовый ящик, если пользователь не активен или запретил прием сообщений (SOML) или и на терминал, и в почтовый ящик (SAML).

Описанные в RFC 821 устаревшими. Если их все же используют, то они должны быть перечислены в ответе на команду EHLO , как расширения **ESMTP** .

1.12 TURN (Смена направления передачи)

Формат команды:

TURN **CRLF**

Эта команда предназначена для почтовых серверов, не имеющих постоянного соединения с сетью. Они должны периодически, обычно по телефонной сети, соединяться с серверами, выполняющими функции промежуточных хранилищ сообщений, и забирать накопившуюся почту. Поскольку протокол **SMTP**предусматривает отправку сообщений только от клиента к серверу, для передачи в обратном направлении им необходимо поменяться ролями.

Команда TURN представляет потенциальную опасность, так как она может быть использована для перехвата чужой почты. Потому RFC 2821 категорически не рекомендует ее использовать. Хорошими альтернативами команде TURN являются расширения **ESMTP** ETRN и ATRN , рассматриваемые ниже.

3.2 Ответы сервера SMTP

На каждую команду клиента сервер посылает ответ, состоящий из числового кода и отделенной от него пробелом текстовой строки.

В большинстве случаев для правильной интерпретации ответа клиенту достаточно числового кода. Текстовая строка нужна для интерпретации ответа человеком. Исключение составляет ответ на команду EHLO , содержащий список расширений **ESMTP** , поддерживаемых сервером, а так же ответы на некоторые команды **ESMTP** .

Согласно RFC 2821 , код ответа состоит из трех цифр. Первая цифра кода может принимать следующие значения:

1. Предварительный положительный результат. Команда принята, но для ее выполнения сервер ожидает реакции клиента на посылаемую в этом ответе информацию. Клиент должен послать следующую команду для продолжения работы. В базовом протоколе **SMTP** не предусмотрено команд, требующих ответов такого типа.
2. Команда выполнена успешно.
3. Промежуточный положительный результат. Команда принята, но сервер ожидает от клиента дополнительные данные для завершения операции. Дополнительными данными может, например, быть текст сообщения в команде DATA .
4. Исполнение команды временно невозможно. Команда не может быть выполнена, но проблема может быть устранена. Клиенту следует попытаться повторить попытку через некоторое время.
5. Исполнение команды невозможно.

Вторая цифра может принимать следующие значения:

0 Синтаксическая ошибка, неправильное или недопустимое использование команды.
1 Ответ содержит запрошенную информацию.
2 Ответ о состоянии канала передачи.
5 Ответ информирует о состоянии принимающей почтовой системы.

Если ответ состоит из нескольких строк, то каждая из них начинается числовым кодом, который отделяется от

сопровождающего текста не пробелом, а символом “минус” (-). В последней строке цифровой код отделяется от текста пробелом. Каждая строка ответа заканчивается последовательностью **CRLF** .

В табл. 2 собраны ответы, предусмотренные для команд **SMTP** .

Код	**Расшифровка**	**Команды**
211	Состояние системы	HELP
214	Информация об использовании команд	HELP
220	Готовность к работе	Установление соединения
221	Канал передачи закрыт	QUIT
250	Команда выполнена успешно	EHLO, HELO, MAIL, RCPT DATA, RSET, VRFY, EXPN, NOOP
251	Почта для данного пользователя переадресована и будет доставлена по новому адресу *	RCPT , VRFY
252	Команда не будет выполнена, но доставка сообщения возможна. Ответ свидетельствует о том, что выполнение команд заблокировано из соображений безопасности, и не может быть интерпретирован как информация об опрашиваемом почтовом ящике	VRFY , EXPN
354	Команда DATA принята, ожидается текст сообщения, заканчивающийся строкой, состоящей из одной точки	DATA
421	Служба недоступна, связь прекращается. Ответ выдается при прекращении работы сервера во время сеанса связи	Любая

450	Доставка сообщения в данный момент не возможна: почтовый ящик не доступен	RCPT
451	Выполнение команды прервано: ошибка сервера	MAIL, RCPT, DATA
452	Команда не выполнена: недостаточно памяти	MAIL, RCPT, DATA
500	Синтаксическая ошибка, команда не понята (возможно, превышена допустимая длина строки)	Несуществующая команда
501	Синтаксическая ошибка в параметрах или аргументах (например, использование параметров в командах, не допускающих параметров)	Любая
502	Команда не поддерживается (отключена администратором)	VRFY, EXPN, HELP
503	Неправильный порядок команд	MAIL, RCPT, DATA
504	Параметр команды не поддерживается	EHLO, HELO, VRFY, EXPN, HELP
550	Команда не выполнена: почтовый ящик недоступен (не найден, доступ запрещен, выполнение команды запрещено администратором)	EHLO, HELO, MAIL, RCPT, VRFY, EXPN
551	Адрес пользователя изменился	RCPT , VRFY
552	Выполнение команды прервано: превышен выделенный объем памяти	MAIL, RCPT, DATA
553	Неправильный синтаксис адреса	MAIL, RCPT, VRFY
554	Служба **SMTP** на вызываемой машине не запущена	Установление соединения
554	Доставка не может быть осуществлена ни по одному адресу	DATA

* В случае переадресации почты допускается также использование

ответа 250. В этом случае клиент о переадресации не информируется. Сервер может также отказать в приеме почты для уже не существующего пользователя и послать ответ 551 с указанием нового адреса или ответ 550.

3.3 Пример диалога SMTP

Рассмотрим пример диалога по протоколу **SMTP** . В этом и в последующих примерах команды клиента помечены буквой C , а ответы сервера — буквой S .

