МИНИСТЕРСТВО РОССИЙСКОЙ ФЕДЕРАЦИИ
ПО СВЯЗИ И ИНФОРМАТИЗАЦИИ
Санкт-Петербургский
государственный университет телекоммуникаций
им. проф. М.А. Бонч-Бруевича

О.С. Когновицкий, Е.М. Доронин, Л.М. Свердлов

СТРУКТУРА И ПРОТОКОЛЫ ЭЛЕКТРОННОЙ ПОЧТЫ В ИНТЕРНЕТ

УЧЕБНОЕ ПОСОБИЕ

200900, 220200, 220400

САНКТ-ПЕТЕРБУРГ
2004

УДК 681.322-181.4:681.3.06

Когновицкий О.С., Доронин Е.М., Свердлов Л.М. Структура и протоколы электронной почты в Интернет (спец. 200900, 220200, 220400): учеб. пособие / СПбГУТ. СПб, 2004.

Утверждено редакционно-издательским советом университета в качестве учебного пособия.

Рассматриваются действующие на сегодняшний день стандарты, описывающие структуру электронной почты в Интернет и возможности ее протоколов, а также примеры их использования. Представляет интерес для специалистов, занимающихся администрированием систем электронной почты.

Может быть использовано при выполнении курсовых и дипломных работ, а также при самостоятельной работе студентов.

Ответственный редактор *О.С. Когновицкий*
Рецензент к.т.н., доц. *О.Р. Рыкин* (СПбГПИ)

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Электронная почта – одна из самых популярных услуг, предоставляемых глобальными сетями компьютерной связи. Ее изначальная задача – доставка текстового сообщения от одного пользователя сети другому.

Для решения этой задачи в разное время использовались различные протоколы. Когда сети передачи данных были доступны только крупным учреждениям, а интеграция этих сетей и их услуг не была так велика как в последние годы, оборудование, необходимое для работы с электронной почтой, было дорого, и немногие пользователи могли себе позволить купить его, использовалась в основном электронная почта на базе протокола X.400.

Эта система доступна только крупным организациям, использует сложную адресацию, обеспечивает конфиденциальность и гарантирует доставку сообщения.

Те же пользователи, которые не могли себе позволить постоянное подключение к сетям передачи данных, использовали почту, построенную на базе протокола UUCP.

Этот протокол позволяет слать письма через цепочки серверов, часть из которых не имеет постоянного подключения к сети. Серверы могут связываться друг с другом по коммутируемым телефонным линиям и передавать накопившиеся со времени последнего сеанса связи сообщения. Время передачи сообщения по такой цепочке было довольно большим. В адресе получателя приходилось описывать весь маршрут, по которому должно пройти сообщение по сети, а маршрут зависел от того, откуда посылается письмо. Такая маршрутно-зависимая адресация была очень неудобна в использовании.

В начале 80-х гг. прошлого века была предложена новая система электронной почты, построенная на базе протокола SMTP. Впоследствии именно эта система стала стандартной электронной почтой для IP-сетей, а с распространением Интернет в последние десятилетия, она практически вытеснила системы, построенные на других протоколах.

Электронная почта, используемая в сети Интернет, задумывалась как простая в использовании и реализации, доступная и не дорогая. Ее первые разработчики не могли предвидеть и не учитывали лавинообразный рост сети, ее коммерциализацию, снижение среднего образовательного и культурного уровня пользователей. Потому электронная почта оказалась во многом не готова к вызовам последних лет: росту сетевой преступности и хулиганства, проблемам распространения вирусов и широковещательной рассылки бесполезной информации – спама.

Кроме того, существенно изменились требования к услугам, предоставляемым электронной почтой. Нынешние пользователи хотят не только обмениваться короткими сообщениями на английском языке, но и использовать различные форматы сообщений, кодировки, используемые в различных языках, прикреплять к электронным письмам файлы всевозможных форматов. Эти новые требования также заставляют вносить изменения в существующие стандарты.

Изменения, призванные улучшить электронную почту и решить встающие

перед ней проблемы, вносятся в стандарты регулярно, но, предлагая улучшения, к сожалению, необходимо учитывать, что потребовать их внедрения от администраторов сотен тысяч сетей и почтовых систем очень сложно. Потому, вводя улучшения и дополнения в действующие стандарты, приходится сохранять преемственность, чтобы не нарушить связность сети и не лишить электронную почту в Интернет ее главных преимуществ: доступности, дешевизны и простоты.

Как и все службы Интернет, электронная почта описывается издаваемыми IETF документами RFC. Основным руководящим документом для нее является RFC 2821 [1], в котором описывается протокол SMTP, используемый для доставки почтовых сообщений от почтовой программы отправителя до электронного почтового ящика получателя, а также основные принципы построения и функционирования электронной почты.

1. СТРУКТУРА ЭЛЕКТРОННОГО СООБЩЕНИЯ

Согласно RFC 2822 [2], электронное сообщение состоит из трех частей:

- конверт (envelope), содержащий адреса отправителя и получателей сообщения, эта информация используется только при пересылке сообщения по протоколу SMTP, получателю она недоступна;
- заголовок (header), содержащий служебную информацию, формируемую программами, участвующими в передаче сообщения, такую как адреса отправителя и получателей, которые могут отличаться от используемых в конверте, тему сообщения, время отправки, сведения о пересылке и об используемых для создания сообщения программах и т.д., заголовок завершается пустой строкой;
- тело (body), содержащее само сообщение, созданное отправителем и подлежащее доставке получателю.

Таким образом, сообщение доставляется получателю в виде заголовка и отделенного от него пустой строкой тела.

Заголовок состоит из полей: текстовых строк, состоящих из имени поля: слова, заканчивающегося двоеточием, и содержимого поля.

В заголовке допускается использование только символов в кодировке US-ASCII. Другие символы должны быть закодированы таким образом, чтобы полученная кодовая последовательность содержала только символы кодировки US-ASCII. Это правило нередко нарушается, например, тема сообщения записывается в заголовке сообщения на русском языке без перекодирования. Этого следует избегать, так как на приемном конце не будет известна используемая кодировка русских букв, а значит, полученная последовательность может быть интерпретирована неправильно. Этого не произойдет, если текст будет закодирован в соответствии с RFC 2047 [3].

Длинные поля заголовка разбиваются на несколько строк (табл. 1), при этом каждая строка, продолжающая предыдущую, начинается с пробельного символа.

Заголовок обычно показывается не полностью. Получатель видит только

некоторые поля: адреса отправителя и получателей, время отправки и тему сообщения.

Тело сообщения, если это не просто текст, записанный латинскими буквами, должно быть закодировано в соответствии со спецификацией MIME, как описано в RFC 2045 [4]. На приемной стороне оно при необходимости декодируется и преобразуется в понятный пользователю вид.

Таблица 1

Название поля	Значение поля
Date:	Время отправки сообщения
From:	Адрес отправителя
Reply-To:	Адрес для ответа
To:	Адреса получателей
Cc:	Адреса получателей копий
Bcc:	Адреса получателей скрытых копий. Это поле используется в процессе передачи сообщения, при доставке получателю соответствующие поля или часть их содержимого могут быть удалены
Message-ID:	Уникальный идентификатор сообщения
In-Reply-To:	Уникальный идентификатор сообщения, на которое отвечает данное сообщение
References:	Уникальные идентификаторы всех сообщений в цепочке ответов
Subject:	Тема сообщения
Return-Path:	Адрес отправителя, указанный на конверте сообщения
Received:	Информация о прохождении сообщения. Каждый узел, через который прошло сообщение, должен добавить в заголовок поле "Received:", содержащее имена и адреса IP узлов, пославших и принявших сообщение, время прохождения и пр.
MIME-Version:	Используемая версия MIME
Content-type:	Тип данных, используемых в теле сообщения
Content-Transfer-Encoding:	Способ кодирования символов не US-ASCII, используемый в тексте сообщения

1.1. Адреса электронной почты

Электронная почта в Интернет использует маршрутно-независимую адресацию. Это значит, что адрес пользователя остается неизменным независимо от того, откуда посылается сообщение. Такая адресация очень удобна для пользователей, но усложняет процесс доставки сообщения, так как определение маршрута доставки полностью ложится на программное обеспечение электронной почты.

В адресе допускается указание маршрута сообщения, но такие маршрутно-зависимые адреса используются редко (обычно, в отладочных целях). В общем случае их использование не имеет смысла.

Согласно RFC 2822 [2], электронный адрес имеет следующий формат:

имя_пользователя@почтовый_домен

где

имя_пользователя – идентификатор пользователя, уникальный в пределах одного почтового домена;

@ (коммерческое at) – символ-разделитель;

почтовый_домен – уникальный идентификатор почтовой системы.

Имя пользователя может состоять из цифр, латинских букв и символов (!; #; $; %; &; '; *; +; -; /; =; ?; ^; _; `; {; |; }; ~), может состоять из нескольких полей, разделенных точкой. Если имя пользователя содержит символы, отличные от перечисленных, его следует заключать в кавычки.

На практике имена пользователей обычно состоят только из цифр, латинских букв, символов «-», «_» и точки, которая интерпретируется не как разделитель полей, а как часть имени пользователя. Использование других символов может привести к неоднозначным интерпретациям, потому нежелательно.

Имя почтового домена имеет тот же формат, какой используется в доменных именах Интернет (RFC 1034 [5]), но, несмотря на внешнее сходство, функциональные назначения почтового домена и доменного имени узла существенно различаются. Подробнее почтовые домены будут рассмотрены ниже.

Кроме значимой части, используемой при маршрутизации сообщения, адрес может содержать комментарии в виде произвольных текстовых строк до и после значимой части. Чтобы отделить комментарий от значимой части адреса, последнюю заключают в угловые скобки.

В окончательном виде адрес электронной почты имеет следующий формат:

комментарий < имя_пользователя@почтовый_домен > комментарий

Пример

Леонид Свердлов <lonk@lonk.pp.ru> (каф. ОПДС)

Для отправки сообщения достаточно указать в качестве адреса lonk@lonk.pp.ru. Информация по обе стороны угловых скобок при доставке сообщения игнорируется.

Адрес электронной почты не всегда указывает непосредственно на существующий почтовый ящик. Он может указывать на другой адрес (почтовый псевдоним) или на множество почтовых ящиков (список рассылки). Почтовый ящик, соответствующий адресу, может вообще не существовать, а сообщения, поступающие на данный адрес, могут передаваться для обработки специальной программе.

Часто при отправке сообщения внутри одного почтового домена допускается использование адресов, состоящих только из имен пользователей.

1.2. Почтовые домены

На заре электронной почты в Интернет левая часть адреса электронной почты представляла собой регистрационное имя пользователя на машине, имя которой

указывалось в правой части адреса. Эта машина использовалась и для приема, и для отправки сообщений, и для хранения почтовых ящиков. Такое возможно и сейчас, но во всякой крупной организации с большим входящим и исходящим почтовым трафиком обработкой почты занимается множество машин, выполняющих различные функции (рис. 1).

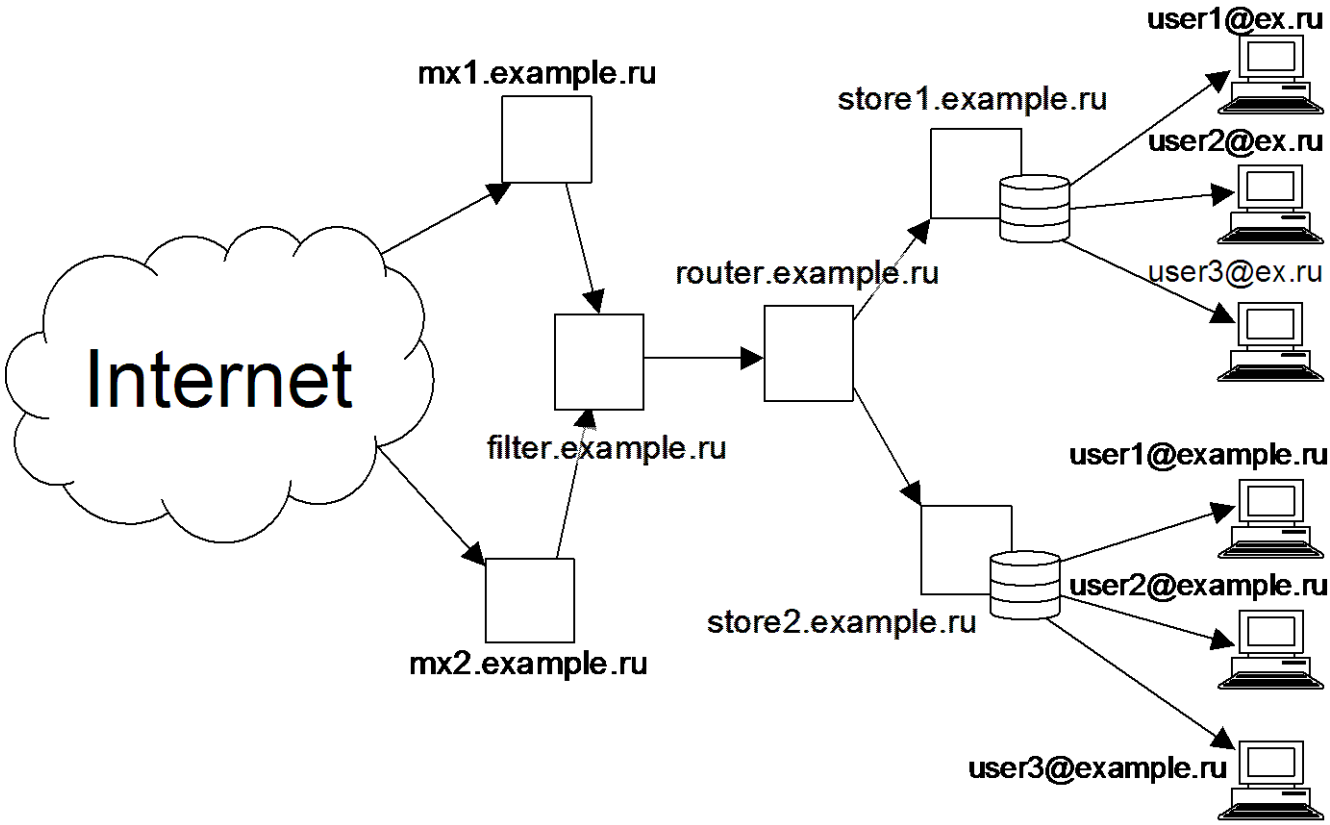

Рис. 1. Пример корпоративной системы, принимающей электронную почту

Организации (рис. 1) принадлежат два доменных имени: example.ru и ex.ru, причем первое имя используется для именования сетевых устройств и почтового домена, второе – только как имя почтового домена. Организация располагает двумя независимыми каналами, подключенными к Интернет. Узел mx1.example.ru предназначен для приема входящей почты для доменов example.ru и ex.ru, но, если он по какой-то причине недоступен, почту принимает узел mx2.example.ru, подключенный к другому каналу. Оба узла проверяют адреса получателей, принимают почту только для доменов example.ru и ex.ru, и отказываются принимать сообщения, предназначенные для пользователей других доменов. Принятая почта переправляется на filter.example.ru, где производится фильтрация спама и проверка на наличие вирусов. После этого почта поступает на почтовый маршрутизатор router.example.ru, который определяет, куда сообщения должны передаваться дальше. Имеются два хранилища сообщений: store1.example.ru содержит почтовые ящики пользователей домена ex.ru, store2.example.ru – example.ru.

Видно, что между почтовыми доменами и доменными именами узлов не

существует прямой связи, но, зарегистрировав доменное имя, его можно использовать и для наименования узлов, и как имя почтового домена.

Для маршрутизации электронной почты, как и для установления соответствия между доменными именами узлов сети и их адресами IP, используется система DNS. Получив сообщение, предназначенное для отправки, почтовый сервер посылает запрос DNS с указанием имени почтового домена получателя. В ответ почтовый сервер получает список узлов, принимающих почту для данного домена. Список представляется в виде так называемых записей MX (Mail eXchange). Одному имени почтового домена могут соответствовать несколько записей MX с различными приоритетами. Приоритеты обозначаются целыми числами, с их помощью определяется, в каком порядке следует обращаться к узлам, принимающим почту для данного домена.

Почтовый сервер, отправляющий сообщение пользователю с адресом в домене example.ru или ex.ru (рис. 1), в ответ на запрос DNS получит две записи MX: mx1.example.ru с приоритетом, обозначенным меньшим числом, и mx2.example.ru с приоритетом, обозначенным большим числом. Выяснив их адреса IP, почтовый сервер попытается соединиться с узлом, приоритет которого обозначен меньшим числом, если это не удается, то попытка повторяется для узла, приоритет которого обозначен большим числом и т.д.

Для просмотра информации DNS (в том числе записей MX) можно воспользоваться командой nslookup. После ее вызова из командной строки программа переходит в режим ожидания команд.

Для опроса записей MX следует ввести команду set type=mx

После этого вводится имя почтового домена, записи MX которого мы хотим просмотреть. Программа nslookup ко всем вводимым запросам добавляет стандартное имя домена, чтобы это предотвратить, имя опрашиваемого почтового домена следует завершить точкой.

Пример

lonk.pp.ru.

Ответ на этот запрос:

```
lonk.pp.ru          preference = 10, mail exchanger = sverdlov.pp.ru

lonk.pp.ru          preference = 20, mail exchanger = pds.sut.ru
sverdlov.pp.ru      internet address = 213.221.51.66
pds.sut.ru          internet address = 195.19.219.136
```

Эти строки означают, что почту для домена lonk.pp.ru принимают два узла: sverdlov.pp.ru (адрес IP: 213.221.51.66) с приоритетом 10 и pds.sut.ru (адрес IP: 195.19.219.136) с приоритетом 20.

Значит, почта, предназначенная для получателя lonk@lonk.pp.ru должна посылаться через узел sverdlov.pp.ru. Если соединиться с этим узлом не удается, то ее следует посылать через узел pds.sut.ru. Если и это невозможно, то исходящее сообщение следует поместить в очередь и через некоторое время повторить попытку.

1.3. Контрольные вопросы

1. Из каких частей состоит электронное сообщение? Какие из них доставляются получателю?
2. В какой части электронного сообщения находится адрес получателя, используемый при доставке сообщения?
3. Какие поля заголовка содержат адреса отправителя и получателей? Чем различаются эти поля?
4. В какой части электронного сообщения находится его тема?
5. Каким образом передается тело сообщения, содержащее не только текст, записанный латинскими буквами?
6. Чем отличается маршрутно-зависимая адресация от маршрутно-независимой? Какая из них обычно используется в сети Интернет?
7. Какова структура адреса электронной почты в Интернет?
8. Что такое почтовый домен? Что общего и в чем различие между почтовым доменом и доменным именем сетевого узла?
9. Что такое запись MX? Для чего она используется?
10. Каким образом определяется, какой сетевой узел принимает почту для домена? Как можно задать альтернативные узлы принимающие почту для того же домена?

1.4. Практическое задание

Покажите, из каких компонентов состоит ваш адрес электронной почты. В каком почтовом домене находится ваш почтовый ящик.

С помощью команды nslookup выясните, какой именно сетевой узел принимает почту для вашего домена. Если таких узлов несколько, определите, какой из них принимает почту по умолчанию, а какие в случае недоступности основного сервера.

С помощью клиентской почтовой программы откройте исходный текст одного из полученных вами сообщений (в программе The Bat! для этого нужно отметить сообщение и нажать кнопку F9). Какие компоненты сообщения вы видите?

Проанализируйте заголовок сообщения? Какие из полей заголовка показывает клиентская почтовая программа? Как найти конец заголовка?

Сравните тело сообщения с тем, что показывает вам клиентская почтовая программа. Всегда ли они совпадают?

2. СТРУКТУРА ЭЛЕКТРОННОЙ ПОЧТЫ

Рассмотрим функции каждого компонента почтовой системы, построенной на базе протокола SMTP (рис. 2).

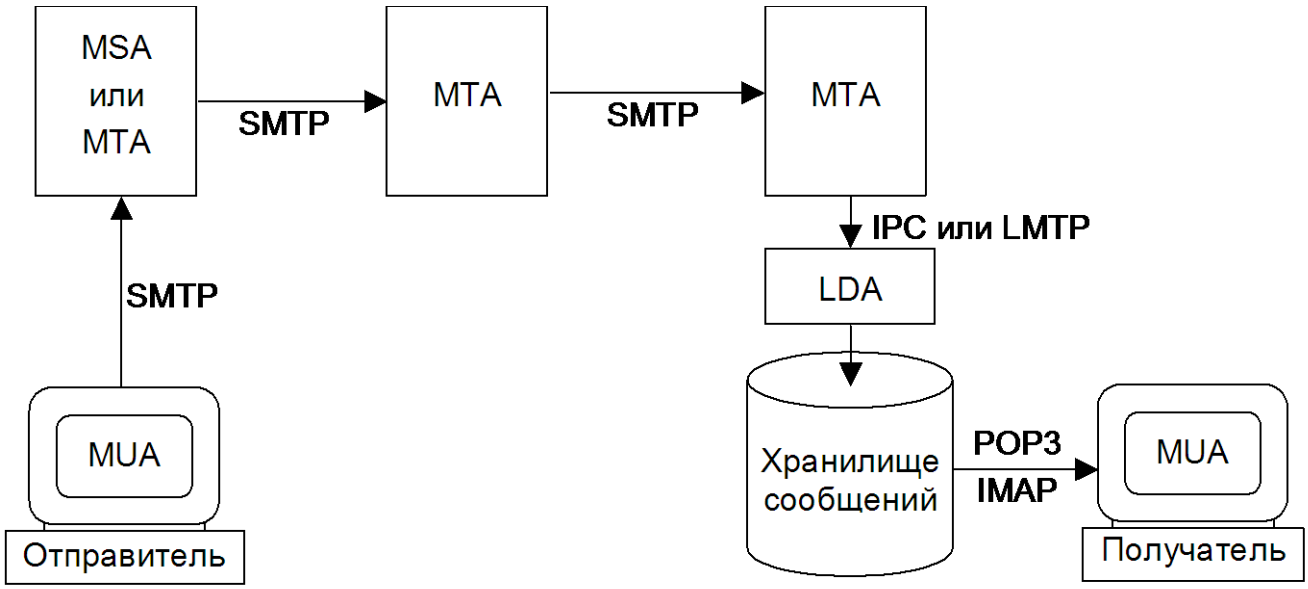

Рис. 2. Структура электронной почты в Интернет:

Mail User Agent (MUA) – пользовательский агент (или клиентская почтовая программа;
Mail Transfer Agent (MTA) – транспортный агент (или почтовый сервер);
Local Delivery Agent (LDA) – агент локальной доставки;
Message Submission Agent (MSA) – агент подачи сообщения

2.1. MUA

MUA предназначен для подготовки, отправки, получения и просмотра электронных писем. Это программа, установленная на компьютере пользователя. Задача электронной почты, по сути дела, сводится к тому, чтобы доставить сообщение от MUA отправителя на MUA получателя.

Подготовка к отправке заключается в приведении сообщения к принятому формату (RFC 2822 [2]).

MUA отправителя должен сформировать заголовок сообщения, а также закодировать и оформить его тело в соответствии со стандартом, чтобы MUA принимающей стороны смог правильно интерпретировать и представить как текст, так и вложения письма.

MUA обычно устанавливается на машине пользователя и, как правило, запускается только на время работы пользователя, а сам компьютер может не иметь постоянного подключения к сети, поэтому MUA не может выступать в качестве сервера: он может быть только инициатором соединения, т. е. клиентом.

MUA посылает сообщения по протоколу SMTP через MSA или MTA, используемый для отправки почты.

Входящие письма MUA забирает из хранилища сообщений по протоколу, предназначенному для получения почты, для чего используется один из двух протоколов:

- Post Office Protocol - Version 3 (POP3) – протокол почтового отделения, версия 3 (RFC 1939 [6]), позволяющий просматривать сообщения в почтовом ящике, забирать и удалять их;
- Internet Message Access Protocol (IMAP) – протокол доступа к сообщениям (RFC 3501 [7]), обладающий более широкими возможностями манипулирования почтовыми ящиками, чем POP3 (в частности позволяет работать с несколькими ящиками одновременно, не только считывать и удалять, но и создавать и исправлять сообщения).

Возможны и другие способы получения почты (например, использование локальной доставки, если хранилище сообщений доступно MUA по локальной сети).

Существует множество различных программных реализаций пользовательского агента: Microsoft Outlook, Netscape Communicator, The Bat!, Eudora, Elm, Pine и др.

Довольно большое распространение получили агенты пользователя, использующие интерфейс CGI для доступа оконечного пользователя к его почтовому ящику по протоколу HTTP или более безопасному HTTPS при помощи web-браузера. Такую реализацию MUA часто называют web-mail (рис. 3).

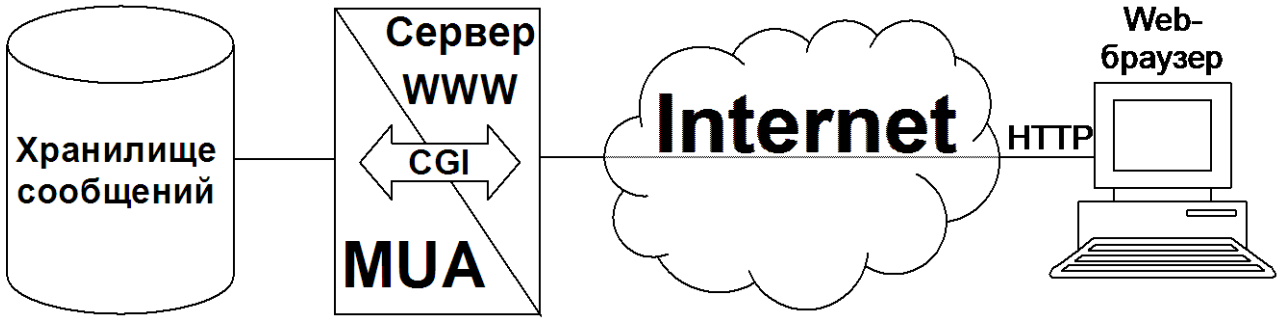

Рис. 3. Структура web-mail

Пользовательский интерфейс реализуется с помощью технологий WWW. Функции MUA выполняет приложение, взаимодействующее с web-сервером при помощи интерфейса CGI. MUA получает доступ к хранилищу сообщений по протоколам POP3 или IMAP или путем непосредственного обращения: MUA при такой реализации может быть включен в ту же локальную сеть, что и хранилище сообщений (они даже могут быть запущены на одной и той же машине).

Преимущество web-mail перед MUA, установленным на компьютере пользователя – это возможность работать со своей почтой с любого компьютера, подключенного к сети, без предварительной настройки и без инсталляции программного обеспечения. Недостаток web-mail заключается в том, что пользователю для работы с почтой необходим постоянный доступ к Интернет, так как каждый запрос выполняется не на пользовательской машине, а на web-

сервере, и должен быть передан по сети.

Услуга web-mail предоставляется популярными системами бесплатной общедоступной электронной почты: Hotmail, Mail.ru, Yahoo, GMX и др. Впрочем, наряду с этой услугой, многие из таких систем позволяют получать почту по протоколу POP3.

Web-mail нередко реализуется в корпоративных почтовых системах, пользователи которых должны читать почту вне рабочих мест.

2.2. MTA

MTA представляют собой узлы, через которые передаются электронные сообщения. Письмо, сформированное MUA, достигает хранилище сообщений, содержащее почтовый ящик получателя, проходя через один или несколько MTA, последний из которых передает письмо агенту локальной доставки (LDA).

Все сетевые узлы (рис. 1), обозначенные квадратами, являются MTA. Как видно, функции MTA могут быть очень разнообразны и не сводятся только к передаче сообщения.

Как правило, MTA должны быть доступны круглосуточно и постоянно ожидать подключения по протоколу SMTP, по которому происходит обмен данными между MTA. MTA, отправляющий почту, инициирует соединение и выступает в качестве клиента, а MTA, принимающий почту, является сервером.

На MTA также возлагается разбор адресов получателей, раскрытие списков рассылки и почтовых псевдонимов, определение маршрута сообщения на основании анализа адресов получателей и записей MX, которые MTA получает от сервера DNS.

MTA должен проверять достоверность идентификационных данных, получаемых от встречного MTA. Следует проверять соответствие доменного имени, которое клиент сообщает в приветствии, его адресу IP. Также нужно удостовериться в существовании почтового домена, указанного в почтовом адресе отправителя. Если в доменной части адреса получателя указан почтовый домен, обслуживаемый данным MTA, то следует проверить, зарегистрирован ли в этом домене указанный адресат.

В целях предотвращения анонимных рассылок спама, RFC 2505 [8] рекомендует принимать почту только при выполнении хотя бы одного из следующих условий:

- адрес IP клиента входит в список адресов клиентов, обслуживаемых данным MTA;
- получатель сообщения зарегистрирован в почтовом домене, обслуживаемом данным MTA;
- клиент прошел процедуру аутентификации.

Если не выполнено ни одно из названных условий, MTA должен отказать в приеме почты. MTA, принимающий почту, не отвечающую перечисленным требованиям, может быть внесен в списки серверов, не препятствующих распространению спама. В этом случае многие почтовые системы будут

отказываться принимать от него почту.

MTA может производить обработку проходящих через него сообщений: проверку на наличие вирусов, фильтрацию спама и пр.

Каждый MTA, через который проходит почтовое сообщение, добавляет к его заголовку информацию о том, когда и откуда пришло это сообщение, а также результаты произведенных проверок.

В случае невозможности немедленной доставки сообщения, оно помещается в очередь. MTA регулярно предпринимает новые попытки отправить сообщения из очереди. Если это не удается за определенный срок (обычно 4 ч), отправителю посылается предупреждение о задержке доставки. Но сообщение остается в очереди, и попытки его отправить продолжаются. Если в течение длительного времени (обычно 5 дней) сообщение так и не удается доставить, оно удаляется из очереди, а отправителю посылается сообщение о невозможности доставки письма.

MTA могут также выполнять и другие функции, в зависимости от используемого программного обеспечения.

Основные требования к MTA и MUA приведены в RFC 1123 [9] и уточнены в RFC 2821 [1], RFC 2822 [2].

Существует множество разнообразных программных реализаций MTA. Старейшей из них и до сих пор одной из наиболее популярных является программа sendmail, разработанная в начале 80-х гг. Эриком Оллмэном, тогда еще студентом Калифорнийского университета в Беркли. Эта программа многократно дорабатывалась и стала фактически стандартом для этого типа программного обеспечения. Она продолжает совершенствоваться и по сей день. Существует как свободно распространяемая для операционных систем, совместимых с UNIX, так и коммерческая версия.

Позже появились и другие программные продукты, реализующие функции MTA для различных операционных систем: Postfix, smail, qmail, exim, ZMailer и многие другие.

2.3. MSA

Как отмечалось, MTA, через которые проходит сообщение, добавляют некоторые строки в его заголовок. Однако информацию, уже содержащуюся в сообщении, MTA не изменяют, хотя необходимость в этом может возникнуть, если заголовок неправильно оформлен (не полностью определено имя домена, ошибочно указано время или дата). Может возникнуть необходимость в корректировке адреса отправителя, если в почтовой сети предприятия используется адресация, отличная от принятой в Интернет (использование адресов без указания почтового домена для пользователей, зарегистрированных в почтовой системе предприятия).

Функции корректировки заголовка сообщения можно возложить на MTA, принимающий почту от агентов пользователя, но, если поток почты велик, имеет смысл использовать для этого специальный процесс – MSA.

Таким образом, можно сказать, что MSA это разновидность MTA, занимающаяся предварительной обработкой исходящей почты. Подробнее задачи и особенности реализации MSA рассмотрены в RFC 2476 [10].

Чтобы различать MTA и MSA, рекомендуется запускать MSA, используя не порт 25, предназначенном для MTA, а другой порт TCP, либо использовать порт 25 на сервере, где не запущен MTA.

2.4. LDA

Последний MTA на пути следования электронного почтового сообщения должен передать его агенту локальной доставки. Обычно LDA расположен на одной машине с MTA и представляет собой программу, которая вызывается агентом передачи сообщения при поступлении новых сообщений. В этом случае для взаимодействия между MTA и LDA используются механизмы межпроцессного взаимодействия (IPC). В некоторых случаях LDA также может быть реализован как сервер, принимающий от MTA почту по протоколу Local Mail Transfer Protocol, LMTP (RFC 2033 [11]).

Агентом доставки называется программа, производящая обработку поступившей почты. В основном эта обработка заключается в помещении сообщений в почтовые ящики адресатов, то есть в добавлении сообщений к соответствующим файлам или в размещении их в специальных каталогах пользователей или в базах данных. Пользователь сможет получить сохраненные сообщения, соединившись с хранилищем сообщений по протоколу POP3 или IMAP.

Другой вид обработки сообщений – передача их каким-либо программам для дальнейшей обработки. Для выполнения этих функций LDA должен при необходимости раскрывать почтовые псевдонимы и списки рассылки.

2.5. Хранилище сообщений

Электронные сообщения обычно не доставляются автоматически на машину пользователя, а помещаются в хранилище сообщений, откуда пользователь может их забрать в удобное для него время. Каждому пользователю выделяется ограниченный или неограниченный объем дискового пространства, физически реализованный в виде файла специального формата, каталога специальной структуры или набора записей в базе данных. Элемент хранилища сообщений, содержащий электронные сообщения, поступившие на определенный адрес, называется почтовым ящиком.

Доступ пользователей к сообщениям, находящимся в хранилище, обычно осуществляется по протоколам POP3 или IMAP. В качестве клиента выступает MUA пользователя. Сервер имеет непосредственный доступ к хранилищу сообщений, ожидает подключений пользовательских агентов и, после обязательной аутентификации, определяет права доступа, установленные для данного пользователя (пользователь должен иметь доступ не менее чем к одному

почтовому ящику).

Какие именно манипуляции пользователь может проделывать со своими почтовыми ящиками и с содержащимися в них сообщениями, зависит от используемого программного обеспечения. При минимальной реализации пользователь получает доступ к одному почтовому ящику, сообщения в который помещаются LDA. Пользователь может получать и удалять отдельные сообщения. Такой вид доступа, в большом числе случаев достаточный, реализуется при использовании протокола POP3. Другой популярный протокол доступа к электронным почтовым ящикам – IMAP предоставляет более широкие возможности.

Все чаще возникает необходимость в создании масштабируемых и гибких систем, выполняющих функции хранилищ электронных сообщений. Имеется в виду как физическое размещение почтовых ящиков одного хранилища на разных сетевых узлах, так и возможность дублирования хранилища сообщений или его части. Первая задача решается средствами протокола IMAP, вторая задача сейчас находится на начальном этапе разрешения. В RFC 3656 [12] описан экспериментальный протокол обновления почтового ящика – MUPDATE, который, возможно, получит распространение в будущем и позволит создавать распределенные хранилища сообщений с дублированием почтовых ящиков на разных машинах. Это даст возможность снизить нагрузку на серверы POP3 и IMAP и избежать проблем, связанных с выходом из строя единственного хранилища сообщений.