S	220 foo.com Service Ready	Сервер представляется как foo.com и сообщает о готовности к приему команд
C	EHLO bar . com	Клиент представляется как bar.com
S	250-foo.com greets bar.com	
S	250-8 BITMIME	Сервер сообщает о поддерживаемых расширениях ESMTP
S	250-SIZE	
S	250-DSN	
S	250 HELP	
C	MAIL FROM:<Smith@bar.com>	Адрес отправителя: Smith@bar.com
S	250 OK	
C	RCPT TO:<Jones@foo.com>	Адрес первого получателя: Jones@foo.com
S	250 OK	
C	RCPT TO:<Green@foo.com>	Адрес второго получателя: Green@foo.com
S	550 No such user here	Ошибка: ящик не существует

C	RCPT TO:<Brown@foo.com>	Адрес третьего получателя: Brown@ foo.com
S	250 OK	
C	DATA	Адреса переданы, клиент готов передавать сообщение
S	354 Start mail input; end with <CRLF>.<CRLF>	Сервер готов к приему сообщения
C	Клиент передает сообщение	Сообщение заканчивается строкой, состоящей из одной точки
C	.	
S	250 OK	Сообщение принято
C	QUIT	Клиент завершает связь
S	221 foo.com Service closing transmission channel	Сервер подтверждает завершение связи

3.4 Расширения ESMTP

Механизм расширений **ESMTP** позволяет дополнять протокол **SMTP** новыми функциональными возможностями, не предусмотренными в RFC 2821 . Расширения могут добавлять к протоколу **SMTP** новые функции или модифицировать существующие. При этом должна сохраняться обратная совместимость: функции базового протокола **SMTP** должны выполняться независимо от установленных расширений.

Многие расширения **ESMTP** описаны в документах **RFC** . Их мы рассмотрим ниже. Разработчики программного обеспечения также могут использовать в своих продуктах нестандартизованные расширения. Естественно, работать с ними могут только программы того же производителя. Чтобы не допустить ситуации, при которой новое стандартное расширение получит название, которое уже было использовано каким-либо производителем, названия таких расширений должны начинаться с буквы X.

Название стандартного расширения с буквы X начинаться не может.

Расширения **ESMTP** могут добавлять новые команды, не предусмотренные базовым протоколом **SMTP** , а также вводить дополнительные параметры команд MAIL и RCPT . Формат дополнительных параметров:

Название_параметра=аргумент

Клиент узнает, какие именно расширения поддерживаются сервером, из ответа на команду EHLO . Каждая строка ответа может содержать ключевое слово, соответствующее названию поддерживаемого сервером расширения **ESMTP** , и, если необходимо, параметры этого расширения.

4.1 8BITMIME – передача текста сообщения восьмибитным кодом

В базовом протоколе **SMTP** не предусмотрена передача текста сообщения в кодировке, отличной от us – ascii . В сообщении могут использоваться только символы, кодируемые семью битами. Если при передаче используется восьмибитный транспорт, старший бит должен сбрасываться в ноль. Это делает невозможной передачу по протоколу **SMTP** сообщений на языках, использующих не только латинские символы, без перекодирования их в семибитный код.

В реальных сетях это правило обычно игнорируется, но, чтобы быть уверенным в результате, стандарты лучше не нарушать.

RFC 1652 . Хотя, и этим правилом, к сожалению, часто пренебрегают на практике.

Это расширение не отменяет ограничения на длину строки сообщения – 1000 символов, потому использоваться оно может только для передачи текста. Для пересылки двоичных вложений предусмотрены другие механизмы.

Если сервер **SMTP** поддерживает это расширение, то в ответе на команду EHLO должна быть передана строка вида:

250 8BITMIME

Эта строка сообщает клиенту, что сервер поддерживает прием сообщений, в теле которых используются восьмибитные символы. Если ответ на команду EHLO не содержит такой строки, клиент должен преобразовать сообщение таким образом, чтоб оно содержало только семибитные символы. В противном случае возможно искажение информации.

Расширение модифицирует команду MAIL , вводя в нее дополнительный параметр BODY , аргумент которого может принимать следующие значения:

8BITMIME – тело сообщения от указанного отправителя содержит восьмибитные символы, все биты сообщения должны передаваться без изменений;

7BIT – сообщение содержит только символы в кодировке us – ascii , при передаче по восьмибитному каналу старший бит каждого октета должен быть установлен в ноль (принято по умолчанию).

Диалог **SMTP** с использованием описанного расширения выглядит таким образом:

S	220 dbc.mtview.ca.us **SMTP** service ready	
C	EHLO ymir.edu	
S	250-dbc.mtview.ca.us says hello	Сервер поддерживает расширение 8 BITMIME
S	250 8BITMIME	

C	MAIL FROM:<ned@ymir.edu> BODY=8BITMIME	Модифицированная команда MAIL . Тело сообщения содержит восьмибитные символы
S	250 <ned@ymir.edu>… Sender and 8BITMIME ok	
C	RCPT TO:<mrose@dbc.mtview.ca.us>	
S	250 <mrose@dbc.mtview.ca.us>… Recipient ok	
C	DATA	
S	354 Send 8BITMIME message, ending in **CRLF**.CRLF.	
C	Текст сообщения.	
C	.	
S	250 OK	
C	QUIT	
S	250 Goodbye	

4.2 CHECKPOINT — продолжение передачи сообщения после обрыва связи

Если во время передачи сообщения происходит обрыв связи, клиент повторно устанавливает соединение с сервером и посылает то же сообщение с начала.

Чтобы избежать повторной передачи уже переданной части сообщения, RFC 1845 вводит расширение **ESMTP** , позволяющее принимать сообщение, начиная с того места, до которого оно было передано прежде, чем оборвалась связь.

Если сервер поддерживает это расширение, то в ответе на команду EHLO появляется строка вида:

250 CHECKPOINT

Расширением модифицируется команда MAIL . К ней добавляется новый параметр TRANSID, аргументом которого служит заключенный в угловые скобки уникальный идентификатор сообщения. Он состоит из некоторой последовательности символов, сгенерированной клиентом, символа @ и имени домена, обычно это доменное имя клиента. Таким образом, в идентификаторе сообщения используется тот же синтаксис, что и в адресе электронной почты. Общая длина идентификатора не должна превышать 80 символов. Идентификатор используется для обнаружения прерванной транзакции при повторной передаче сообщения.

Если в процессе передачи связь обрывается, принятая часть сообщения сохраняется. Рекомендуется хранить ее как минимум 48 часов, ожидая повторной попытки клиента передать это сообщение. Когда клиент вновь устанавливает соединение, он передает в поле TRANSID тот же идентификатор, что и при предыдущем, разорванном сеансе связи. Сервер находит начало соответствующего сообщения и посылает ответ с кодом 355, в котором сообщает, начиная с какого октета следует продолжать передачу.