2.6. Доставка почтового сообщения

Рассмотрим путь почтового сообщения (рис. 4), порядок следования отдельных событий обозначен числами на стрелках.

1. Сообщение, сформированное MUA отправителя, по протоколу SMTP посылается MSA. MSA проверяет, имеет ли данный MUA или пользователь право посылать почту из этой почтовой системы. В случае положительного результата, сообщение принимается для дальнейшей доставки.
2. MSA проверяет заголовок сообщения и, при необходимости, исправляет его. Готовое к отправке сообщение по протоколу SMTP отправляется на MTA исходящей почты.
3. MTA исходящей почты анализирует адрес получателя. Если сообщение предназначено для получателя домена, обслуживаемого данной почтовой системой, то оно доставляется получателю (см. пп. 6–10), в противном случае MTA запрашивает информацию о почтовом домене, указанном в адресе получателя, сервер DNS. Получив запрашиваемые данные, сервер DNS сообщает MTA, какие узлы принимают почту для данного домена, их адреса IP и приоритеты.
4. MTA отправителя пытается установить соединение по протоколу с принимающими почту узлами в соответствии с приоритетами, указанными в записях MX, полученных от сервера DNS. Если ни с одним узлом соединение

не удается установить, сообщение помещается в очередь, и через некоторое время попытки установить соединение повторяются. Если соединение установлено, принимающий MTA, удостоверившись, что сообщение предназначено для пользователя его домена, и что почтовый ящик с указанным адресом действительно существует, принимает сообщение.

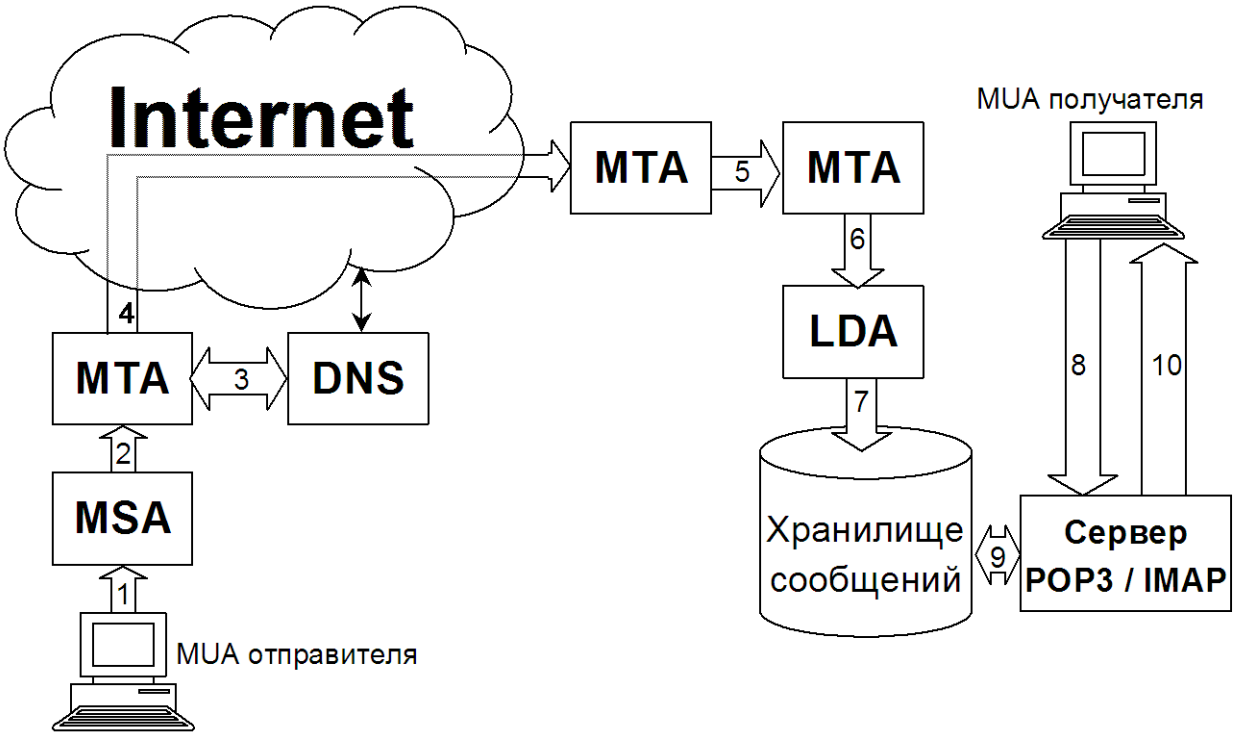

Рис. 4. Доставка электронного сообщения от отправителя к получателю

5. В принимающей почтовой системе сообщение может пройти через несколько промежуточных MTA, выполняющих различные виды обработки входящей почты: проверку на вирусы, фильтрацию спама, перенаправление к нужному хранилищу сообщений и пр. Внутри принимающей системы может использоваться как SMTP, так и LMTP.
6. Последний MTA, используя межпроцессное взаимодействие или протокол LMTP, передает сообщение LDA для локальной доставки.
7. LDA помещает сообщение в почтовый ящик адресата.
8. Получатель обращается к серверу POP3 или IMAP, чтобы проверить поступившую почту.
9. Сервер забирает сообщение из почтового ящика.
10. Сервер посылает сообщение пользовательскому агенту получателя.

Таким образом сообщение доставляется от отправителя к получателю. Далее рассматриваются протоколы, используемые в процессе доставки.

2.7. Контрольные вопросы

1. Какие компоненты системы электронной почты в Интернет вы знаете? Каково их назначение?
2. Какие протоколы используются в процессе доставки электронного сообщения?
3. В функции какого компонента системы электронной почты входит определение маршрута электронного сообщения? Каким образом это осуществляется?
4. Какой компонент системы электронной почты формирует сообщение, подлежащее отправке?
5. Какие программы вы используете в качестве MUA?
6. Что такое web-mail? В чем его преимущества и недостатки по сравнению с MUA, установленным на компьютере пользователя?
7. В чем различия между MTA и MSA?
8. В каких случаях MTA или MSA принимает сообщение для дальнейшей пересылки, а в каких отвергает? Почему нельзя принимать для пересылки все поступающие сообщения с существующими адресами отправителей и получателей?
9. Как пользователь получает доступ к хранилищу электронных сообщений? Что оно из себя представляет?
10. Опишите процесс доставки сообщения от отправителя к получателю.

2.8. Практическое задание

Проанализируйте маршруты нескольких почтовых сообщений на основании полей «Received» их заголовков. Обратите внимание, что записи «Received» следуют в обратном порядке, т. е. первая запись соответствует последнему узлу, через который прошло сообщение.

3. ПРОТОКОЛ SMTP

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 80-х гг. прошлого века. Окончательная версия была закреплена в RFC 821 [13]. Все годы, прошедшие с того времени, 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 80-е гг. Эти команды, к сожалению, создают удобства не только для сетевых администраторов, но и для злоумышленников, поэтому их обычно используют только на этапе настройки почтовой системы, а в работающих системах их обычно отключают.

В 2001 г. RFC 821 [13] был заменен RFC 2821 [1], который на сегодняшний день является основным стандартом, описывающим протокол SMTP. Новый стандарт учитывает изменения, произошедшие в сети за последние годы.

SMTP может работать с различными протоколами транспортного уровня (см. RFC 821 [13] и RFC 1090 [14]), 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, но 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, т.е. символов, кодируемых 7 битами. Это цифры, латинские буквы, и знаки препинания. Если информация передается 8-битными блоками (октетами), старший бит должен быть равен нулю. Корректная интерпретация символов, старший, 8-й бит которых равен единице, например, русских букв, не гарантируется, поэтому использовать такие символы не следует.

Конец строк в протоколе SMTP обозначается последовательностью символов «возврат каретки» (16-ричный код 0D) и «перевод строки» (16-ричный код 0A). Эта последовательность обозначается CRLF. Сервер начинает выполнение команды, только получив от клиента строку, завершающуюся последовательностью CRLF.

Серверы 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.

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 [13] 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.

HELO (Приветствие)

Формат команды

HELO полное_доменное_имя_клиента CRLF

или

HELO адрес_отправителя CRLF

RFC 2821 [1] 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.

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 HELO.

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.

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 [1], серверы SMTP должны быть готовы принять до 100 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.

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 впоследствии окажется невозможна, следует просто отправить письмо обратно, сообщив в квитанции причину отказа. Это делается для того, чтобы избежать проблемы, описанной в RFC 1047 [15]: 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 10 мин. На ожидание других ответов отводятся от 2 до 5 мин.

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 кбайта, включая тело и заголовок сообщения.

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 [1] предусматривает еще ряд команд, которые могут быть использованы в основном в процессе отладки сервера.

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, клиенту посылается описание этой команды. Серверам SMTP рекомендуется поддерживать эту команду без параметров. Описание отдельных команд посылать не обязательно.

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.

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, не давая ответа о существовании его на сервере.

Согласно RFC 821 [1], команды VRFY и EXPN не являются обязательными, если сервер их поддерживает, они должны быть перечислены в ответе на команду EHLO как расширения ESMTP.

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.

RSET (Сброс)

Формат команды

RSET CRLF

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

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 на терминал и в почтовый ящик (SAML).

Описанные в RFC 821 [13], эти команды признаны в RFC 2821 [1] устаревшими. Если их все же используют, то они должны быть перечислены в ответе на команду EHLO как расширения ESMTP.

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. Протокол SMTP предусматривает отправку сообщений только от клиента к серверу, поэтому для передачи в обратном направлении им необходимо поменяться ролями.

Команда TURN представляет потенциальную опасность, так как она может быть использована для перехвата чужой почты, поэтому RFC 2821 [1] 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 [1], 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 кода 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.

Третья цифра кода зависит от содержания ответа.

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.

Таблица 2

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 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 S S S S	250-foo.com greets bar.com 250-8BITMIME 250-SIZE 250-DSN 250 HELP	Сервер сообщает о поддерживаемых расширениях ESMTP
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	
C	.	Сообщение заканчивается строкой, состоящей из одной точки
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 [1]. 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. Они будут рассмотрены 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 его параметры, если необходимо.

8BITMIME – передача текста сообщения 8-битным кодом

В базовом протоколе SMTP не предусмотрена передача текста сообщения в кодировке, отличной от US-ASCII. В сообщении могут использоваться только символы, кодируемые 7 битами. Если при передаче используется 8-битный 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 7-битный код.

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

RFC 1652 [16] предлагает расширение ESMTP, позволяющее передавать в поле команды DATA 8-битные символы. Действие этого расширения распространяется только на тело сообщения, но не на его заголовок. Использование 8-битных символов в заголовке сообщения не допускается, при необходимости они кодируются в соответствии с RFC 2047 [3], 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

250 8BITMIME

Эта строка сообщает клиенту, что сервер поддерживает прием сообщений, в теле которых используются 8-битные символы. Если ответ на команду EHLO не содержит такой строки, клиент должен преобразовать сообщение, чтобы оно содержало только 7-битные 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 8-битные символы, все биты сообщения должны передаваться без изменений;

7BIT – сообщение содержит только символы в кодировке US-ASCII, при передаче по 8-битному каналу старший бит каждого октета должен быть установлен в ноль (принято по умолчанию).

Пример

S	220 dbc.mtview.ca.us SMTP service ready	
C	EHLO ymir.edu	
S	250-dbc.mtview.ca.us says hello	
S	250 8BITMIME	Сервер поддерживает расширение 8BITMIME
C	MAIL FROM:<ned@ymir.edu> BODY=8BITMIME	Модифицированная команда MAIL. Тело сообщения содержит 8-битные 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	

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 самое сообщение. Чтобы избежать повторной передачи уже переданной части сообщения, RFC1845 [17]

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

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 ч, 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.

Пример

S	220 dbc.mtview.ca.us SMTP service ready	
C	EHLO ymir.edu	
S	250-dbc.mtview.ca.us says hello	
S	250 CHECKPOINT	Сервер поддерживает расширение 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, при передаче сообщения, связь 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. Ранее было принято 6135 октетов
C	DATA	
S	354 Send previously checkpointed message starting at octet 6135	
C	Информация передается, начиная с октета 6136	Передача заканчивается строкой из одной точки
C	.	
S	250 OK	
C	QUIT	
S	221 Goodbye	

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 неизвестен, 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 [18], 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.

Если сервер SMTP поддерживает расширение SIZE, в ответе на команду EHLO должна быть передана строка

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.

Пример

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>	
S	250 OK; can accomodate 500000 byte message	Сообщение может быть доставлено получателю
C	RCPT TO:<ned@ymir.claremont.edu>	
S	552 Channel size limit exceeded	Получатель не может принимать сообщения такого размера
C	RCPT TO:<ned@hmcvax.claremont.edu>	
S	452 Insufficient channel storage	Доставка сообщения временно невозможна. Почтовый ящик переполнен
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	

ETRN – получение сообщений из удаленной очереди

Если сервер SMTP не подключен к сети постоянно, 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 [19] 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.

Если сервер SMTP поддерживает расширение ETRN, в ответе на команду EHLO должна быть передана строка

250 ETRN

Вводится дополнительная команда ETRN.

ETRN доменное_имя_оконечного_сервера CRLF

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

Пример

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	
S	250 OK, queuing started	Команда выполнена успешно. Промежуточный сервер установит соединение с оконечным сервером и передаст ему накопившуюся почту
C	ETRN sigurd.innosoft.com	

S	251 OK, no messages waiting	Команда принята, но почты для передачи оконечному серверу нет
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	

ENHANCEDSTATUSCODES – расширенные коды ответов

Расширенные коды ответов сервера SMTP (RFC 3463 [20]) позволяют представить ответы сервера в более понятном для машины виде, чем обычные коды ответов.

Расширение ESMTP, описанное в RFC 2034 [21], позволяет использовать такие коды.

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

250 ENHANCEDSTATUSCODES

Далее 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. Описания расширенных кодов даны в RFC 3463 [20] (здесь не приводятся).

Пример

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

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 [22], позволяет использовать простой уровень аутентификации и безопасности – Simple Authentication and Security Layer, SASL (RFC 2222 [23]), 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, целесообразно использовать этот способ аутентификации на участке между MUA отправителя и MSA или MTA, принимающим почту от пользовательских агентов).

Если сервер SMTP поддерживает расширение AUTH, в ответе на команду EHLO должна быть передана строка

250 AUTH параметры

В качестве параметров передаются разделенные пробелами ключевые слова, соответствующие механизмам аутентификации, поддерживаемым сервером:

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, начинается обмен аутентификационной информацией, закодированной по алгоритму Base64 (RFC 3548 [24]). Ответы сервера имеют код 334. Если первая строка не была передана в команде AUTH, 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, 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, дополнительный параметр AUTH команды MAIL используется для аутентификации отправителя, т.е. параметр 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 (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=<>

аутентификация отправителя по какой-то причине не состоялась.

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

STARTTLS – передача данных по защищенному каналу с использованием сертификатов

Расширение ESMTP STARTTLS (RFC 3207 [25]) предоставляет функции аутентификации клиента и криптографической защиты (шифрования) данных. Для этого используется механизм безопасности транспортного уровня – Transport Layer Security (TLS) (RFC 2246 [26]). Шифрование происходит на основе сертификатов, с использованием открытых и закрытых ключей.

На основании сертификатов происходит аутентификация и встречной стороны: узел не может вести диалог, пользуясь чужим сертификатом. Однако такой способ аутентификации не позволяет проверить подлинность передаваемого сообщения: аутентифицируются только узлы, а не пользователи. 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.

Если сервер SMTP поддерживает расширение STARTTLS, в ответе на команду EHLO должна присутствовать строка

250 STARTTLS

Убедившись, что сервер поддерживает это расширение, клиент посылает ему команду STARTTLS без 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 кодом 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, а 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 незащищенном соединениях, 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. Следовательно, сообщение, посылаемое через Интернет, может часть пути проделать по защищенным соединениям, а другую часть пройти открытым текстом. На этих участках оно может быть перехвачено злоумышленником.

Пример

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	

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 [27] 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, передача почты ни для одного

из них не осуществляется, команда отвергается с кодом ошибки 450.

Рассмотрим выполнение команды ATRN. Поскольку клиент и сервер меняются в процессе диалога ролями, участники диалога обозначены так: A – вызывающий и B – вызываемый.

Пример

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	

DSN – оповещения о ходе доставки

Протокол SMTP предусматривает посылку отправителю сообщений о проблемах при доставке корреспонденции, но часто таких сообщений бывает недостаточно. Расширение ESMTP DSN – Delivery Status Notifications (RFC 3461 [28]), увеличивает возможности имеющейся услуги оповещений о ходе доставки.

Если сервер SMTP поддерживает расширение DSN, в ответе на команду EHLO должна быть передана строка

250 DSN

Вводятся два дополнительных параметра команды MAIL:

- RET. Если RET=FULL, 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;
- 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:

- 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 ч). Если уведомление вообще не должно посылаться, аргумент принимает значение NEVER;
- 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, т.е. 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.

Пример

(из-за нехватки места некоторые команды разделены на две строки, 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

DELIVERBY – управление временем доставки сообщения

Расширение ESMTP DELIVERBY (RFC 2852 [29]) позволяет отправителю определять, в какой срок должна быть осуществлена доставка сообщения и как следует поступить с сообщением, которое не удалось своевременно доставить.

Если сервер SMTP поддерживает расширение DELIVERBY, в ответе на команду EHLO должна быть передана строка

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, 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 уведомление 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.

Пример

C	EHLO acme.net	
S	250-mail.other.com	
S	250 DELIVERBY 30	Сервер поддерживает расширение DELIVERBY. Время доставки, выставляемое в параметре BY должно быть не меньше 30 с
		Сообщение следует доставить не

C	MAIL FROM:<eljefe@bigbiz.com> BY=98;R	более, чем за 98 с. Если письмо задержится дольше, его следует удалить, оповестив отправителя.
S	250 <eljefe@bigbiz.com> Sender okay	

PIPELINING – группирование команд

Согласно RFC 821 [13] и RFC 2821 [1], клиент SMTP может передавать серверу следующую команду, только получив ответ на предыдущую. Ожидание ответа требует времени и, если список получателей большой, обмен множеством коротких дейтаграмм TCP может значительно затянуться. Предложенный RFC 2920 [30] 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

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 неперечисленных 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.

Пример

C	EHLO dbc.mtview.ca.us
S	250-innosoft.com 250 PIPELINING
C	MAIL FROM:<mrose@dbc.mtview.ca.us> RCPT TO:<ned@innosoft.com> RCPT TO:<dan@innosoft.com> RCPT TO:<kvc@innosoft.com> DATA
S	250 sender <mrose@dbc.mtview.ca.us> OK 250 recipient <ned@innosoft.com> OK 250 recipient <dan@innosoft.com> OK 250 recipient <kvc@innosoft.com> OK 354 enter mail, end with line containing only "."
C	Передается сообщение . QUIT
S	250 message sent 221 goodbye

Горизонтальные линии разделяют блоки информации, которые могут быть

переданы за один раз – одной дейтаграммой TCP.

Как видим, в данном примере 7 команд, текст сообщения и 8 ответов сервера могут быть переданы всего 6 дейтаграммами TCP.

CHUNKING, BINARYMIME – передача двоичных файлов большого объема

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 предназначено для передачи текстов, так как оно сохраняет ограничение на длину строки.

Основным способом передачи двоичных файлов по электронной почте является их кодирование в 7-битные 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 [31] описывает два расширения ESMTP, предназначенные для передачи двоичных файлов большого объема. Первое расширение – CHUNKING водит новую команду BDAT, заменяющую команду DATA.

Если сервер SMTP поддерживает расширение CHUNKING, в ответе на команду EHLO должна быть передана строка

```
250 CHUNKING
```

Команда BDAT позволяет передавать двоичные файлы любого формата, причем передавать большой файл можно по частям. Сервер может сообщить о неполадках только после передачи файла, 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, 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.

RFC 3030 [31] 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 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

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

Local Mail Transfer Protocol – LMTP (RFC 2033 [11]) 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 собой 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

При помощи программы Telnet соединитесь с сервером SMTP, принимающим почту для вашего ящика.

Выполните команду EHLO. Какие параметры следует передать с ней? Насколько они должны быть достоверны?

Какие расширения ESMTP поддерживает сервер?

Попробуйте проверить свой почтовый адрес с помощью команды VRFY. Какой код ответа вы получили? Что означает этот код?

Укажите произвольный адрес в качестве адреса отправителя. Какие адреса можно использовать? Каким образом сервер SMTP проверяет достоверность указанного адреса отправителя?

Укажите в качестве получателя адрес своего почтового ящика.

Передайте сообщение с помощью команды DATA.

Получив подтверждение о приеме сообщения, завершите сеанс связи командой QUIT.

С помощью клиентской почтовой программы проверьте, получили ли вы отправленное только что тестовое сообщение.

4. ПРОТОКОЛ POP3

Протокол почтового отделения – версия 3 – Post Office Protocol – Version 3 (POP3) (RFC 1939 [6]) предназначен для получения сообщений, находящихся в почтовом ящике пользователя на удаленном сервере электронной почты.

Как правило, не целесообразно устанавливать серверы SMTP на рабочие станции, предназначенные для чтения писем. Сервер SMTP должен быть доступен постоянно, а рабочие станции обычно включают только на время работы пользователя, соединение с сервером они нередко устанавливают по коммутируемым линиям только для того, чтобы забрать накопившуюся почту.

По протоколу SMTP почта доставляются только в хранилище сообщений, откуда пользователь может ее забрать в удобное для него время.

Таким образом, в качестве клиента POP3 выступает MUA пользователя, а сервер должен иметь доступ к хранилищу сообщений. Информация по протоколу POP3 передается от сервера к клиенту.

POP3 прост в реализации и предоставляет минимальные необходимые возможности для работы с почтовым ящиком. Вопреки распространенному мнению, версия 3 протокола POP дает возможность работать не только с ящиком в целом, но и с отдельными сообщениями, находящимися в нем, позволяя просматривать информацию о письмах, получать и удалять их по отдельности. К сожалению, не все существующие MUA используют эти возможности протокола POP3. Пользователь не всегда хочет скачивать с сервера все содержимое почтового ящика, и часто предпочел бы получать только некоторые сообщения, а другие сообщения, возможно, удалил бы, не получая. Все это можно сделать, используя протокол POP3. Более широкие возможности предоставляет протокол IMAP4 [7], но в большом числе случаев возможностей протокола POP3 оказывается вполне достаточно.

Сеанс протокола POP3 (рис. 5) делится на три этапа (состояния).

Рис. 5. Состояния сеанса POP3

Сервер ожидает соединения по порту TCP 110.

После установления соединения сервер посылает клиенту строку приветствия, свидетельствующую о готовности к диалогу, и сеанс переходит в состояние авторизации (AUTHORIZATION State). На этом этапе выясняется, доступ к какому именно почтовому ящику запрашивает клиент и имеет ли он соответствующие права. Успешное прохождение авторизации необходимо для продолжения работы.

Если авторизация проходит успешно, то сеанс переходит в состояние транзакции (TRANSACTION State). На этом этапе клиент может проделывать все необходимые манипуляции с почтовым ящиком: он может просмотреть информацию о состоянии ящика и отдельных сообщений, получить выбранные сообщения и пометить письма, подлежащие удалению.

По окончании всех операций, клиент сообщает об окончании связи, и сеанс переходит в состояние обновления (UPDATE State). На этом этапе сервер стирает из ящика сообщения, помеченные на предыдущем этапе как подлежащие удалению, и закрывает соединение. Переход в состояние обновления в принципе возможен, только если клиент выходит из состояния транзакции по команде QUIT. Ни при каких других обстоятельствах (например, если сеанс связи прерывается по таймауту или из-за обрыва связи) переход в состояние обновления происходить не должен. Если состояние транзакции прерывается не по команде QUIT, никакие удаления не должны производиться, пометки для удаления должны быть аннулированы. К сожалению, как показывает практика,

это требование выполняется не всегда.

В ходе сеанса клиент посылает серверу команды, а сервер сообщает о результате выполнения каждой из них. Ответ состоит из индикатора состояния (status indicator) и, если нужно, дополнительной информации, отделенной пробелом. Строка ответа может содержать до 512 символов, включая последовательность CRLF, обозначающую конец строки.

Предусмотрено два индикатора состояния: «+OK» – успешное завершение и «-ERR» – неуспешное завершение. Если строка ответа не содержит дополнительной информации, то после индикатора состояния сразу должна идти последовательность CRLF. Однако некоторые клиенты ожидают пробела после индикатора состояния. Это противоречит существующим стандартам, но наличие таких клиентов все же следует принимать во внимание (см. RFC 1957 [32]).

Если команда предусматривает многострочный ответ, то индикатор состояния передается только в первой строке, а последняя строка ответа должна состоять из одной точки. Эта строка не является частью ответа, а только обозначает его завершение. Чтобы сделать возможным использование строк, состоящих из одной точки, в ответах сервера, ко всем строкам ответа, начинающимся с точки, добавляется еще одна точка, аналогично тому, как это делается при передаче текста сообщения в команде DATA протокола SMTP. Если на приемном конце в ответе сервера обнаруживается строка, начинающаяся с точки, то, если непосредственно за этой точкой стоит последовательность CRLF, строка интерпретируется как конец ответа, если же за точкой следуют любые другие символы, то ведущая точка удаляется, а строка интерпретируется как часть ответа.

Каждая команда POP3 состоит из ключевого слова и, возможно, из аргументов, разделенных пробелами. Ключевые слова состоят из 3 или 4 букв, передаваемых независимо от регистра. Аргументы могут содержать только символы ASCII. Каждый аргумент может состоять не более чем из 40 символов.

Рассмотрим команды протокола POP3, которые обязательно должны быть реализованы.

4.1. Основные команды POP3

STAT

В ответ на команду STAT сервер возвращает количество сообщений в почтовом ящике и общий размер ящика в октетах. Сообщения, помеченные для удаления, при этом не учитываются. Например, ответ

```
+OK 4 223718
```

означает, что в почтовом ящике имеется 4 сообщения общим объемом 223718 октетов.

LIST

Ответ на команду LIST без аргумента: список сообщений в почтовом ящике, содержащий их порядковые номера и размеры в октетах.

Пример

C	LIST
S	+OK 4 visible messages (223718 octets)
S	1 333
S	2 111293
S	3 111285
S	4 807
S	.

В данном примере в ящике имеется 4 сообщения, длины которых 333, 111293, 111285 и 807 октетов соответственно,

Обратите внимание, что, поскольку ответ состоит из нескольких строк, заканчивается он строкой, состоящей из одной точки.

Если в качестве аргумента команды LIST указать номер сообщения, то в ответе будет содержаться информация только об одном запрошенном сообщении.

Пример

C	LIST 2
S	+OK 2 111293

RETR

Требует в качестве аргумента номер существующего не помеченного для удаления сообщения.

В ответ сервер присылает запрошенное сообщение.

DELE

Требует в качестве аргумента номер существующего не помеченного для удаления сообщения. Указанное сообщение помечается для удаления. До конца сеанса обращаться к нему становится невозможно. После окончания диалога, когда сеанс переходит в состояние обновления, сообщение удаляется окончательно.

NOOP

На эту команду сервер должен дать положительный ответ. Никаких других действий не производится.

RSET

Сервер снимает все установленные ранее пометки для удаления.

QUIT

Завершение сеанса. Если в ходе сеанса какие-то сообщения были помечены для удаления, то после выполнения команды QUIT они удаляются из ящика.

4.2. Дополнительные возможности POP3

Кроме обязательных команд, перечисленных выше, программное обеспечение, реализующее взаимодействие по протоколу POP3, поддерживает

дополнительные возможности (capabilities), вводящие новые команды, влияющие на исполнение основных команд, облегчающие взаимодействие клиента и сервера, информирующие об особенностях реализации сервера и хранилища сообщений.

В число дополнительных возможностей входят, например, команды авторизации. Хотя бы один механизм авторизации должен быть реализован, так как доступ к почтовому ящику предоставляется только после аутентификации. Но, поскольку таких механизмов несколько, и их выбор оставляется на усмотрение разработчиков и администраторов, соответствующие команды не входят в число обязательных.

Предусмотрена команда CAPA, позволяющая клиенту получить информацию о дополнительных возможностях, реализованных на сервере, и их параметрах. Ответ на эту команду аналогичен ответу на команду EHLO протокола SMTP.

Команда CAPA

В ответ на команду CAPA сервер присылает клиенту список ключевых слов, соответствующих дополнительным возможностям протокола POP3, поддерживаемым сервером, и, при необходимости, параметры этих возможностей.

Стандартные ключевые слова не могут начинаться с буквы X. Эта буква зарезервирована для ключевых слов, соответствующих дополнительным возможностям, вводимым разработчиками и не описанным в официальных стандартах.

Эта команда может выполняться на любом этапе диалога клиента и сервера, причем результаты ее выполнения до и после авторизации могут различаться, так как параметры некоторых дополнительных возможностей могут быть различными для разных пользователей.

Команда CAPA и ряд дополнительных команд протокола POP3 определены в RFC 2449 [33].

Пример

C	CAPA
S	+OK Capability list follows
S	TOP
S	USER
S	SASL CRAM-MD5 KERBEROS_V4
S	RESP-CODES
S	LOGIN-DELAY 900
S	EXPIRE 60
S	UIDL
S	IMPLEMENTATION Shlemazle-Plotz-v302
S	.

Ниже приведены ключевые слова и описаны соответствующие дополнительные возможности протокола POP3.

USER – авторизация по паролю

Простейший механизм авторизации (RFC 1939 [6]) предусматривает передачу

имени пользователя и пароля открытым текстом.

Сначала клиент направляет серверу команду USER. В качестве аргумента команды используется имя, под которым на сервере зарегистрирован ящик, к которому запрашивается доступ. Если такой ящик существует, сервер дает положительный ответ, а клиент посылает команду PASS с паролем для доступа к данному ящику в качестве аргумента.

Пример

C	USER lonk
S	+OK Password required for lonk.
C	PASS my_password
S	+OK lonk has 4 visible messages (0 hidden) in 223718 octets.

Авторизация по зашифрованному паролю

Серьезный недостаток использования команд USER и PASS – необходимость передавать пароль по сети открытым текстом. Это дает потенциальному злоумышленнику возможность перехватить пароль. Опасность возрастает, если пользователь часто проверяет свою почту: в этом случае соединение устанавливается регулярно, с небольшим интервалом, и каждый раз по сети передается пароль.

В качестве альтернативного способа авторизации RFC 1939 [6] рекомендует команду APOP, предполагающую отправку пароля в зашифрованном виде, причем в каждом сеансе используется новый ключ.

Сервер, поддерживающий команду APOP, в начальном приветствии посылает клиенту уникальный идентификатор, содержащий метку времени в формате msg-id (RFC 822 [34]). Поскольку наличие такой метки уже свидетельствует о поддержке сервером команды APOP, ключевого слова, соответствующего этой команде, возвращаемого в ответе сервера на команду CAPA, не предусмотрено.

Клиент в качестве аргументов команды APOP передает серверу имя пользователя и 16-значную последовательность 16-ричных чисел в нижнем регистре, вычисляемую по алгоритму MD5 (RFC 1321 [35]). В качестве исходного текста для вычисления этой последовательности берется строка, состоящая из переданного сервером уникального идентификатора и пароля пользователя. Чем длиннее строка пароля, тем сложнее злоумышленнику вычислить ее на основании передаваемой в команде APOP последовательности. Рекомендуется использовать пароли, содержащие не меньше восьми символов.

Пример

S	+OK POP3 server ready <1896.697170952@dbc.mtview.ca.us>
C	APOP mrose c4c9334bac560ecc979e58001b3e22fb
S	+OK maildrop has 1 message (369 octets)

уникальный идентификатор: <1896.697170952@dbc.mtview.ca.us>;

имя пользователя: mrose;

пароль: tanstaaf.

Имя пользователя передается открытым текстом в первом аргументе команды APOP, второй аргумент: c4c9334bac560ecc979e58001b3e22fb вычисляется по алгоритму MD5 из строки <1896.697170952@dbc.mtview.ca.us>tanstaaf,

состоящей из уникального идентификатора и пароля.

SASL – авторизация с использованием SASL

Команда AUTH, предложенная в RFC 1734 [36], позволяет использовать SASL [23] для авторизации с использованием различных механизмов: Kerberos version 4, GSSAPI [37], S/Key и др., и, при необходимости, для шифрования передаваемых по протоколу POP3 данных. Аргументом команды служит ключевое слово, соответствующее избранному механизму авторизации, соответственно KERBEROS_V4, GSSAPI, SKEY и т.д. Если сервер поддерживает указанный механизм, начинается обмен данными для авторизации и согласования параметров шифрования. Данные кодируются по алгоритму Base64 [24]. Ответы сервера предваряются символами плюс и пробел: «+ ». Если клиент хочет прервать аутентификацию, он посылает символ звездочка: «*».

В строке ответа сервера на команду CAPA после ключевого слова «SASL» сервер передает список ключевых слов, соответствующих поддерживаемым сервером механизмам авторизации. Ключевые слова разделяются пробелами.

Пример

```
SASL CRAM-MD5 KERBEROS_V4
```

STLS – криптографическая защита сеанса POP3 с использованием протокола TLS

Команда AUTH, описанная выше, предоставляет простейший механизм шифрования данных, передаваемых по протоколу POP3. Более надежный способ криптографической защиты обеспечивается протоколом TLS, используемым совместно с протоколом POP3 в соответствии в RFC 2595 [38].

Для начала защищенного сеанса клиент посылает серверу команду STLS. После этого клиент и сервер согласуют параметры взаимодействия в соответствии с протоколом TLS. Выполнять команду STLS следует на этапе авторизации до передачи аутентификационных данных, что позволяет защитить эти данные от перехвата. После успешного начала защищенного сеанса клиент может аутентифицироваться.

Если реализована возможность шифрования сеанса, сервер может отклонять попытки клиента аутентифицироваться без установления защищенного сеанса. Такой запрет может быть установлен как для всех, так и для отдельных пользователей. Допустимо использование разных портов TCP для работы по протоколу POP3 с шифрованием и без.

В процессе согласования параметров шифрования клиент должен удостовериться, соответствует ли доменное имя сервера используемому им сертификату, чтобы убедиться, что данные на пути к серверу не перехватываются злоумышленниками.

Результаты выполнения команды CAPA до начала защищенного сеанса и после могут различаться.

TOP – первые строки сообщения

Дополнительная команда TOP, описанная в RFC 1939 [6], позволяет клиенту

получить заголовок и первые строки тела указанного в аргументе сообщения. Эта команда дает пользователю возможность, не скачивая все сообщение, на основании адреса отправителя и темы, указанных в заголовке, а также из первых строк письма, принять решение, получать данное сообщение сейчас, сделать это позже или удалить, не читая.

Формат команды

TOP номер_сообщения число_строк

Если второй аргумент больше, чем число строк в теле сообщения, то клиент получает сообщение целиком.

UIDL – уникальный идентификатор сообщения

Сервер присваивает каждому сообщению идентификатор, уникальный в пределах почтового ящика. Клиент может использовать уникальные идентификаторы для того, чтобы найти сообщение в ящике, так как, в отличие от порядкового номера, идентификатор не меняется при удалении сообщений.