Диалог **SMTP** с использованием этого расширения принимает следующий вид:

S	220 dbc.mtview.ca.us **SMTP** service ready	
C	EHLO ymir.edu	
S	250-dbc.mtview.ca.us says hello	Сервер поддерживает расширение CHECKPOINT
S	250 CHECKPOINT	
C	MAIL FROM:<ned@ymir.edu> TRANSID=<12345@ymir.edu>	Уникальный идентификатор сообщения: 12345@ymir . edu

S	250 <ned@ymir.edu>… Sender and TRANSID ok	
C	RCPT TO:<mrose@dbc.mtview.ca.us>	
S	250 <mrose@dbc.mtview.ca.us>… Recipient ok	
C	DATA	
S	354 Send checkpointed message, ending in **CRLF**.CRLF	

Далее передается текст сообщения до тех пор, пока связь не обрывается. Через некоторое время соединение снова устанавливается и происходит следующий диалог:

S	220 dbc.mtview.ca.us **SMTP** service ready	
C	EHLO ymir.edu	
S	250-dbc.mtview.ca.us says hello	
S	250 CHECKPOINT	
C	MAIL FROM:<ned@ymir.edu> TRANSID=<12345@ymir.edu>	В поле команды MAIL передается тот же идентификатор
S	355 6135 is the transaction offset	Сервер обнаружил начало сообщения. В ходе прошлого сеанса было принято 6135 октетов
C	DATA	
S	354 Send previously checkpointed message starting at octet	
C	Информация передается, начиная с октета 6136. Передача заканчивается как обычно строкой, состоящей из одной точки	
C	.	
S	250 OK	
C	QUIT	
S	221 Goodbye	

4.3 SIZE — объявление размера сообщения

Прием сообщений большого размера часто приводит к переполнению дискового пространства, выделенного для хранения писем. Чтобы избежать этой проблемы многие администраторы ограничивают максимально допустимый объем принимаемого сообщения и общий объем почтового ящика. Но, поскольку размер принимаемого сообщения заранее не известен, сервер вынужден принимать любое сообщение до конца, и только после этого он определяет, может ли оно быть принято или его следует отвергнуть из-за недостатка места на диске или из-за превышения допустимых лимитов.

Механизм, предложенный в RFC 1870 , позволяет избежать бесполезного расходования ресурсов на получение сообщений большого размера, которые все равно будут стерты сразу после приема. Он позволяет заранее проинформировать клиента о максимально допустимом размере сообщения, а сервер о размере сообщения, которое ему предстоит принять.

Ответ сервера, поддерживающего такое расширение, на команду EHLO имеет следующий вид:

250 SIZE число

где число — необязательный параметр, максимальный размер принимаемого сообщения в октетах. Сообщения большего размера будут отвергнуты.

Расширение модифицирует команду MAIL , добавляя к ней параметр SIZE. Его аргументом служит размер подготовленного к отправке сообщения. При его определении учитываются заголовок и тело, пустая строка между ними и все символы **CRLF** . Не учитываются точки, которые добавляются на передающей стороне, чтобы избежать ложного обнаружения конца сообщения; и завершающая строка, состоящая из одной точки. Желательно, чтобы аргумент параметра SIZE отражал истинный размер сообщения или хотя бы был не меньше.

Получив сведения о размере сообщения, сервер еще до получения самого письма может решить, будет ли он его обрабатывать. Если на диске недостаточно места для сохранения сообщения, сервер отвечает кодом 452. В этом случае клиент может повторить попытку позже, когда место, возможно, освободится. Если размер сообщения превышает допустимый лимит, сервер может отвергнуть это сообщение кодом 552. Такое сообщение не будет доставлено и вернется к отправителю с соответствующей пометкой.

Ограничения по размеру сообщений и объему ящиков могут быть наложены не только на весь сервер, но и на ящики отдельных пользователей. В этом случае после положительного ответа на команду MAIL возможны отказы в ответ на команду RCPT . Ниже приведен пример диалога **SMTP** , при котором сообщение в целом принимается, но отвергается для некоторых адресатов.

S	220 sigurd.innosoft.com – Server SMTP	
C	EHLO ymir.Claremont.edu	
S	250-sigurd.innosoft.com	
S	250-EXPN	
S	250-HELP	
S	250 SIZE 1000000	Сервер поддерживает расширение SIZE . Максимальный допустимый размер принимаемого сообщения: 1000000 байт.
C	MAIL FROM:<ned@innosoft.com> SIZE=500000	Размер передаваемого сообщения: 500000 байт, что не превышает установленный лимит
S	250 Address Ok.	
C	RCPT TO:<ned@innosoft.com>	Сообщение может быть доставлено получателю
S	250 OK; can accomodate 500000 byte message	

C	RCPT TO:<ned@ymir.claremont.edu>	Получатель не может принимать сообщения такого размера
S	552 Channel size limit exceeded	
C	RCPT TO:<ned@hmcvax.claremont.edu>	Доставка сообщения временно невозможна. Почтовый ящик переполнен.
S	452 Insufficient channel storage	
C	DATA	
S	354 Send message, ending in **CRLF**.CRLF.	
C	…	
C	.	
S	250 Some recipients OK	Сообщение может быть доставлено не всем получателям
C	QUIT	
S	221 Goodbye	

4.4 ETRN — получение сообщений из удаленной очереди

Если сервер **SMTP** не подключен к сети постоянно, то его очередь входящих сообщений может временно храниться на промежуточном сервере. Оконечный сервер может периодически соединяться с ним и забирать почту. Использование при этом описанной выше команды TURN потенциально небезопасно, если клиент не проходит процедуру обязательной аутентификации. Злоумышленник может подключиться к промежуточному серверу и, сообщив о себе неверные данные, получить почту, ему не предназначенную. Чтобы предотвратить это, RFC 1985 рекомендует вместо того, чтобы использовать для передачи почты тот же сеанс, открывать новое соединение **SMTP** по инициативе промежуточного сервера.

Получение почты оконечным сервером в этом случае происходит

таким образом: оконечный сервер соединяется с промежуточным сервером по протоколу **SMTP**и запрашивает прием почты; после этого промежуточный сервер устанавливает соединение с оконечным сервером и передает накопившиеся для этого сервера сообщения.

Ответ сервера, поддерживающего данное расширение, на команду EHLO имеет следующий вид:

250 ETRN

Вводится дополнительная команда ETRN. Ее формат:

ETRN доменное_имя_оконечного_сервера **CRLF**

Команда может быть выполнена в любое время, но не в процессе подготовки и отправки сообщения, то есть после успешно выполненной командой MAIL и до завершения выполнения команды DATA команда ETRN не может быть выполнена.