Так, для того, чтобы определить, есть ли в почтовом ящике новые письма, клиент может запросить список идентификаторов сообщений в ящике и сравнить его со списком идентификаторов уже полученных сообщений.

Доступ к уникальным идентификаторам сообщений предоставляет дополнительная команда UIDL (RFC 1939 [6]). Если она используется без аргументов, в ответ на нее сервер присылает порядковые номера и уникальные идентификаторы всех сообщений в почтовом ящике, непомеченных для удаления. Если в качестве аргумента указан номер сообщения в ящике, сервер возвращает порядковый номер и уникальный идентификатор только этого сообщения.

Пример

C	UIDL
S	+OK
S	1 whqtswO00WBw418f9t5JxYwZ
S	2 QhdPYR:00WBw1Ph7x7
S	.
C	UIDL 2
S	+OK 2 QhdPYR:00WBw1Ph7x7

RESP-CODES – расширенные коды ответов

Наличие в ответе команды CAPA строки «RESP-CODES» не свидетельствует о поддержке какой-либо команды. Эта дополнительная возможность позволяет серверу использовать расширенные коды ответов.

Расширенные коды ответов передаются после стандартного индикатора состояния: «+OK» или «–ERR» и заключаются в квадратные скобки. Они могут иметь иерархическую структуру, т. е. могут состоять из нескольких кодов, каждый из которых уточняет предыдущие. В этом случае коды разделяются косой чертой: «/». Необязательный текст, поясняющий ответ сервера, может следовать после расширенного кода.

RFC 2449 [33] описывает только один расширенный код ответа: «[IN-USE]».

Этот код передается после успешной аутентификации и свидетельствует о том, что доступ к почтовому ящику, тем не менее, невозможен, поскольку в настоящее время ящик уже используется, возможно, другим сеансом POP3.

Пример

C	APOP mrose c4c9334bac560ecc979e58001b3e22fb
S	-ERR [IN-USE] Do you have another POP session running?

LOGIN-DELAY – определение минимального интервала времени между сеансами

Многие пользователи часто проверяют свою почту, инициируя все новые сеансы POP3. Это приводит к неоправданному росту нагрузки на сервер, ведь даже если пользователь не производит никаких манипуляций с ящиком, только проверяя наличие новой почты и убеждаясь, что ее нет, необходимо выполнить как минимум авторизацию клиента.

Чтобы уменьшить нагрузку на сервер, можно установить время, в течение которого клиенту не разрешается инициировать повторный сеанс.

Если оно установлено, в ответе сервера на команду CAPA должна содержаться строка вида:

LOGIN-DELAY время_задержки

где время_задержки – интервал, в течение которого не разрешается начинать повторный сеанс POP3, всекундах. Время отсчитывается от момента положительного ответа сервера на одну из команд авторизации (PASS, APOP или AUTH) до следующей попытки авторизации того же пользователя.

Время задержки может различаться для разных пользователей. В этом случае, если команда CAPA выполняется на этапе авторизации, когда вызывающий пользователь еще не известен, рекомендуется в ответе давать строку:

LOGIN-DELAY время_задержки USER

где время_задержки – максимальное из всех установленных на сервере времен задержки. Пометка «USER» свидетельствует о том, что на самом деле время задержки зависит от пользователя и может быть уточнено после авторизации. Выполнив команду CAPA на этапе транзакции, клиент может получить точное время задержки именно для данного пользователя.

PIPELINING - группирование команд

Аналог одноименного расширения ESMTP. Эта дополнительная возможность не вводит новых команд, но позволяет клиенту посылать серверу команды, не дожидаясь ответов на предыдущие. Это ускоряет обмен данными (клиент может посылать команды DELE в то время, когда сервер шлет ему запрошенные ранее сообщения).

EXPIRE – ограничение времени хранения сообщений на сервере

Сообщения могут храниться на сервере, пока пользователи сами их не удалят. К сожалению, далеко не всегда можно ожидать от пользователей бережного отношения к дисковому пространству сервера. Трудно потребовать от людей

регулярно забирать почту и удалять с сервера старую корреспонденцию, поэтому многие администраторы прибегают к использованию технических средств.

Программное обеспечение хранилища сообщений может само удалять из ящиков старые письма. Администратор может определять довольно сложные правила удаления: они могут различаться для разных пользователей, время хранения может отсчитываться от разных моментов: от поступления сообщения, от первого сеанса POP3 после поступления сообщения, от первого прочтения командой RETR или TOP и т.д. Если подобные правила определены, сервер POP3 должен информировать об этом клиента в ответе на команду CAPA.

Соответствующая строка содержит ключевое слово «EXPIRE» и параметр, обозначающий время хранения сообщения в днях. Если правилами предусмотрены разные значения времени хранения при разных обстоятельствах, выдается минимальное из них.

Параметр может принимать значения:

0 – прочитанные сообщения на сервере не хранятся;

целое число – минимальное время хранения сообщения в днях;

NEVER – сообщения автоматически не удаляются.

Если для разных пользователей предусмотрены разные правила удаления старых сообщений, то на запрос CAPA в состоянии авторизации сервер возвращает минимальное значение параметра EXPIRE для всех пользователей, за которым следует модификатор «USER». Это означает, что после авторизации клиенту следует повторить команду CAPA, чтобы получить более точные данные об удалении с сервера старой корреспонденции именно этого пользователя.

Примеры

EXPIRE 5 USER	– сообщения на сервере хранятся не менее пяти дней, но эта цифра может различаться для разных пользователей. После авторизации клиенту следует повторить запрос, чтобы узнать, сколько дней хранится корреспонденция данного пользователя
EXPIRE 30	– сообщения на сервере хранятся не менее тридцати дней
EXPIRE NEVER	– сообщения автоматически не удаляются
EXPIRE 0	– прочитанные сообщения удаляются с сервера автоматически

Следует учитывать, что удаление устаревших писем производит программное обеспечение хранилища сообщений, а не сервер POP3. Настройка этих программных продуктов зависит от реализации, и в общем случае параметры EXPIRE могут не соответствовать реальным установкам хранилища сообщений, если администраторы не взяли на себя труд настроить их соответствующим образом.

IMPLEMENTATION – информация о сервере

Строка ответа на команду CAPA IMPLEMENTATION содержит версию программного обеспечения сервера POP3. Эта информация может быть передана также в строке приветствия сервера при установлении соединения. Но из соображений безопасности эти сведения не желательно делать доступными не аутентифицированным пользователям. Потому строка IMPLEMENTATION

обычно передается клиенту только в состоянии транзакции.

Строка IMPLEMENTATION предусматривает только один параметр, потому информация о сервере, передаваемая в ней, не должна содержать пробелов.

4.3. Пример сеанса POP3

S	+OK ready <6584.1077893295@myhost.ru>	В начальном приветствии сервера присутствует уникальный идентификатор, что свидетельствует о поддержке сервером команды APOP
C	CAPA	Запрос возможностей сервера
S	+OK Capability list follows	
S	TOP	Поддерживается команда TOP
S	USER	Поддерживается авторизация открытым паролем
S	LOGIN-DELAY 0	На сервере может быть установлен минимальный период времени между сеансами одного пользователя, но в настоящее время такого ограничения нет
S	EXPIRE 0	Прочитанная почта не хранится на сервере (если эта возможность правильно настроена)
S	UIDL	Поддерживается команда UIDL
S	RESP-CODES	Поддерживаются расширенные коды ответов
S	X-LOCALTIME Fri, 27 Feb 2004 15:48:46 +0100	Нестандартное сообщение, установленное разработчиком сервера
S	.	Конец ответа
C	USER lonk	Имя пользователя: lonk
S	+OK Password required for lonk.	Имя принято, ожидается ввод пароля
C	PASS my_passwd	Пароль пользователя lonk: my_passwd
S	+OK lonk has 3 visible messages in 223385 octets.	Доступ к почтовому ящику разрешен. Имеется 3 сообщения общим объемом 223385 октетов
C	CAPA	Повторный запрос возможностей сервера
S	+OK Capability list follows	
S	TOP	
S	USER	
S	LOGIN-DELAY 0	
S	EXPIRE 0	
S	UIDL	
S	RESP-CODES	

S	X-LOCALTIME Fri, 27 Feb 2004 15:49:04 +0100	
S	IMPLEMENTATION Qpopper-version-4.0.3	Строка IMPLEMENTATION доступна только авторизованным пользователям
S	.	Конец ответа
C	LIST	Запрос списка сообщений
S	+OK 3 visible messages (223385 octets)	3 сообщения, 223385 октетов
S	1 111293	Размер первого сообщения: 111293 октета
S	2 111285	Размер второго сообщения: 111295 октетов
S	3 807	Размер третьего сообщения: 807 октетов
S	.	Конец списка
C	UIDL	Запрос списка идентификаторов сообщений
S	+OK uidl command accepted.	
S	1 #i,"!;)L"!#`S"!3SN"!	Идентификаторы
S	2 =>X!!\e\!!WcE"!&Qc"!	
S	3 2A*!!j#~!!D@L!!Jj&"!	
S	.	Конец списка
C	TOP 3 1	Запрос первой строчки сообщения 3
S	+OK Message follows	
S	Return-Path: <lonk@pds.sut.ru>	Передается заголовок сообщения Пустая строка – конец заголовка
S	…	
S	Status: RO	
S		
S	Привет!	Первая строка сообщения
S	.	Конец ответа
C	DELE 3	Удалить сообщение 3
S	+OK Message 3 has been deleted.	Сообщение удалено (на самом деле только помечено для удаления)
C	STAT	Запрос количества сообщений
S	+OK 2 222578	Осталось 2 сообщения
C	RETR 3	Запрос сообщения 3
S	-ERR Message 3 has been deleted.	Невозможно получить сообщение, помеченное для удаления
C	RSET	Отмена всех удалений
S	+OK Maildrop has 3 messages (223385 octets)	В ящике снова 3 сообщения
C	RETR 3	Запрос сообщения 3
S	+OK 807 octets	
S	Return-Path: <lonk@pds.sut.ru>	Передается заголовок сообщения
S	…	
S	Status: RO	
S		
S	Привет!	Передается полный текст сообщения
S	Это тестовое сообщение	
S	для проверки возможностей	

S	протокола POP3.	
S	.	
C	QUIT	Конец работы
S	+OK Pop server at myhost.ru signing off.	

4.4. Контрольные вопросы

1. Для чего предназначен протокол POP3? На каких участках системы электронной почты он используется?
2. Какой порт TCP использует протокол POP3?
3. Из каких этапов состоит сеанс POP3?
4. В каком состоянии сеанса сервер не принимает команд от клиента?
5. В каком состоянии сеанса происходит удаление сообщений из почтового ящика?
6. В каком направлении происходит передача постовых сообщений: от сервера к клиенту или от клиента к серверу?
7. Что такое индикатор состояния? Как обозначается конец ответа, если ответ состоит из нескольких строк?
8. Возможен ли доступ к почтовому ящику без предварительной авторизации? Почему команды авторизации не входят в число обязательных?
9. Какие способы аутентификации используются протоколом POP3? Какие команды используются для аутентификации?
10. Для чего используется команда CAPA? В каких состояниях сеанса она выполняется, и чем различаются результаты ее выполнения в зависимости от того, в каком состоянии она выполнена?
11. С помощью каких команд можно получить текст сообщения?

4.5. Практическое задание

При помощи программы Telnet соединитесь с сервером POP3, обеспечивающим доступ к вашему почтовому ящику.

Выполните команду CAPA. Какие дополнительные возможности реализованы на этом сервере POP3?

Произведите авторизацию при помощи команд USER и PASS.

Повторно выполните команду CAPA. Что изменилось в ответе сервера?

Выполните команду LIST. Сколько писем находится в вашем ящике?

Прочитайте первую строчку одного из писем. Получите то же самое письмо целиком. Какие команды можно использовать для этого?

Пометьте одно из писем для удаления. Убедитесь, что письмо действительно стало недоступным. С помощью каких команд можно это сделать?

Снимите пометку об удалении. Убедитесь, что сообщение вновь стало доступным.

Завершите сеанс.

5. ПРОТОКОЛ IMAP

Область применения протокола IMAP (Internet Message Access Protocol) аналогична области применения протокола POP3: он тоже предназначен для получения почты и используется на участке между MUA получателя и хранилищем сообщений. IMAP предоставляет более широкие возможности работы с почтовыми ящиками, чем POP3: он позволяет работать с несколькими почтовыми ящиками на одном или нескольких серверах IMAP как с файлами и каталогами на собственной машине пользователя. Обычно почтовые ящики сервера IMAP действительно представляют собой файлы в специальном каталоге сервера и его подкаталогах.

Сервер IMAP способен анализировать сообщение: выделять заданные поля заголовка и разбирать структуру тела сообщения.

В отличие от серверов POP3, серверы IMAP не должны блокировать ящик на время сеанса – несколько клиентов могут одновременно работать с одним и тем же ящиком. Это связано с рядом проблем, особенно, если информация в ящиках доступна для записи (см. RFC 2180 [39]).

Довольно часто IMAP используется в организациях, где пользователям нужно предоставить возможность совместно работать с одними и теми же почтовыми ящиками. Он удобен для работы с новостями USENET. Также протокол можно использовать для работы с личными каталогами и файлами пользователя, расположенными на сервере.

Впрочем, для этой цели целесообразнее использовать протоколы, специально предназначенные для этого.

Хотя программное обеспечение, реализующее протокол IMAP, постоянно совершенствуется, IMAP менее защищен, чем POP3. Возможность хранить сообщения на сервере может стать причиной злоупотреблений со стороны пользователей, которые будут переполнять хранилище сообщений ненужной информацией.

Протокол IMAP предполагает в основном работу пользователей с почтовыми ящиками непосредственно на сервере, в отличие от протокола POP3, который ориентирован на то, что клиент забирает пришедшую почту и разбирает ее уже на своей машине (RFC 1733 [40]). Это делает IMAP неудобным для пользователей, подключающихся к сети кратковременно, только для того, чтобы получить или отослать почту. Во всяком случае, многие преимущества IMAP таким пользователям недоступны. При работе по протоколу IMAP клиенту желательно иметь доступ к сети все время, пока он работает с почтой.

Протокол IMAP позволяет пользователю работать с множеством почтовых ящиков, расположенных, возможно, на разных серверах.

Допускается иерархическое расположение почтовых ящиков в каталогах и их подкаталогах, причем имена каталогов и почтовых ящиков сами по себе не различаются. Почтовый ящик может быть только конечным элементом иерархической структуры, он не может содержать никаких нижестоящих элементов. Каталог может содержать подкаталоги и почтовые ящики, но он не содержит сообщений и не может быть выбран командой SELECT.

Символ, используемый в качестве иерархического разделителя, может различаться в зависимости от используемого на сервере программного обеспечения. Обычно это косая черта: «/», если сервер работает под управлением операционной системы, совместимой с UNIX, обратная косая черта: «\» для операционной системы Windows и точка для имен групп новостей USENET.

Допускается использование различных пространств имен почтовых ящиков и, соответственно, разных иерархических разделителей (например, если сервер IMAP предоставляет доступ к ящикам, расположенным в каталогах файловой системы UNIX и к группам новостей USENET, то в первом случае в качестве иерархического разделителя используется косая черта, а во втором – точка). Чтобы использовать и различать разные пространства имен на одном сервере IMAP, имена, принадлежащие каждому из используемых пространств, должны начинаться с некоторого префикса, обычно начинающегося символом «#». Естественно, запросы, в которых путь к ящику начинается с одного префикса, будут давать отличные результаты от таких же запросов, начинающихся с другого префикса. Используемое по умолчанию пространство имен может префикса не иметь.

Клиент может выяснить, какие именно пространства имен для почтовых ящиков каких типов поддерживаются данным сервером IMAP, если сервер поддерживает расширение NAMESPACE. Префикс и иерархический разделитель конкретного имени почтового ящика или каталога можно выяснить при помощи команды LIST.

Большие возможности, предоставляемые протоколом IMAP, создают большие сложности при разработке, настройке и эксплуатации серверов и клиентов. Некоторые рекомендации по этим вопросам даны в RFC 2683 [41]. В общем случае можно посоветовать использовать протокол IMAP только в том случае, если возможности протокола POP3 не достаточны для работы пользователей с их почтовыми ящиками.

Последняя на момент написания данного пособия версия протокола IMAP: IMAP4rev1 (RFC 3501 [7]).

Сервер IMAP ожидает соединения от клиентов на порту TCP 143. После установления соединения сервер посылает свое приветствие клиенту, и начинается диалог, в котором клиент посылает серверу команды, а сервер сообщает о результатах их выполнения или присылает затребованную клиентом информацию. Как и сеанс POP3, сеанс IMAP делится на несколько состояний (states). Допустимый набор команд зависит от текущего состояния сеанса.

Рис. 6. Переходы между состояниями сеанса IMAP:
1 – соединение без предварительной аутентификации,
2 – соединение с предварительной аутентификацией,
3 – отвергнутое соединение,
4 – успешная аутентификация,
5 – успешное выполнение команды SELECT или EXAMINE,
6 – команда CLOSE или неудачное завершение команды SELECT или EXAMINE,
7 – команда LOGOUT или потеря связи

Сеанс IMAP (рис. 6) может находиться в одном из следующих состояний:

– неаутентифицированное состояние (Not Authenticated State): клиент должен пройти процедуру аутентификации прежде, чем сможет выполнять большинство команд;

– аутентифицированное состояние (Authenticated State): клиент аутентифицирован и должен выбрать почтовый ящик, прежде чем сможет работать с отдельными сообщениями;

– выбранное состояние (Selected State): почтовый ящик выбран;

– состояние выхода (Logout State): сеанс завершается.