Пример использования команды ETRN:

S	220 sigurd.innosoft.com	
C	EHLO ymir.Claremont.edu	
S	250-sigurd.innosoft.com	
S	250-EXPN	
S	250-HELP	
S	250 ETRN	Сервер поддерживает расширение ETRN
C	ETRN	Команда ETRN требует в качестве аргумента полностью определенное доменное имя оконечного сервера
S	500 Syntax Error	
C	ETRN localname	
S	501 Syntax Error in Parameters	

C	ETRN innosoft . com	Команда выполнена успешно. Промежуточный сервер установит соединение с оконечным сервером и передаст ему накопившуюся почту
S	250 OK, queuing started	
C	ETRN sigurd.innosoft.com	Команда принята, но почты для передачи оконечному серверу нет
S	251 OK, no messages waiting	
C	ETRN mysoft.com	Для оконечного сервера принято 14 сообщений, они будут переданы по соединению, которое будет установлено по инициативе промежуточного сервера
S	253 OK, 14 pending messages	
C	ETRN foo.bar	В данный момент команда не может быть выполнена: не удалось найти оконечный сервер
S	459 Unable to resolve name.	
C	QUIT	
S	250 Goodbye	

4.5 ENHANCEDSTATUSCODES — расширенные коды ответов

RFC 3463 описывает расширенные коды ответов сервера **SMTP** , позволяющие представить ответы сервера в более понятном для машины виде, чем обычные коды ответов.

Расширение **ESMTP** , описанное в RFC 2034 , позволяет использовать такие коды.

Сервер, поддерживающий это расширение должен посылать клиенту

в ответе на команду EHLO строку вида:

250 ENHANCEDSTATUSCODES

После этого в ответах, посылаемых сервером клиенту, после обычных трехзначных кодов ответов появляются расширенные коды, состоящие из трех чисел, разделенных точкой, и сопровождаемые текстом, поясняющим значение расширенного ответа. Здесь мы не будем приводить описания расширенных кодов: их довольно много, они описаны в RFC 3463 .

Пример использования расширенных кодов ответов:

S	220 dbc.mtview.ca.us **SMTP** service ready
C	EHLO ymir.claremont.edu
S	250-dbc.mtview.ca.us says hello
S	250 ENHANCEDSTATUSCODES
C	MAIL FROM:<ned@ymir.claremont.edu>
S	250 2.1.0 Originator <ned@ymir.claremont.edu> ok
C	RCPT TO:<mrose@dbc.mtview.ca.us>
S	250 2.1.5 Recipient <mrose@dbc.mtview.ca.us> ok
C	RCPT TO:<nosuchuser@dbc.mtview.ca.us>
S	550 5.1.1 Mailbox “nosuchuser” does not exist
C	RCPT TO:<remoteuser@isi.edu>
S	551-5.7.1 Forwarding to remote hosts disabled
S	551 5.7.1 Select another host to act as your forwarder
C	DATA
S	354 Send message, ending in **CRLF**.CRLF.
C	……..
C	.
S	250 2.6.0 Message accepted
C	QUIT
S	221 2.0.0 Goodbye

4.6 AUTH – аутентификация и шифрование

Базовый протокол **SMTP** не предусматривает аутентификацию отправителя или **MTA**, что часто используется злоумышленниками, желающими скрыть или фальсифицировать авторство сообщения. Еще один существенный недостаток протокола **SMTP** заключается в том, что вся информация передается открытым текстом, может быть перехвачена при передаче и использована злоумышленниками.

Расширение **ESMTP** AUTH, описанное в RFC 2554 , для аутентификации клиента и отправителя и, если необходимо, для шифрования передаваемой информации.

Все узлы, использующие расширение AUTH для аутентификации друг друга, должны хранить пароли других узлов. Смена этой информации должна отразиться на всех остальных узлах, причем передача новых паролей не должна происходить по открытым каналам, так как они могут быть перехвачены злоумышленником. Это ограничивает возможности использования расширения AUTH в глобальной сети. Обычно его используют для взаимодействия между почтовыми узлами одной почтовой системы.

Например, весьма целесообразно использовать этот способ аутентификации на участке между **MUA** отправителя и **MSA** или **MTA**, принимающим почту от пользовательских агентов.

Строка ответа на команду EHLO , соответствующая этому расширению, имеет вид:

250 AUTH параметры

в качестве параметров передаются разделенные пробелами ключевые слова, соответствующие механизмам аутентификации, поддерживаемым сервером.

Например :

250 AUTH CRAM-MD5 DIGEST-MD5

Вводится дополнительная команда AUTH. Ее формат:

AUTH механизм_аутентификации первая_строка_диалога **CRLF**

Команда сообщает серверу механизм аутентификации, используемый клиентом. Если, в соответствии с этим механизмом, первая строка диалога должна посылаться клиентом серверу, то ее тоже можно передать в команде AUTH. Если предложенный механизм поддерживается, то начинается обмен аутентификационной информацией, закодированной по алгоритму Base 64 (RFC 3548). Ответы сервера имеют код 334. Если первая строка не была передана в команде AUTH, то ответ сервера на эту команду будет состоять только из кода 334.

Пример :

C	AUTH CRAM-MD5
S	334 PENCeUxFREJoU0NnbmhNWitOMjNGNndAZWx3b29kLmluLmNvbT4=
C	ZnJlZCA5ZTk1YWVlMDljNDBhZjJiODRhMGMyYjNiYmFlNzg2ZQ==
S	235 Authentication successful.

Если клиенту нужно прервать диалог, он посылает серверу строку, состоящую из одной звездочки (*), сервер возвращает сообщение об ошибке 501 и ожидает следующей команды **SMTP** .

Если клиент и сервер договариваются шифровать передаваемую информацию, то дальнейший диалог клиента и сервера начинается с повторного выполнения команды EHLO .

За время одного сеанса **SMTP** допускается только одна успешно выполненная команда AUTH. Попытки повторить команду после ее успешного завершения должны отвергаться.

Если команда AUTH используется для аутентификации клиента, то дополнительный параметр AUTH команды MAIL используется для аутентификации отправителя, то есть, он позволяет убедиться в том, что сообщение отправляется действительно владельцем почтового ящика с указанным в команде MAIL адресом.