5.1. Команды клиента и ответы сервера IMAP

Команда клиента состоит из идентификатора (ярлыка) – короткой строки, состоящей из букв и цифр, не повторяющейся в других командах в течение всего сеанса. За ярлыком следует сама команда и ее аргументы. Регистр символов в названиях команд, как и в большинстве аргументов, как правило, не имеет значения.

Кроме стандартных команд, которые обязательно должны поддерживаться, имеются также дополнительные команды, описанные в стандартах и поддерживаемые серверами IMAP как элементы расширений. Разработчики также могут добавлять в своих реализациях новые команды. Названия таких нестандартизированных команд должны начинаться с буквы X. Имена стандартных команд с буквы X начинаться не могут.

Все ответы сервера начинаются с метки, после которой следует отделенный пробелом текст.

В ответах сервера, сообщающих об исполнении команд, в качестве метки используется ярлык соответствующей команды. Это помеченные (tagged) ответы.

За ним следует одно из ключевых слов:

OK (успешное выполнение);

NO (невыполнение);

BAD (ошибка в команде).

Ответы, содержащие информацию, запрошенную клиентом или посылаемую сервером без запроса, начинаются с метки «*», такие ответы называются непомеченными (untagged). После метки следуют номер опрашиваемого сообщения и ключевое слово или только ключевое слово. Ключевых слов для непомеченных ответов предусмотрено значительно больше, чем для помеченных, обычно они соответствуют имени команды или характеру передаваемой клиенту информации.

Каждая команда должна получить один помеченный ответ сервера, свидетельствующий о возможности или невозможности исполнения команды, и, возможно, один или несколько непомеченных ответов, содержащих запрошенную или дополнительную информацию.

Клиент может послать несколько команд подряд, не дожидаясь ответа на каждую из них, если принятие решения о следующей команде не зависит от ответа на предыдущую. Сервер, получив несколько команд подряд, может выполнять их параллельно, если для успешного выполнения следующей команды не требуется выполнение предыдущей. Ответы в этом случае могут поступать в порядке, отличном от того, в каком команды посылались. Соответствие помеченных ответов посланным командам клиент определяет по ярлыкам.

Надо соблюдать осторожность и следить за тем, чтобы при одновременном выполнении команд не возникало неоднозначностей (при удалении сообщения из ящика изменяется нумерация оставшихся сообщений, что может привести к неоднозначной интерпретации команды, одновременно обращающейся к другому сообщению в том же почтовом ящике).

Информация, передаваемая в команде или в непомеченном ответе, не обязательно должна укладываться в одну строку. Данные, которые передаются как продолжение команды или непомеченного ответа, называются литералом. На конце строки, которая должна быть продолжена литералом, ставится его размер в октетах, заключенный в фигурные скобки.

Сервер может передавать литерал, не дожидаясь разрешения клиента, клиент, прежде чем передавать литерал, должен дождаться разрешения – строки, начинающейся с метки «+».

Пример

```
C | A001 LOGIN {11}
S | + Ready for additional command text
C | FRED FOOBAR {7}
S | + Ready for additional command text
C | Fat man
```

Расширение LITERAL+ (RFC 2088 [42]) позволяет клиенту отправлять серверу продолжение команды, не дожидаясь разрешения. В этом случае между числом и закрывающей фигурной скобкой должен стоять плюс «+».

5.2. Команды, допустимые при любом состоянии сеанса IMAP

CAPABILITY

В ответ на эту команду сервер присылает непомеченную строку с ключевым словом CAPABILITY, содержащую список поддерживаемых возможностей (расширений) и их параметров. В число возможностей входит в частности поддерживаемая версия протокола IMAP – IMAP4rev1 и механизмы аутентификации (RFC 2595 [38]).

AUTH=механизмы_аутентификации

Возможности IMAP описываются в различных RFC или могут вводиться разработчиками. В последнем случае их названия должны начинаться с буквы X. Названия стандартных возможностей с этой буквы начинаться не могут.

NOOP

Эта команда не выполняет никаких действий, но сбрасывает таймер неактивности, что позволяет избежать разрыва соединения по таймауту. Кроме того, при определенных обстоятельствах эта или другая команда служит неявным запросом информации об обновлениях, произошедших на сервере. Таким образом, с помощью команды NOOP можно периодически проверять, не появились ли новые сообщения или не изменился ли статус старых. Это можно делать и с помощью других команд, но, поскольку команда NOOP не производит никаких других действий, использование ее с описанной целью предпочтительнее.

LOGOUT

Конец сеанса.

5.3. Команды неаутентифицированного состояния

Клиент должен успешно аутентифицироваться, чтобы перейти в следующее состояние. Алгоритмы аутентификации для протокола IMAP не отличаются принципиально от рассматривавшихся в предыдущих главах алгоритмов аутентификации для протоколов SMTP и POP3.

Информация о поддерживаемых способах аутентификации передается сервером клиенту в ответе на команду CAPABILITY. Так запись STARTTLS свидетельствует о поддержке одноименной команды (RFC 2595 [38]), LOGINDISABLED – расширение, исключающее аутентификацию с использованием незашифрованных имени и пароля, параметры AUTH указывают, какие механизмы аутентификации с использованием SASL поддерживает сервер.

Подробно о механизмах аутентификации IMAP см. RFC 1731 [43].

Серверы IMAP могут допускать анонимный доступ к некоторым почтовым ящикам. Анонимный пользователь регистрируется под именем anonymous, в качестве пароля используется адрес электронной почты пользователя, имя его домена, произвольный набор символов или пустая строка. Анонимный доступ возможен как при передаче пароля открытым текстом, так и с использованием SASL. В последнем случае, в соответствии с RFC 2245 [44], в ответе на команду CAPABILITY клиент информируется о наличии механизма аутентификации ANONYMOUS:

```
AUTH=ANONYMOUS
```

При аутентификации с использованием этого механизма клиент должен передать серверу некоторую информацию о себе. Характер этой информации, как и ее достоверность не имеют значения, в принципе, это может быть произвольная, даже пустая, строка. Поскольку анонимным доступом может воспользоваться любой желающий, а идентифицировать такого пользователя не представляется возможным, следует проявлять осторожность, предоставляя анонимный доступ. Возможности анонимного клиента должны быть строго ограничены, как правило, он не получает прав на изменение какой-либо информации на сервере.

STARTTLS

Перевод сеанса в защищенный режим. После получения сервером команды STARTTLS клиент и сервер согласовывают параметры дальнейшего взаимодействия. Все данные, которыми обмениваются клиент и сервер после успешного завершения этой команды, передаются в зашифрованном виде. Однако аутентификация при помощи этой команды не производится, сеанс остается в неаутентифицированном состоянии.

LOGIN регистрационное_имя_пользователя пароль

Аутентификация при помощи регистрационного имени и пароля, передаваемых открытым текстом.

AUTHENTICATE механизм

Передача зашифрованных аутентификационных данных с использованием SASL.

5.4. Команды аутентифицированного состояния

В аутентифицированном состоянии клиент производит различные манипуляции с почтовыми ящиками.

SELECT имя_ящика

Открывает доступ к указанному почтовому ящику. Сеанс переходит в состояние выбора, после этого клиент может работать с отдельными сообщениями в ящике.

В ответ на эту команду сервер присылает ряд непомеченных ответов, содержащих информацию о почтовом ящике: количество сообщений, список допустимых флагов (см. описание команды APPEND), количество новых сообщений, номер первого непрочитанного сообщения, идентификатор почтового ящика.

EXAMINE имя_ящика

Аналогично команде SELECT, но почтовый ящик открывается только для чтения.

CREATE имя_объекта

Создает новый почтовый ящик или каталог.

Если объект создается не в корневом каталоге, то надо указать путь к нему.

Если на конце указанного имени стоит символ, используемый в качестве иерархического разделителя, создается каталог.

Пример

```
C | A003 CREATE owatagusiam/
S | A003 OK CREATE completed
C | A004 CREATE owatagusiam/blurdybloop
S | A004 OK CREATE completed
```

В этом примере, если косая черта (/) служит на сервере иерархическим разделителем, создается каталог owatagusiam и в нем почтовый ящик blurdybloop. В противном случае создаются два почтовых ящика в текущем каталоге.

DELETE имя_ящика

Удаляет указанный почтовый ящик. Эта же команда удаляет также и каталоги, если они не содержат почтовые ящики.

RENAME имя_ящика новое_имя_ящика

Переименование почтового ящика.

SUBSCRIBE имя_ящика

Почтовый ящик помечается как «активный». Эта пометка используется для вывода списка почтовых ящиков при помощи команды LSUB.

UNSUBSCRIBE имя_ящика

Снимает с почтового ящика пометку «активный». Эта пометка может быть снята с почтового ящика только при помощи команды UNSUBSCRIBE. Даже если ящик больше не существует, это не может само по себе стать причиной снятия пометки «активный».

LIST путь_к_ящику имя_ящика

Возвращает список каталогов и почтовых ящиков, соответствующих указанным аргументам.

В имени ящика могут использоваться групповые символы:

«*», обозначающий любые символы,

«%», обозначающий любые символы кроме иерархических разделителей.

Если имя ящика само по себе содержит полный путь к ящику, то первый аргумент игнорируется.

В ответ сервер присылает одну или несколько непомеченных строк с ключевым словом LIST. Каждая из них содержит атрибуты, если таковые имеются, иерархический разделитель и имя почтового ящика или каталога, отвечающего заданным в аргументах условиям. Атрибуты заключаются в скобки, иерархический разделитель – в кавычки.

RFC 3501 [7] предусматривает четыре атрибута:

\Noinferiors – объект нижнего уровня иерархии – почтовый ящик, не каталог;

\Noselect – объект – каталог, он не может быть выбран командой SELECT;

\Marked – почтовый ящик отмечен как представляющий интерес, скорее всего это означает, что со времени последнего обращения к ящику туда были добавлены новые сообщения;

\Unmarked – со времени последнего обращения в почтовый ящик не поступило новых сообщений.

В расширении CHILDREN (RFC 3348 [45]) предусмотрены еще два атрибута, позволяющие определить, содержит ли каталог почтовые ящики или подкаталоги:

\HasChildren – каталог содержит почтовые ящики или подкаталоги;

\HasNoChildren – объект не содержит почтовых ящиков или подкаталогов, доступных пользователю, аутентифицированному в данном сеансе. Этот атрибут не следует путать с атрибутом \Noinferiors, означающим, что никакие нижестоящие иерархические элементы не могут быть созданы. Атрибут \HasNoChildren означает только то, что такие элементы в настоящее время не существуют. Такой атрибут могут иметь как почтовые ящики, так и пустые каталоги.

Оставив пустым второй аргумент команды LIST, можно выяснить используемый в первом аргументе префикс пространства имен. В этом случае ответ сервера вместо имени ящика будет содержать имя корневого каталога, то есть префикс пространства имен с иерархическим разделителем на конце.

Примеры

C	A101 LIST "" ""	Опрашиваются иерархический разделитель пространства имен, используемого по умолчанию. Это «/»
S	* LIST (\Noselect) "/" ""	
S	A101 OK LIST Completed	

C	A102 LIST #news.comp.mail.misc ""	Опрашиваются иерархический разделитель и префикс пространства, к которому принадлежит имя #news.comp.mail.misc. Иерархический разделитель: «.», префикс: #news
S	* LIST (\Noselect) "." #news.	
S	A102 OK LIST Completed	

C	A103 LIST /usr/staff/jones ""	Для имени /usr/staff/jones иерархический разделитель: «/», корневой каталог: /
S	* LIST (\Noselect) "/" /	
S	A103 OK LIST Completed	

C	A202 LIST ~/Mail/ %	Список объектов в каталоге ~/Mail/: каталог foo и почтовый ящик meetings
S	* LIST (\Noselect) "/" ~/Mail/foo	
S	* LIST () "/" ~/Mail/meetings	
S	A202 OK LIST completed	

Если оба аргумента команды LIST пустые, сервер возвращает только используемый по умолчанию иерархический разделитель.

Если вместо иерархического разделителя в ответе указана пустая строка: "", это означает, что иерархия имен на данном сервере вообще не предусмотрена: все почтовые ящики могут быть расположены только в одном каталоге.

LSUB путь_к_ящику имя_ящика

Команда LSUB аналогична команде LIST, но она возвращает только имена почтовых ящиков с пометкой «активный».

STATUS имя_ящика (имена_элементов)

Возвращает запрошенные элементы информации об указанном почтовом ящике. Имена элементов информации разделяются пробелами и все вместе заключаются в скобки.

Предусмотрены следующие имена элементов информации:

MESSAGES – общее количество сообщений в ящике;

RECENT – количество новых сообщений;

UIDNEXT – уникальный идентификатор, который изменяется всякий раз, когда в почтовый ящик помещается новое сообщение, используется для того, чтобы определить, появились ли в ящике новые сообщения за время, прошедшее после предыдущей проверки;

UIDVALIDITY – уникальный идентификатор почтового ящика;

UNSEEN – количество сообщений, не помеченных как прочитанные.

Ответ представляет собой непомеченную строку с ключевым словом STATUS, за которым следует имя опрашиваемого почтового ящика, после которого в скобках перечисляются имена запрошенных элементов и их числовые значения.

Пример

C	A042 STATUS blurdybloop (UIDNEXT MESSAGES)
S	* STATUS blurdybloop (MESSAGES 231 UIDNEXT 44292)
S	A042 OK STATUS completed

APPEND имя_ящика (флаги_сообщения) метка_времени сообщение

Добавляет сообщение в конец указанного почтового ящика. В качестве аргументов указываются имя ящика, флаги сообщения (не обязательно), метка времени (не обязательно) и само сообщение – заголовок и тело.

Существуют следующие флаги сообщений:

\Seen – прочитано;

\Answered – написан ответ;

\Flagged – срочное;

\Deleted – помечено для удаления;

\Draft – черновик;

\Recent – новое сообщение, оно поступило в почтовый ящик после окончания прошлого сеанса.

Если в команде указаны флаги, то они устанавливаются для добавляемого сообщения. В любом случае для сообщения устанавливается флаг \Recent.

Если в команде задана метка времени, то это время будет установлено в качестве времени создания сообщения, в противном случае за время создания принимается текущее время.

Поскольку сообщение состоит не из одной строки, используются литералы.

Пример

C	A003 APPEND saved-messages (\Seen) {247}
S	+ Ready for literal data
C	Date: Mon, 7 Feb 1994 21:52:25 -0800 (PST)
C	From: Fred Foobar <foobar@Blurdybloop.COM>
C	Subject: afternoon meeting
C	To: mooch@owatagu.siam.edu
C	Message-Id: <B27397-0100000@Blurdybloop.COM>
C	
C	Hello Joe, do you think we can meet at 3:30 tomorrow?
S	A003 OK APPEND completed

Расширение MULTIAPPEND (RFC 3502 [46]) позволяет одной командой добавлять в почтовый ящик несколько сообщений.

5.5. Команды выбранного состояния

В выбранном состоянии клиент может выполнять те же действия, что и в аутентифицированном состоянии, и производить манипуляции с сообщениями в выбранном почтовом ящике.

CHECK

Команда производит проверку выбранного почтового ящика, характер которой зависит от реализации программного обеспечения сервера.

CLOSE

Выбранный почтовый ящик закрывается. При этом, если почтовый ящик был открыт для чтения и записи, все помеченные для удаления сообщения в ящике удаляются. Сеанс возвращается в аутентифицированное состояние.

EXPUNGE

Из выбранного почтового ящика удаляются все помеченные для удаления сообщения. Для каждого удаляемого сообщения посылается непомеченный ответ, содержащий номер сообщения и ключевое слово EXPUNGE.

SEARCH кодировка_символов критерии_поиска

Поиск в выбранном почтовом ящике сообщений, отвечающих указанным критериям поиска.

Необязательный первый аргумент команды состоит из слова «CHARSET» и названия кодировки, используемой в критериях поиска. Кодировку нужно указывать, только если она отличается от стандартной US-ASCII.

Второй аргумент содержит один или более критериев поиска.

Критерий поиска состоит из ключевого слова и аргумента (табл. 3).

Таблица 3

Ключевое слово	Аргумент	Значение
	x:y	Сообщения с номерами в интервале от x до y
ALL		Все сообщения
BCC CC FROM SUBJECT TO	Строка	Сообщения, содержащие указанную строку в соответствующем поле заголовка
HEADER	Имя_поля Строка	Сообщения, содержащие указанную строку в указанном поле заголовка
BODY	Строка	Сообщения, в теле которых содержится указанная строка
TEXT		Сообщения, содержащие указанную строку в заголовке или в теле
SENTON	Дата	Сообщения, у которых в поле Date заголовка записано указанное число
SENTSINCE	Дата	Сообщения, у которых в поле Date заголовка записано число, более позднее, чем указанное

SENTBEFORE	Дата	Сообщения, у которых в поле Date заголовка записано число, более раннее, чем указанное
ON	Дата	Сообщения, с указанной датой создания
SINCE	Дата	Сообщения, созданные позднее указанной даты
BEFORE	Дата	Сообщения, созданные раньше указанной даты
ANSWERED DELETED DRAFT FLAGGED RECENT SEEN		Сообщения, для которых установлен соответствующий флаг
UNANSWERED UNDELETED UNDRAFT UNFLAGGED UNSEEN		Сообщения, для которых не установлен соответствующий флаг
KEYWORD	Флаг	Сообщения, для которых установлен указанный флаг
UNKEYWORD	Флаг	Сообщения, для которых не установлен указанный флаг
NEW		Сообщения с установленным флагом \Recent и не установленным флагом \Seen
OLD		Сообщения, для которых не установлен флаг \Recent
SMALLER	Число	Сообщения, размер которых меньше указанного числа октетов
LARGER	Число	Сообщения, размер которых больше указанного числа октетов
NOT	Критерий	Сообщения, не отвечающие указанному критерию поиска
OR	Критерий Критерий	Сообщения, отвечающие одному из указанных критериев поиска
UID	x:y	Сообщения с уникальным идентификатором в интервале от x до y

При поиске строки регистр символов всегда игнорируется.