Аргументом параметра AUTH команды MAIL является идентификатор, под которым был аутентифицирован пользователь, обычно это

адрес его электронной почты. Этот идентификатор не может содержать ряд символов, таких, как, например, пробелы, плюсы (+), равно (=) или русские буквы. Если в идентификаторе содержатся такие символы, он должен быть перекодирован. Каждый запрещенный символ заменяется последовательностью, состоящей из плюса (+) и шестнадцатиричного кода заменяемого символа. Пример команды MAIL с параметром AUTH:

MAIL FROM:<e=mc2@example.com> AUTH=e+3Dmc2@example.com

Если параметр AUTH используется с пустым аргументом:

AUTH =<>

это свидетельствует о том, что аутентификация отправителя по какой-то причине не состоялась.

Параметр AUTH может рассматриваться как достоверный, только если была успешно выполнена команда AUTH . В противном случае его следует интерпретировать так же, как если бы его параметр был равен <>: аутентичность отправителя не подтверждена.

4.7 STARTTLS – передача данных по защищенному каналу с использованием сертификатов

Расширение **ESMTP** STARTTLS, описанное в RFC 3207 . Шифрование происходит на основе сертификатов, с использованием открытых и закрытых ключей.

На основании сертификатов также происходит аутентификация встречной стороны: узел не может вести диалог, пользуясь чужим сертификатом. Но такой способ аутентификации не позволяет проверить подлинность передаваемого сообщения: аутентифицируются только узлы, а не пользователи. Однако, после выполнения команды STARTTLS и перехода в защищенный режим, можно произвести аутентификацию с помощью команды AUTH . Аутентификационные данные можно передавать открытым текстом, так как все данные после успешного завершения команды STARTTLS

все равно шифруются.

В ответе сервера, поддерживающего это расширение **ESMTP** , на команду EHLO должна присутствовать строка:

250 STARTTLS

Убедившись, что сервер поддерживает это расширение, клиент посылает ему команду STARTTLS. Ее формат:

STARTTLS **CRLF**

У команды STARTTLS нет параметров.

Предусмотрены три варианта ответа сервера:

220 Готов к обмену данными **TLS**

501 Синтаксическая ошибка (использование параметров не предусмотрено)

454 **TLS** временно недоступна

В случае отказа, клиент должен, основываясь на установленных администратором правилах, принять решение, продолжать передачу открытым текстом, повторить попытку позже или отказаться от передачи сообщения, послав соответствующее уведомление отправителю.

Серверам, предназначенным для приема электронной почты из глобальной сети, нельзя требовать от клиентов обязательного использования протокола **TLS** , иначе их пользователи не смогут получать почту от серверов **SMTP** , не поддерживающих данное расширение. Однако для серверов, обслуживающих внутреннюю почтовую сеть предприятия, использование протокола **TLS** может быть обязательным. На любую команду клиента, кроме NOOP, EHLO, STARTTLS и QUIT, такой сервер может отвечать:

530 Выполните команду STARTTLS

Серверы исходящей почты также могут отказаться отправлять сообщения на серверы, не поддерживающие **TLS** .

В случае положительного ответа сервера на команду STARTTLS , клиент начинает обмен данными по протоколу **TLS** . Установив защищенное соединение, клиент и сервер принимают решение, достаточен ли им согласованный уровень безопасности. Если для клиента этот уровень не достаточен, он прекращает соединение, передав серверу команду QUIT. Если предложенный уровень не устраивает сервер, то на все дальнейшие команды он будет сообщать об ошибке с кодом 554 – по соображениям безопасности команда не может быть выполнена.

Клиент может отказаться продолжать диалог с сервером, чей сертификат не подтверждает доменное имя вызываемого сервера. Серверу, принимающему входящую почту, следует продолжать диалог с клиентом независимо от доменного имени, удостоверенного предъявленным сертификатом, но информацию о сертификате следует включать в заголовок почтового сообщения.

Приняв решение о продолжении диалога, клиент и сервер начинают обмен данными по протоколу **SMTP** с самого начала, с команды EHLO , причем ответ сервера на эту команду может отличаться от того, который он давал до установления защищенного соединения. Списки расширений, поддерживаемые сервером при защищенном и при незащищенном соединении, могут различаться. В любом случае, расширение STARTTLS после начала защищенного диалога поддерживаться не должно.

Используя шифрование, нужно учитывать, что на пути к получателю почта может проходить через множество серверов **SMTP** , и нельзя быть уверенным, что все эти сервера поддерживают шифрование. Таким образом, сообщение, посылаемое через Internet , может часть пути проделать по защищенном соединениям, а остальную часть пройти открытым текстом. На этих участках оно может быть перехвачено злоумышленником.

Пример диалога **SMTP** с использованием расширения STARTTLS:

C	EHLO mail.example.com	
S	250-mail.imc.org welcome	

S	250-8BITMIME	
S	250- STARTTLS	Сервер поддерживает расширение STARTTLS
S	250 DSN	
C	STARTTLS	Клиент запрашивает защищенное соединение
S	220 Go ahead	Сервер готов к установлению защищенного соединения
	Клиент и сервер обмениваются сертификатами, согласовывают уровень защиты информации и принимают решение о продолжении диалога	
C	EHLO mail.example.com	Диалог по защищенному соединению начинается с повторного выполнения команды EHLO
S	250-mail.imc.org touches your hand	Ответ на команду EHLO по защищенному соединению отличается от предыдущего ответа. Расширение STARTTLS не поддерживается, так как повторное установление защищенного соединения не имеет смысла.
S	250-8BITMIME	
S	250 DSN	

4.8 ATRN — передача почты по запросу

Выше рассматривалась команда **ESMTP** ETRN , используемая взамен устаревшей команды TURN . Использование этого расширения возможно только если оконечный сервер имеет постоянный адрес **IP** и соответствующую ему запись в **DNS** . Однако многие узлы, не имеющие постоянного подключения к сети, получают адрес только временно и записи в **DNS** не имеют.

RFC 2645 предлагает расширение **ESMTP** , вводящее новую команду ATRN , аналог команды TURN , который однако может быть использован только после успешной аутентификации клиента по команде AUTH . Строка ответа сервера на команду EHLO , свидетельствующая о поддержке команды ATRN :

250 ATRN

Формат команды:

ATRN список_доменов **CRLF**

В отличие от команды TURN , команда ATRN принимает в качестве параметров список почтовых доменов, разделенных запятыми. Если команда вызывается без параметров, оконечному серверу посылается почта для всех доменов, к которым он имеет доступ. Если доступ к почте хотя бы одного домена из списка параметров команды ATRN не разрешен, то передача почты ни для одного из них не осуществляется, команда отвергается с кодом ошибки 450.

Рассмотрим пример выполнения команды ATRN . Поскольку клиент и сервер меняются в процессе диалога ролями, участники диалога обозначены здесь как вызывающий (A) и вызываемый (B).

B	220 EXAMPLE.NET mail relay server ready	Вызываемый узел EXAMPLE . NET готов к приему команд
A	EHLO example.org	Вызывающий узел — example . org
B	250-EXAMPLE.NET	Вызываемый узел поддерживает команды AUTH и ATRN
B	250-AUTH CRAM-MD5 EXTERNAL	
B	250 ATRN	
A	AUTH CRAM-MD5	
B	334 MTg5Ni42OTcxNzA5NTJVNQLkNPTQo=	

A	Zm9vYmFyLm5ldCBiOTGU2MzRkMzg5MAo=	example.org успешно аутентифицирован
B	235 now authenticated as example.org	
A	ATRN example.org,example.com	Смена направления передачи. Теперь example . org стал сервером
B	250 OK now reversing the connection	
A	220 example.org ready to receive email	
B	EHLO EXAMPLE.NET	EXAMPLE . NET стал клиентом
A	250-example.org	
A	250 SIZE	
B	MAIL FROM: <Lester.Tester@dot.foo.bar>	Почта от EXAMPLE . NET передается на example . org
A	250 OK	
B	RCPT TO: <l.eva.msg@example.com>	
A	250 OK, recipient accepted	
	…	
B	QUIT	
A	221 example.org closing connection	

4.9 DSN — оповещения о ходе доставки

Протокол **SMTP** предусматривает посылку отправителю сообщений о проблемах при доставке корреспонденции. Однако часто таких сообщений бывает недостаточно. RFC 3461 описывает расширение **ESMTP DSN** (Delivery Status Notifications), увеличивающее возможности имеющейся услуги оповещений о ходе

доставки.

Строка ответа на команду EHLO , свидетельствующая о том, что сервер поддерживает это расширение, имеет вид:

250 **DSN**

Вводятся два дополнительных параметра команды MAIL:

- параметр RET. Если RET=FULL, то в оповещение о неуспешной доставке будет включено сообщение целиком, если RET= HDRS, то в оповещение будет включен только заголовок. Если параметр не используется, то сервер может включать в оповещение о неудачной доставке как все сообщение, так и только заголовок. Действие параметра распространяется только на оповещения о неудаче. Оповещения об успешной доставке всегда включают в себя только заголовок сообщения;
- параметр ENVID. Аргументом для этого параметра служит некий уникальный идентификатор конверта – произвольная последовательность букв, цифр и знаков препинания кроме “=” и “+”. Тот же идентификатор должен быть использован и в уведомлении, что позволит отправителю легко определить, к какому именно сообщению относится данное уведомление.

Вводятся два дополнительных параметра команды RCPT:

- параметр NOTIFY. Аргументы параметра NOTIFY указывают, в каких случаях надо посылать уведомление. Если аргументов несколько, они разделяются запятыми. Значения аргументов: SUCCESS (сообщение доставлено), FAILURE (сообщение не доставлено), DELAY (сообщение не могло быть отправлено дольше некоторого значения времени ожидания, установленного на **MTA**, где произошла задержка – обычно 4 часа). Если уведомление вообще не должно посылаться, аргумент принимает значение NEVER;
- параметр ORCPT. Аргумент параметра ORCPT состоит из двух частей, разделенных точкой с запятой (;). В первой части

указывается тип адреса получателя – обычно ” rfc822″. Во второй части приводится сам адрес получателя, при необходимости перекодированный по правилу, описанному выше, в разделе, посвященном расширению **ESMTP** AUTH . Серверы **SMTP** должны пересылать значение этого параметра без изменений даже в регистре символов. То есть, даже если адрес получателя изменится в процессе передачи, например, при раскрытии списка рассылки или почтового псевдонима, значение ORCPT не изменится и будет отражено в посылаемом отправителю уведомлении вместе с адресом, на который сообщение на самом деле должно было поступить. Это уведомление может быть использовано для отладки и для выяснения причин проблем, возникающих при передаче.

Ниже приведен пример диалога **SMTP** , в процессе которого нескольким пользователям отправляется сообщение с различными аргументами параметра NOTIFY. В данном примере из-за нехватки места некоторые команды разбиты на две строки. На самом деле, каждая команда, естественно, передаются как одна строка.

S	220 Example.ORG **SMTP** server here
C	EHLO Example.ORG
S	250-Example.ORG
S	250-DSN
S	250-EXPN
S	250 SIZE
C	MAIL FROM:<Alice@Example.ORG> RET=HDRS ENVID=QQ314159
S	250 <Alice@Example.ORG> sender ok
C	RCPT TO:<Bob@Example.COM> NOTIFY=SUCCESS ORCPT=rfc822;Bob@Example.COM
S	250 <Bob@Example.COM> recipient ok
C	RCPT TO:<Carol@vory.EDU> NOTIFY=FAILURE ORCPT=rfc822;Carol@Ivory.EDU

S	250 <Carol@Ivory.EDU> recipient ok
C	RCPT TO:<Dana@Ivory.EDU> NOTIFY=SUCCESS,FAILURE ORCPT=rfc822;Dana@Ivory.EDU
S	250 <Dana@Ivory.EDU> recipient ok
C	RCPT TO:<Fred@Bombs.AF.MIL> NOTIFY=NEVER
S	250 <Fred@Bombs.AF.MIL> recipient ok
C	DATA
S	354 okay , send message
C	Передается сообщение
C	.
S	250 message accepted
C	QUIT
S	221 goodbye

4.10 DELIVERBY – управление временем доставки сообщения

Расширение **ESMTP** , описанное в RFC 2852 позволяет отправителю определять, в какой срок должна быть осуществлена доставка сообщения и как следует поступить с сообщением, которое не удалось своевременно доставить.