Если задано несколько критериев поиска, то ищутся сообщения, отвечающие всем критериям. Например, строка

```
DELETED FROM "SMITH" SINCE 1-Feb-1994
```

означает поиск всех помеченных для удаления сообщений со строкой «SMITH» в адресе отправителя, помещенных в почтовый ящик после 1.02.94.

Критерии также можно объединять, заключая в скобки. Это бывает удобно при использовании ключевых слов NOT и OR.

Непомеченные ответы на команду SEARCH содержат номера сообщений, отвечающих указанным критериям.

Примеры

Поиск непрочитанных сообщений, пришедших после 1.02.1994, в адресе отправителя которых нет строки «Smith» (этому критерию отвечают три

сообщения с номерами 2, 84, 882).

C	A282 SEARCH UNSEEN SINCE 1-Feb-1994 NOT FROM "Smith"
S	* SEARCH 2 84 882
S	A282 OK SEARCH completed

Поиск сообщений, содержащих строку «string not in mailbox» (таких нет).

C	A282 SEARCH UNSEEN SINCE 1-Feb-1994 NOT FROM "Smith"
S	* SEARCH 2 84 882
S	A282 OK SEARCH completed

Поиск сообщений, содержащих строку «что-нибудь» на русском языке в кодировке KOI-8-R (найдено одно сообщение с номером 43).

C	A284 SEARCH CHARSET KOI-8-R TEXT {10}
C	что-нибудь
S	* SEARCH 43
S	A284 OK SEARCH completed

FETCH x:y имя_элемента_сообщения_или_макрос

Сервер возвращает информацию, относящуюся к сообщениям, обозначенным первым аргументом команды. Это может быть либо число, обозначающее номер сообщения, либо интервал от номера x до номера y, записанный в формате x:y.

Во втором аргументе перечисляются запрашиваемые информационные элементы (табл. 4).

Таблица 4

Информационный элемент	Значение
BODY	Структура сообщения. Может различаться в зависимости от характера содержимого. Описание структуры сообщения, не содержащего никаких вложений, состоит из следующих подэлементов: - тип содержимого*; - подтип содержимого*; - параметры тела сообщения, представляющие собой пары строк, где первая строка пары – название параметра, а вторая строка – значение, например запись "foo" "bar" "baz" "rag" следует интерпретировать как foo=bar, baz=rag; - идентификатор содержимого*; - описание содержимого*; - кодирование тела сообщения (7BIT, 8BIT, BINARY, BASE64, QUOTED-PRINTABLE)*; - размер тела сообщения в октетах; - информация, зависящая от типа сообщения (для текста это число строк в теле сообщения). Сообщения могут иметь сколь угодно сложную структуру, включать в себя различные вложения, содержать альтернативные форматы текста (простой текст, гипертекст и документ в формате Word), из которых клиент должен

	выбрать наиболее предпочтительный. Каждая часть может сама состоять из нескольких подчастей. Протокол IMAP позволяет работать по отдельности с разными частями сообщения и их подчастями . * См. описание спецификации MIME в RFC 2045 [4]
BODYSTRUCTURE	Содержит ту же информацию, что и BODY, а также некоторые дополнительные данные
BODY[секция]<x.y>, BODY.PEEK[секция]<x.y>	Текст соответствующей части сообщения: - секция – необязательный набор спецификаторов, разделенных точкой, если это поле оставлено пустым: [], берется сообщение целиком, в качестве спецификаторов выступают номера части и подчастей или одно из следующих ключевых слов: HEADER, HEADER.FIELDS, HEADER.FIELDS.NOT, MIME, TEXT. Каждое сообщение содержит как минимум одну часть. Структуру сообщений, включая количество и вложенность частей сообщения, можно выяснить, опросив элементы BODY или BODYSTRUCTURE этих сообщений; - x.y, где x – номер первого октета, запрошенного клиентом, y – количество запрошенных октетов. Используется в том случае, если надо получить не всю запрошенную часть сообщения, а только некоторый фрагмент. Если x больше, чем количество октетов в запрошенной части, клиент получает пустую строку. Объем информации, посылаемой клиенту не должен превышать y октетов. *Примеры* BODY[HEADER.FIELDS (DATE FROM)] – поля заголовка DATE и FROM; BODY[]<0.2048> – первые 2048 октетов сообщения; BODY[TEXT] – текст сообщения; BODY[HEADER] – заголовок сообщения. После выполнения, для опрошенного сообщения выставляется флаг \Seen, если используется BODY, и не выставляется, если используется BODY.PEEK
ENVELOPE	Содержимое полей Date, Subject, From, Sender, reply-to, to, cc, bcc, in-reply-to и message-id заголовка сообщения. Адреса в ответе сервера переписываются в следующем формате: имя пользователя, маршрут сообщения (обычно пустой), регистрационное имя в почтовой системе (левая часть адреса), почтовый домен (правая часть адреса)
FLAGS	Флаги, выставленные для сообщения
INTERNALDATE	Дата сообщения
RFC822	Эквивалентно BODY[]
RFC822.HEADER	Эквивалентно BODY.PEEK[HEADER]
RFC822.SIZE	Размер сообщения
RFC822.TEXT	BODY[TEXT]
UID	Уникальный идентификатор сообщения

Расширение BINARY (RFC 3516 [47]), дополнительно предусматривает еще три типа информационных элементов:

BINARYсекция[x.y] – аналогично BODY[секция]<x.y>, но данные передаются без

всякого кодирования, как двоичный поток;

BINARY.PEEKсекция[x.y] – тоже самое, но без установки флага \Seen;

BINARY.SIZEсекция – размер указанной секции в незакодированном виде.

Кодирование в соответствии со спецификацией MIME увеличивает размер сообщений. Отказ от кодирования двоичных файлов позволяет уменьшить объем передаваемых данных и, соответственно, сократить время передачи.

В ответе перед каждым информационным элементом записывается его название, если он состоит из нескольких составных частей, каждая из них заключается в скобки, если отдельные части тоже состоят или могут состоять из нескольких частей, то и их тоже заключают в скобки. Строковые константы заключаются в кавычки. Если обязательная составная часть информационного элемента оказывается пустой, то на ее месте записывается слово NIL.

Кроме типов информационных элементов установлено три макроса, описывающие наиболее часто используемые комбинации информационных элементов:

ALL – соответствует (FLAGS INTERNALDATE RFC822.SIZE ENVELOPE)

FAST – соответствует (FLAGS INTERNALDATE RFC822.SIZE)

FULL – соответствует (FLAGS INTERNALDATE RFC822.SIZE ENVELOPE BODY)

STORE x:y имя_элемента_данных (список_флагов)

Команда STORE изменяет значения флагов для указанного в первом аргументе сообщения или сообщений. В ответ сервер возвращает непомеченный ответ с ключевым словом FETCH, содержащий значения флагов, если в имени элемента данных не используется суффикс .SILENT.

Определены следующие имена элементов данных:

FLAGS – изменяет значения указанных флагов, кроме флага \Recent;

FLAGS.SILENT – аналогично FLAGS, значения флагов не возвращаются;

+FLAGS – устанавливает значения указанных флагов;

+FLAGS.SILENT – аналогично +FLAGS, значения флагов не возвращаются;

–FLAGS – сбрасывает значения указанных флагов;

–FLAGS.SILENT – аналогично –FLAGS, значения флагов не возвращаются.

C	A003 STORE 2:4 +FLAGS (\Deleted)
S	* 2 FETCH (FLAGS (\Deleted \Seen))
S	* 3 FETCH (FLAGS (\Deleted))
S	* 4 FETCH (FLAGS (\Deleted \Flagged \Seen))
S	A003 OK STORE completed

COPY x:y имя_ящика

Копирует сообщения с номерами от x до y из текущего почтового ящика в указанный почтовый ящик.

UID команда аргументы

Команда UID принимает в качестве аргументов команды COPY, FETCH или STORE с их аргументами, но наряду с номерами сообщений в ответах указываются уникальные идентификаторы сообщений.

Также команду UID можно использовать совместно с командой SEARCH. В этом случае интерпретация аргументов команды SEARCH не изменяется, но в ответах на эту команду будут приведены уникальные идентификаторы сообщений.

5.6. Пример сеанса IMAP

S	* OK IMAP4rev1 Service Ready
C	A001 login mrc secret
S	A001 OK LOGIN completed
C	A002 select inbox
S	* 18 EXISTS
S	* FLAGS (\Answered \Flagged \Deleted \Seen \Draft)
S	* 2 RECENT
S	* OK [UNSEEN 17] Message 17 is the first unseen message
S	* OK [UIDVALIDITY 3857529045] UIDs valid
S	A002 OK [READ-WRITE] SELECT completed
C	A003 fetch 12 full
S	* 12 FETCH (FLAGS (\Seen) INTERNALDATE "17-Jul-1996 02:44:25 -0700" RFC822.SIZE 4286 ENVELOPE ("Wed, 17 Jul 1996 02:23:25 -0700 (PDT)" "IMAP4rev1 WG mtg summary and minutes" (("Terry Gray" NIL "gray" "cac.washington.edu")) (("Terry Gray" NIL "gray" "cac.washington.edu")) (("Terry Gray" NIL "gray" "cac.washington.edu")) ((NIL NIL "imap" "cac.washington.edu")) ((NIL NIL "minutes" "CNRI.Reston.VA.US") ("John Klensin" NIL "KLENSIN" "MIT.EDU")) NIL NIL "<B27397-0100000@cac.washington.edu>") BODY ("TEXT" "PLAIN" ("CHARSET" "US-ASCII") NIL NIL "7BIT" 3028 92))
S	A003 OK FETCH completed
C	A004 fetch 12 body[header]
S	* 12 FETCH (BODY[HEADER] {342}
S	Date: Wed, 17 Jul 1996 02:23:25 -0700 (PDT)
S	From: Terry Gray <gray@cac.washington.edu>
S	Subject: IMAP4rev1 WG mtg summary and minutes
S	To: imap@cac.Washington.edu
S	Cc: minutes@CNRI.Reston.VA.US, John Klensin <KLENSIN@MIT.EDU>
S	Message-Id: <B27397-0100000@cac.washington.edu>
S	MIME-Version: 1.0
S	Content-Type: TEXT/PLAIN; CHARSET=US-ASCII
S:	
S	)
S	A004 OK FETCH completed
C	A005 store 12 +flags \deleted
S	* 12 FETCH (FLAGS (\Seen \Deleted))
S	A005 OK +FLAGS completed
C	A006 logout
S	* BYE IMAP4rev1 server terminating connection
S	A006 OK LOGOUT completed

Пользователь регистрируется на сервере под именем mrc с паролем secret, открывает почтовый ящик inbox, содержащий 18 сообщений, 2 из которых

получены после предыдущего сеанса. Номер первого непрочитанного сообщения – 17, уникальный идентификатор почтового ящика – 3857529045, также приведен список флагов, которые могут быть установлены для сообщений в выбранном почтовом ящике.

После этого клиент запрашивает сведения о двенадцатом сообщении в выбранном ящике. Ответ сервера содержит только одну непомеченную строку (нам пришлось разбить ее на несколько строк, чтобы она поместилась на странице).

Сервер отвечает, что это сообщение уже было просмотрено; оно получено 17.07.96 в 2:22:25 в часовом поясе –07:00; размер сообщения – 4286 октетов; в заголовке сообщения имеются следующие поля:

Date: Wed, 17 Jul 1996 02:23:25 -0700 (PDT),
Subject: IMAP4rev1 WG mtg summary and minutes,
From: Terry Gray <gray@cac.washington.edu>,
Sender: Terry Gray <gray@cac.washington.edu>,
reply-to: Terry Gray <gray@cac.washington.edu>,
to: imap@cac.washington.edu,
cc: minutes@CNRI.Reston.VA.US, John Klensin <KLENSIN@MIT.EDU>,
bcc: отсутствует,
in-reply-to: отсутствует,
message-id: <B27397-0100000@cac.washington.edu>.

Обратите внимание, адреса в ответе сервера на команду FETCH 12 FULL записаны в формате, отличающемся от принятого в адресах SMTP.

Тело сообщения имеет следующую структуру:

тип содержимого: TEXT;
подтип содержимого: PLAIN;
параметры тела сообщения: CHARSET=US-ASCII;
идентификатор тела сообщения: отсутствует;
описание тела сообщения: отсутствует;
кодирование тела сообщения: 7BIT;
размер тела сообщения в октетах: 3028;
число строк в теле сообщения: 92.

Клиент запрашивает заголовок сообщения 12. Обратите внимание, передаваемый элемент – заголовок, заключен в скобки, передается в виде литерала.

Клиент помечает сообщение 12 для удаления и завершает сеанс.

5.7. Расширения IMAP

Расширения IMAP аналогичны расширениям, используемым протоколами SMTP и POP3. Они добавляют возможности существующим командам, как, например, упоминавшиеся выше расширения BINARY, CHILDREN, LITERAL+ и MULTIAPPEND, и вводят новые команды. Клиент информируется о поддерживаемых сервером расширениях ответом на команду CAPABILITY.

Расширение ACL

Расширение ACL (RFC 2086 [48]) предназначено для управления доступом пользователей к почтовым ящикам и к информации в них.

Права доступа определяются набором атрибутов, возможно, пустым, устанавливаемым для пользователя в отношении почтового ящика. Каждый атрибут обозначается одной буквой.

Определены следующие атрибуты:

l – lookup, просмотр (для почтового ящика могут быть выполнены команды LIST/LSUB);

r – read, чтение (ящик может быть выбран, сообщения в нем доступны для чтения);

s – значение флага \Seen не может быть изменено для сообщений в ящике;

w – write, запись (без права изменения значений флагов \Seen и \Deleted;)

i – insert, добавление (команды APPEND и COPY);

p – post, отправка сообщения по адресу, установленному для почтового ящика;

c – create, создание ящика (относится к каталогу при иерархическом расположении почтовых ящиков);

d – delete, удаление сообщений;

a – administer, изменение прав доступа.

Как уже отмечалось выше, права доступа к ящику определяются для каждого пользователя. Так право изменения прав доступа может быть только у владельца ящика или у администратора, у других же пользователей в отношении этого ящика могут быть установлены другие права.

Расширение ACL добавляет к набору команд IMAP ряд дополнительных команд (табл. 5).

Таблица 5

Команда	Аргументы	Описание
SETACL	Ящик Пользователь Права_доступа	Изменение прав доступа указанного пользователя к указанному почтовому ящику. Символы, обозначающие права доступа записываются в одну строку, перед ними могут стоять символы плюс «+» (установка прав) или минус «-» (отмена прав). Если перед атрибутами нет символа, то уже установленные права доступа заменяются на указанные
DELETEACL	Ящик Пользователь	Удаляет все права доступа указанного пользователя к указанному почтовому ящику
GETACL	Ящик	Возвращает списки прав доступа к указанному почтовому ящику. Ответ содержит столько непомеченных строк с ключевым словом ACL, для скольких пользователей установлены права доступа к данному ящику. Каждая непомеченная строка ответа содержит имя пользователя и установленные для него права
LISTRIGHTS	Ящик Пользователь	Возвращает список прав доступа указанного пользователя к указанному почтовому ящику
MYRIGHTS	Ящик	Возвращает список прав доступа к указанному

		почтовому ящику пользователя, зарегистрированного в текущем сеансе

Пример

C	A001 LISTRIGHTS ~/Mail/saved smith
S	* LISTRIGHTS ~/Mail/saved smith larswicd
S	A001 OK Listrights completed

Расширение QUOTA

Серверы, поддерживающие расширение QUOTA (RFC 2087[49]), могут налагать ограничения на некоторые виды ресурсов. Названия и характер ресурсов зависят от технической реализации. Например, в качестве ресурсов могут выступать общий размер сообщений или их количество. Каждому почтовому ящику приписываются ноль или более корневых квот (quota roots), каждая из которых содержит ноль или более ограничений ресурсов. Ресурсы, ограниченные корневой квотой, распределяются между всеми почтовыми ящиками, которым она приписана. Расширение QUOTA вводит три команды (табл. 6).

Таблица 6

Команда	Аргументы	Описание
SETQUOTA	Корневая_квота Ограничения	Устанавливает указанные ограничения для указанной корневой квоты
GETQUOTA	Корневая_квота	Возвращает объем используемых ресурсов и размеры ограничений, установленных для указанной корневой квоты
GETQUOTAROOT	Имя_ящика	Возвращает имена корневых квот, установленных для указанного почтового ящика, объемы их используемых ресурсов и размеры ограничений

Пример

C	A003 GETQUOTA ""
S	* QUOTA "" (STORAGE 10 512)
S	A003 OK Getquota completed

Расширение IDLE

Расширение IDLE (RFC 2177 [50]) позволяет клиенту при выбранном состоянии сеанса IMAP получать информацию об изменениях в выбранном почтовом ящике, не запрашивая ее. Таким образом, клиент оперативно информируется о получении почты, а с сервера снимается нагрузка, связанная с обслуживанием регулярных переспросов клиента.

Клиент переходит в режим ожидания, направив серверу команду IDLE без параметров и получив подтверждение, начинающееся с символа «+». После этого сервер, не получая запроса клиента, может посылать ему информацию о выбранном почтовом ящике при каждом изменении его содержимого.