Строка ответа на команду EHLO , свидетельствующая о том, что сервер поддерживает это расширение, имеет вид:

250 DELIVERBY параметр

Необязательный параметр содержит минимальное время в секундах, допустимое в качестве аргумента параметра BY – дополнительного параметра команды MAIL, вводимого данным расширением. Его аргумент состоит из числа, соответствующего времени в секундах, в течение которого сообщение должно быть доставлено. После числа следует отделенный точкой с запятой (;) модификатор, указывающий на то, какие действия следует предпринять в случае, если произвести доставку в указанный

срок невозможно. Модификатор состоит из одной буквы и может принимать следующие значения:

N – отправителю посылается уведомление, сообщение доставляется;

R – сообщение стирается. Если отправитель не отключил уведомления о неудачной доставке и задержке с помощью расширения **DSN** , то ему посылается уведомление.

Дополнительно на конце аргумента параметра BY может стоять модификатор “T”, означающий, что каждый **MTA**, через который будет проходить сообщение, должен будет послать отправителю уведомление о пересылке сообщения для каждого адресата, для которого не определен параметр NOTIFY или значение этого параметра не NEVER. С помощью этого модификатора можно выяснять структуру почтовой сети, потому использовать это расширение следует с осторожностью.

Ниже приведен фрагмент диалога **SMTP** с использованием расширения DELIVERBY:

C	EHLO acme.net	Сервер поддерживает расширение DELIVERBY . Время доставки, выставляемое в параметре BY должно быть не меньше 30 секунд.
S	250-mail.other.com	
S	250 DELIVERBY 30	
C	MAIL FROM:<eljefe@bigbiz.com> BY=98;R	Сообщение следует доставить не более, чем за 98 секунд. Если письмо задержится дольше, его следует удалить, оповестив отправителя.

S	250 <eljefe@bigbiz.com> Sender okay	

4.11 PIPELINING – группирование команд

Согласно RFC 821 механизм группирования команд позволяет уменьшить время ожидания.

Предложенное расширение **ESMTP** не предусматривает ожидание ответа на команды, передаваемые в одной дейтаграмме **TCP**. Если клиент в ответе на команду EHLO получает строку

250 PIPELINING

то он может одной дейтаграммой посылать группу команд RSET, MAIL, SEND, SOML, SAML и RCPT, не дожидаясь ответа на каждую из них. Группа может заканчиваться одной из не перечисленных команд, например, командой NOOP, которую можно использовать как разделитель групп.

Сервер отвечает, только получив одну из не перечисленных команд или если на входе нет больше ни одной команды. Ответы он посылает в том же порядке, в каком получал команды, потому клиент может легко различить, какой ответ относится к какой команде.

Диалог клиента и сервера, поддерживающих расширение PIPELINING, приобретает такой вид:

C	EHLO dbc.mtview.ca.us
S	250-innosoft.com
S	250 PIPELINING
C	MAIL FROM:<mrose@dbc.mtview.ca.us>
C	RCPT TO:<ned@innosoft.com>
C	RCPT TO:<dan@innosoft.com>
C	RCPT TO:<kvc@innosoft.com>
C	DATA

S	250 sender <mrose@dbc.mtview.ca.us> OK
S	250 recipient <ned@innosoft.com> OK
S	250 recipient <dan@innosoft.com> OK
S	250 recipient <kvc@innosoft.com> OK
S	354 enter mail, end with line containing only “.”
C	Передается сообщение
C	.
C	QUIT
S	250 message sent
S	221 goodbye

Горизонтальные линии разделяют блоки информации, которые могут быть переданы за один раз – одной дейтаграммой **TCP**.

Как видим, в данном примере семь команд, текст сообщения и восемь ответов сервера могут быть переданы всего шестью дейтаграммами **TCP**.

4.12 CHUNKING и BINARYMIME – передача двоичных файлов большого объема

Здесь уже несколько раз упоминалась проблема, связанная с тем, что протокол **SMTP** изначально был рассчитан только на пересылку коротких текстовых сообщений на английском языке. Выше был описан один из путей решения этой проблемы – расширение 8BITMIME, но это расширение все же предназначено для передачи текстов, так как оно сохраняет ограничение на длину строки. Основным способом передачи двоичных файлов по электронной почте является их кодирование в семибитные последовательности. Но кодирование приводит к увеличению размера сообщения и к росту нагрузки на серверы, которые должны производить кодирование и декодирование файлов. Особенно заметными становятся эти проблемы при передаче сообщений большого объема.

RFC 3030 описывает два расширения **ESMTP** , предназначенные для

передачи двоичных файлов большого объема. Первое расширение вводит новую команду BDAT , заменяющую команду DATA . Ответ сервера, поддерживающего это расширение, на команду EHLO имеет вид:

250 CHUNKING

Команда BDAT позволяет передавать двоичные файлы любого формата. Причем передавать большой файл можно по частям. Сервер может сообщить о неполадках, только когда передача файла уже закончена. Потому для экономии ресурса сети большие файлы лучше передавать по частям. Формат команды:

BDAT число **CRLF**

где число – объем передаваемой информации в октетах. Указание размера передаваемого блока данных позволяет серверу принимать информацию, не проверяя ее на наличие строки, состоящей из одной точки. Прием заканчивается, когда принято указанное число октетов.

Ответ сервера на команду BDAT не предусмотрен. Передав команду BDAT , клиент сразу начинает передачу. Получив информацию, сервер посылает ответ. Если ответ положительный, то клиент может продолжить передачу.

Поскольку серверу не известен общий объем передаваемого файла, клиент должен сообщить, что передача закончена. Последняя команда BDAT имеет формат:

BDAT число LAST **CRLF**

Последняя команда BDAT может не передавать данные. В этом случае она принимает ноль в качестве первого параметра и используется только для обозначения конца сообщения.

Также RFC 3030 вводит расширение, позволяющее использовать параметр BODY команды MAIL , который также использует расширение 8BITMIME, описанное выше. О поддержке этого расширения свидетельствует строка следующего вида в ответе

сервера на команду EHLO :

250 BINARYMIME

Если это расширение поддерживается, аргумент параметра BODY может принимать значение BINARYMIME. Использование расширения BINARYMIME возможно только совместно с расширением CHUNKING. Передавать сообщения в формате BINARYMIME можно только при помощи команды BDAT , использование команды DATA не допускается. На данные в формате BINARYMIME не налагаются никакие ограничения: они могут содержать произвольную двоичную последовательность.

Пример диалога **SMTP** с использованием расширений BDAT и BINARYMIME :

S	220 cnri.reston.va.us **SMTP** service ready
C	EHLO ymir.claremont.edu
S	250-cnri.reston.va.us says hello
S	250-BINARYMIME
S	250 CHUNKING
C	MAIL FROM:<ned@ymir.claremont.edu> BODY=BINARYMIME
S	250 <ned@ymir.claremont.edu>… Sender and BINARYMIME ok
C	RCPT TO:<gvaudre@cnri.reston.va.us>
S	250 <gvaudre@cnri.reston.va.us>… Recipient ok
C	RCPT TO:<jstewart@cnri.reston.va.us>
S	250 <jstewart@cnri.reston.va.us>… Recipient ok
C	BDAT 100000
C	Передаются первые 100000 октетов сообщения …
S	250 100000 octets received
C	BDAT 324
C	Передаются следующие 324 октета …
S	250 324 octets received

C	BDAT 0 LAST
S	250 Message OK, 100324 octets received
C	QUIT
S	221 Goodbye

3.5 Тестирование сервера SMTP

Протокол **SMTP** прост в использовании, его легко можно тестировать вручную. Для этого нужно при помощи программы Telnet соединиться с сервером **SMTP** по 25 порту **TCP** . Чтобы видеть вводимые команды, следует включить отображение ввода.

После установления соединения, получив ответ сервера, можно начинать диалог, выступая в качестве клиента **SMTP** .

Некоторые клиентские почтовые программы позволяют отслеживать диалог, происходящий между клиентом и сервером.

Для наблюдения за трафиком какого-либо протокола, включая **SMTP** , идущим по сети, можно также использовать сниферы – программы, предназначенные для перехвата и анализа данных, проходящих по сети. С помощью сниферов можно проверять, насколько правильно и эффективно клиенты и серверы используют возможности сетевых протоколов, а также выявлять причины сбоев в работе сетевых служб.

3.6 Протокол LMTP

Описанный в RFC 2033 протокол **LMTP** применяется в основном для связи **MTA** с **LDA** . Он также может использоваться для взаимодействия с **MTA**, не помещающими сообщения в исходящую очередь, и имеющими возможность немедленно ответить, возможна доставка или нет. **LMTP** работает аналогично **SMTP** , использует расширения **ESMTP**, но область его применения отличается от области применения **SMTP** , и он не должен использовать порт **TCP** 25.

Протокол **LMTP** имеет только два отличия от протокола **SMTP** :

- команды HELO и EHLO заменяются командой LHLO, идентичной по синтаксису и действию команде EHLO;
- команды DATA и, если используется расширение **ESMTP** CHUNKING, BDAT LAST возвращают не один ответ после окончания приема сообщения, а столько, сколько было успешно выполненных команд RCPT . Сервер передает клиенту результат доставки сообщения каждому получателю.

Пример :

S	220 foo.edu **LMTP** server ready	
C	LHLO foo.edu	
S	250-foo.edu	
S	250-PIPELINING	
S	250 SIZE	
C	MAIL FROM:<chris@bar.com>	
S	250 OK	
C	RCPT TO:<pat@foo.edu>	
S	250 OK	
C	RCPT TO:<green@foo.edu>	
S	250 OK	
C	DATA	
S	354 Start mail input; end with <CRLF>.<CRLF>	
C	Передается сообщение	
C	.	
S	250 OK	Сообщение успешно доставлено первому адресату: pat @ foo . edu

S	452 <green@foo.edu> is temporarily over quota	Сообщение не доставлено получателю green @ foo . edu
C	QUIT	
S	221 foo.edu closing connection	

3.7 Контрольные вопросы

1. Для чего предназначен протокол **SMTP** ? На каких участках системы электронной почты он используется?
2. Какие команды протокола **SMTP** используются для отправки почтовых сообщений? Назовите последовательность, в которой эти команды выполняются.
3. В каком направлении происходит передача почтовых сообщений: от сервера к клиенту или от клиента к серверу?
4. Какие недостатки протокола **SMTP** вам известны? Каким образом эти недостатки устраняются?
5. Что такое расширения **ESMTP** ? В чем отличие команды EHLO от команды HELO ? Почему не рекомендуется использовать команду HELO? В каких случаях ее нужно использовать?
6. Какой ответ на команду EHLO ожидает от сервера клиент?
7. Как клиент анализирует ответы сервера? Что из себя представляют числовые коды ответов?
8. С помощью каких расширений **ESMTP** можно без дополнительного кодирования отправлять сообщения с телом на русском языке? В чем различия между форматами тела сообщения 7BIT, 8BITMIME и BINARYMIME ? Какие расширения **ESMTP** используются для передачи двоичных файлов без дополнительного кодирования? В чем отличие команды BDAT от команды DATA ? Для чего используется каждая из этих команд?
9. Какие вы знаете команды и расширения **ESMTP** для смены направления передачи почты? В каких случаях они используются? Опишите достоинства и недостатки каждой команды.

10. Какие расширения **ESMTP** используются для обеспечения безопасности электронной почты? За счет чего повышается безопасность?
11. Какие расширения **ESMTP** используются для отправки сообщений большого объема? Каким образом эти расширения повышают эффективность передачи?
12. Какое расширение **ESMTP** используется при отправке сообщений большому числу адресатов? За счет чего достигается повышение скорости передачи?
13. В чем отличия протоколов **LMTP** и **SMTP** ? Для чего используется протокол LMTP?

3.8 Практическое задание

1. При помощи программы Telnet соединитесь с сервером **SMTP**, принимающим почту для вашего ящика.
2. Выполните команду EHLO . Какие параметры следует передать с ней? Насколько они должны быть достоверны?
3. Какие расширения **ESMTP** поддерживает сервер?
4. Попробуйте проверить свой почтовый адрес с помощью команды VRFY. Какой код ответа вы получили? Что означает этот код?
5. Укажите произвольный адрес в качестве адреса отправителя. Какие адреса можно использовать? Каким образом сервер **SMTP** проверяет достоверность указанного адреса отправителя?
6. Укажите в качестве получателя адрес своего почтового ящика.
7. Передайте сообщение с помощью команды DATA.
8. Получив подтверждение о приеме сообщения, завершите сеанс связи командой QUIT.
9. С помощью клиентской почтовой программы проверьте, получили ли вы отправленное только что тестовое сообщение.

[источник]