Чтобы выйти из режима ожидания, клиент посылает серверу строку, состоящую из одного слова «DONE» (это не команда, метка перед ней не ставится).

Клиент должен периодически подтверждать свою активность, прерывая

режим ожидания не реже, чем каждые 29 минут. Иначе сервер сам прекращает сеанс по истечении таймаута.

Пример

C	A001 SELECT INBOX
S	* FLAGS (Deleted Seen)
S	* 3 EXISTS
S	* 0 RECENT
S	* OK [UIDVALIDITY 1]
S	A001 OK SELECT completed
C	A002 IDLE
S	+ idling

Поступило новое сообщение

S	* 4 EXISTS
C	DONE
S	A002 OK IDLE terminated

Расширения MAILBOX-REFERRALS и LOGIN-REFERRALS

Ресурсы, доступ к которым предоставляется по протоколу IMAP, могут быть описаны при помощи унифицированных указателей ресурсов URL (RFC 2192 [51]).

В общем виде URL ресурса IMAP записывается таким образом:

imap://пользователь;AUTH=механизм_аутентификации@сервер/ресурс

где пользователь – регистрационное имя пользователя, если оно не задано, производится анонимная регистрация;

механизм_аутентификации – механизм аутентификации, используемый командой AUTHENTICATE. Если эта часть URL пропущена, то аутентификация проводится при помощи команды LOGIN. Если на месте механизма аутентификации стоит звездочка «*», используется один из механизмов, поддерживаемых клиентом и сервером;

сервер – имя сервера;

ресурс – описывается в зависимости от характера ресурса: список почтовых ящиков, почтовый ящик, список сообщений, сообщение или его часть.

В URL, указывающем на конкретное сообщение, ресурс описывается следующим образом:

ящик;UIDVALIDITY=ид_ящика/;UID=ид_сообщения/;section=секция

где ящик – имя почтового ящика, включающее путь к нему, если почтовые ящики на сервере расположены иерархически;

ид_ящика – необязательный идентификатор почтового ящика. Если он задан, доступ к почтовому ящику с указанным именем будет предоставлен, только если указанный идентификатор совпадает с идентификатором указанного ящика. Это позволяет избежать ошибки, которая может возникнуть, если ящик был удален, а после был создан ящик с тем же именем: в отличие от имени ящика, идентификатор не может совпадать с уже существовавшим;

ид_сообщения – идентификатор сообщения;

секция – необязательный аргумент, используемый в команде
UID FETCH ид_сообщения BODY.PEEK[секция].

Примеры URL и соответствующие последовательности команд, посылаемых клиентом

imap://minbari.org/gray-council;UIDVALIDITY=385759045/;UID=20

A001 LOGIN ANONYMOUS sheridan@babylon5.org

A002 SELECT gray-council

Клиент проверяет, совпадает ли известный ему идентификатор почтового ящика с тем идентификатором, который был получен от сервера

A003 UID FETCH 20 BODY.PEEK[]

imap://;AUTH=KERBEROS_V4@minbari.org/gray-council/;uid=20/;section=1.2

A001 AUTHENTICATE KERBEROS_V4

Обмен аутентификационными данными

A002 SELECT gray-council

A003 UID FETCH 20 BODY.PEEK[1.2]

imap://user;AUTH=*@ minbari.org /shared/foo

A001 AUTHENTICATE KERBEROS_V4

Обмен аутентификационными данными пользователя user. Пароль запрашивается

A002 SELECT "shared/foo"

Пробелы в аргументах и прочие символы, недопустимые в URL, кодируются в соответствии с требованиями RFC 1738 [52], где описываются общие правила оформления унифицированных указателей ресурсов.

URL ресурсов IMAP в частности используются расширением MAILBOX-REFERRALS (RFC 2193 [53]) для работы с распределенными ресурсами, предоставляемыми по протоколу IMAP.

В некоторых случаях бывает целесообразно располагать почтовые ящики не на одном сервере, а на нескольких. Поскольку клиенту заранее не известно, на каком именно сервере находится интересующий его почтовый ящик, он всегда обращается к одному серверу, который, если запрошенный ящик находится на другом сервере, возвращает негативный помеченный ответ со ссылкой на истинное расположение запрошенного ящика. После этого клиент устанавливает соединение с сервером, указанным в полученном URL, и вновь запрашивает почтовый ящик.

Пример

C	A001 SELECT REMOTE
S	A001 NO [REFERRAL imap://user;AUTH=*@server2/remote] Remote mailbox.

Клиенту не удалось открыть почтовый ящик, так как он находится на другом сервере – server2. Получив URL требуемого ресурса, клиент устанавливает соединение с server2.

S	* OK IMAP4rev1 server ready

C	B001 AUTHENTICATE KERBEROS_V4

Далее следует обмен аутентификационной информацией

S	B001 OK user is authenticated
C	B002 SELECT remote
S	* 12 EXISTS
S	* 1 RECENT
S	* OK [UNSEEN 10] Message 10 is first unseen
S	* OK [UIDVALIDITY 123456789]
S	* FLAGS (\Answered \Flagged \Deleted \Seen \Draft)
S	* OK [PERMANENTFLAGS (\Answered \Deleted \Seen)]
S	B002 OK [READ-WRITE] Selected completed

Аналогичные возможности обеспечивает также расширение LOGIN-REFERRALS (RFC 2221 [54]). Это расширение предназначено для перенаправления аутентифицировавшегося пользователя на сервер, содержащий его почтовые ящики.

Если после аутентификации клиент получает отрицательный ответ, содержащий URL другого сервера, клиенту следует аутентифицироваться на указанном сервере, так как там, где он только что аутентифицировался, нет доступных ему почтовых ящиков.

Пример

C	A001 LOGIN MIKE PASSWORD
S	A001 NO [REFERRAL imap://mike@server2/] try server2.

Положительный ответ, содержащий URL другого сервера, полученный клиентом после аутентификации, свидетельствует о том, что на основном сервере есть почтовые ящики, доступные данному пользователю, но его персональный почтовый ящик находится на другом сервере.

Пример

C	A001 LOGIN MATTHEW PASSWORD
S	A001 OK [REFERRAL imap://matthew@server2/] public mailboxes are available

Расширение NAMESPACE

Расширение NAMESPACE (RFC 2342 [58]) дает клиенту возможность с помощью не требующей параметров команды NAMESPACE определить, какие пространства имен задействованы на данном сервере.

Ответ сервера содержит префиксы и иерархические разделители пространств имен

- персональных почтовых ящиков;
- почтовых ящиков других пользователей, если таковые доступны данному пользователю;
- общедоступных почтовых ящиков.

Если какие-то из перечисленных пространств имен не существуют или недоступны данному пользователю, то на их месте в ответе должно стоять слово NIL.

Примеры

Пользователю доступны только его личные почтовые ящики.

C	A001 NAMESPACE
S	* NAMESPACE (("" "/")) NIL NIL
S	A001 OK NAMESPACE command completed

Анонимному пользователю доступны только ящики, содержащие новости Usenet.

C	A001 NAMESPACE
S	* NAMESPACE NIL NIL (("" "."))
S	A001 OK NAMESPACE command completed

Доступны личные почтовые ящики, почтовые ящики других пользователей, обратиться к которым можно, указав префикс "~", а также общедоступные ящики с префиксом "#public/" и новости Usenet с префиксом "#news.".

C	A001 NAMESPACE
S	* NAMESPACE (("" "/")) (("~" "/")) (("#public/" "/")("#news." "."))
S	A001 OK NAMESPACE command completed

Расширение UIDPLUS

Расширение UIDPLUS (RFC 2359 [59]) позволяет использовать команду UID совместно с командой EXPUNGE, а также добавляет информацию об уникальных идентификаторах в ответы на команды APPEND, COPY и UID COPY. Тех же результатов можно добиться, используя команды базового протокола, но данное расширение позволяет получить тот же результат несколько быстрее.

Расширение ID

Расширение ID (RFC 2971 [60]) дает клиенту и серверу возможность обменяться сведениями об установленном на них программном обеспечении.

Команда принимает в качестве аргумента заключенный в скобки набор пар параметр/значение. В общем виде формат команды следующий:

ярлык ID ("параметр1" "значение1" "параметр2" "значение2" …)

Непомеченный ответ сервера имеет аналогичный формат. Как в команде, так и в ответе вместо сведений может стоять слово NIL, если клиент или сервер не желают сообщать информацию о себе.

В качестве имен параметров можно использовать любые строки длиной не более 30 символов.

Следующие имена параметров определены в RFC 2971 [60]:

- ✓ name – название программы;
- ✓ version – номер версии;
- ✓ os – операционная система;
- ✓ os-version – версия операционной системы;
- ✓ vendor – производитель;
- ✓ support-url – URL службы технической поддержки;

- ✓ address – контактный адрес производителя;
- ✓ command – команда, используемая для запуска программы;
- ✓ arguments – аргументы командной строки;
- ✓ environment – переменные окружения.

Пример

C	a023 ID ("name" "sodr" "version" "19.34")
S	* ID ("name" "Cyrus" "version" "1.5" "os" "sunos" "os-version" "5.5")
S	a023 OK ID completed

5.8. Контрольные вопросы

1. Какие возможности для работы с почтовыми ящиками и сообщениями предоставляет протокол IMAP?
2. В чем заключаются преимущества и недостатки протокола IMAP по сравнению с протоколом POP3? На основании каких критериев следует выбирать один из этих протоколов?
3. На какие состояния делится сеанс IMAP? Как происходят переходы из одного состояния в другое? Какие действия характерны для каждого состояния сеанса?
4. Почему на сервере IMAP используются разные пространства имен? Как они различаются?
5. Что такое ярлыки команд? Для чего они используются? Чем отличаются помеченные и непомеченные ответы сервера?
6. Что такое литералы? Как и в каких случаях они используются?
7. Какие способы аутентификации используются протоколом IMAP? Отличаются ли они от используемых протоколами SMTP и POP3?
8. Напишите пример последовательности команд IMAP, которые надо выполнить, чтобы получить текст сообщения, посланного в заданный день заданным отправителем.
9. Как организуется взаимодействие клиента с распределенной системой серверов IMAP, в которой ящики, доступные клиенту, располагаются на разных серверах?

Список сокращений

ACL (Access Control List) – список контроля доступа
CGI (Common Gateway Interface) – общий шлюзовой интерфейс. Сетевой стандарт, предназначенный для создания серверных приложений HTTP
CRLF (Carriage Return, Line Feed) – последовательность символов «возврат каретки», «перевод строки» (шестнадцатиричный код – 0D0A), обозначающая конец строки
DNS (Domain Name System) – система доменных имен
DSN (Delivery Status Notifications) – оповещения о состоянии доставки
ESMTP (Extended SMTP) – расширенный SMTP

HTTP (HyperText Transfer Protocol) – протокол передачи гипертекста
HTTPS (HTTP Secure) – защищенный протокол передачи гипертекста
IETF (Internet Engineering Task Force) – проблемная группа проектирования Интернет
IMAP (Internet Message Access Protocol) – протокол доступа к сообщениям Интернет
IP (Internet Protocol) – межсетевой протокол
IPC (InterProcess Communication) – межпроцессное взаимодействие
LDA (Local Delivery Agent) – агент локальной доставки
LMTP (Local Mail Transfer Protocol) – протокол локальной передачи почты
MIME (Multipurpose Internet Mail Extensions) – многоцелевые расширения электронной почты в Интернет
MSA (Message Submission Agent) – агент подачи сообщения
MTA (Message Transfer Agent) – агент пересылки сообщений
MUA (Mail User Agent) – почтовый агент пользователя
MUPDATE (Mailbox Update) – обновление почтового ящика
MX (Mail eXchange) – обмен почтой, запись в DNS, указывающая на адрес машины, обрабатывающей почту для данного домена
POP3 (Post Office Protocol - Version 3) – протокол почтового отделения, версия 3
RFC (Request for Comments) – запрос на комментарии, официальный документ, описывающий стандарты Интернет и связанные с ними вопросы
SMTP (Simple Mail Transfer Protocol) – простой протокол передачи почты
SASL (Simple Authentication and Security Layer) – простой уровень аутентификации и безопасности
TCP (Transmission Control Protocol) – протокол управления передачей
TCP/IP (Transmission Control Protocol / Internet Protocol) – стек протоколов Интернет
TLS (Transport Layer Security) – безопасность транспортного уровня
UID (Unique IDentifier) – уникальный идентификатор
URL (Uniform Resource Locator) – унифицированный указатель ресурса
UUCP (UNIX to UNIX Copy Protocol) – протокол обмена файлами между системами под управлением операционной системы UNIX
WWW (World Wide Web) – всемирная паутина

ЛИТЕРАТУРА

1. Klensin J., Ed. Simple Mail Transfer Protocol. RFC 2821, April 2001.

2. Resnick P., Ed. Internet Message Format. RFC 2822, April 2001.

3. Moore K. MIME (Multipurpose Internet Mail Extensions) Part Three: Message Header Extensions for Non-ASCII Text. RFC 2047, November 1996.

4. Freed N., Borenstein N. Multipurpose Internet Mail Extensions (MIME) Part One: Format of Internet Message Bodies. RFC 2045, November 1996

5. Mockapetris P. Domain names - concepts and facilities. RFC 1034, November 1987.

6. Myers J., Rose M. Post Office Protocol - Version 3. RFC 1939, May 1996.

7. Crispin M. INTERNET MESSAGE ACCESS PROTOCOL - VERSION 4rev1. RFC 3501, March 2003.

8. Lindberg G. Anti-Spam Recommendations for SMTP MTAs. RFC 2505, February 1999.

9. Braden R., Ed. Requirements for Internet Hosts - Application and Support. RFC 1123, October 1989.

10. Gellens R., Klensin J. Message Submission. RFC 2476, December 1998.

11. Myers J. Local Mail Transfer Protocol. RFC 2033, October 1996.

12. Siemborski R. The Mailbox Update (MUPDATE) Distributed Mailbox Database Protocol. RFC 3656, December 2003.

13. Postel J. Simple Mail Transfer Protocol. RFC 821, August 1982.

14. Ullmann R. SMTP on X.25. RFC 1090, Feb-01-1989.

15. Partridge C. Duplicate messages and SMTP. RFC 1047, Feb-01-1988.

16. Klensin J., Freed N., Rose M., Stefferud E., Crocker D. SMTP Service Extension for 8bit-MIMEtransport. RFC 1652, July 1994.

17. Crocker D., Freed N., Cargille A. SMTP Service Extension for Checkpoint/Restart. RFC 1845, September 1995.

18. Klensin J., Freed N., Moore K. SMTP Service Extension for Message Size Declaration. RFC 1870, November 1995.

19. De Winter J. SMTP Service Extension for Remote Message Queue Starting. RFC 1985, August 1996.

20. Vaudreuil G. Enhanced Mail System Status Codes. RFC 3463, January 2003.

21. Freed N. SMTP Service Extension for Returning Enhanced Error Codes. RFC 2034, October 1996.

22. Myers J. SMTP Service Extension for Authentication. RFC 2554, March 1999.

23. Myers J. Simple Authentication and Security Layer (SASL). RFC 2222, October 1997.

24. Josefsson S., Ed. The Base16, Base32, and Base64 Data Encodings. RFC 3548,

July 2003.

25. Hoffman P. SMTP Service Extension for Secure SMTP over Transport Layer Security. RFC 3207, February 2002.

26. Dierks T., Allen C. The TLS Protocol Version 1.0. RFC 2246, January 1999.

27. Gellens R. ON-DEMAND MAIL RELAY (ODMR) SMTP with Dynamic IP Addresses. RFC 2645, August 1999.

28. Moore K. Simple Mail Transfer Protocol (SMTP) Service Extension for Delivery Status Notifications (DSNs). RFC 3461, January 2003.

29. Newman D. Deliver By SMTP Service Extension. RFC 2852, June 2000.

30. Freed N. SMTP Service Extension for Command Pipelining. RFC 2920, September 2000.

31. Vaudreuil G. SMTP Service Extensions for Transmission of Large and Binary MIME Messages. RFC 3030, December 2000.

32. Nelson R. Some Observations on Implementations of the Post Office Protocol (POP3). RFC 1957, June 1996.

33. Gellens R., Newman C., Lundblade L. POP3 Extension Mechanism. RFC 2449, November 1998.

34. Crocker, D. Standard for the Format of ARPA-Internet Text Messages. RFC 822, August 1982.

35. Rivest R. The MD5 Message-Digest Algorithm. RFC 1321, April 1992.

36. Myers J. POP3 AUTHentication command. RFC 1734, December 1994.

37. Linn, J. Generic Security Service Application Program Interface. RFC 1508, September 1993.

38. Newman C. Using TLS with IMAP, POP3 and ACAP. RFC 2595, June 1999.

39. Gahrns M. IMAP4 Multi-Accessed Mailbox Practice. RFC 2180, July 1997.

40. Crispin M. Distributed Electronic Mail Models in IMAP4. RFC 1733, December 1994.

41. Leiba B. IMAP4 Implementation Recommendations. RFC 2683, September 1999.

42. Myers J. IMAP4 non-synchronizing literals. RFC 2088, January 1997.

43. Myers J. IMAP4 Authentication Mechanisms. RFC 1731, December 1994.

44. Newman C. Anonymous SASL Mechanism. RFC 2245, November 1997.

45. Gahrns M., Cheng R. The Internet Message Action Protocol (IMAP4) Child Mailbox Extension. RFC 3348, July 2002.

46. Crispin M. Internet Message Access Protocol (IMAP) - MULTIAPPEND Extension. RFC 3502, March 2003.

47. Nerenberg L. IMAP4 Binary Content Extension. RFC 3516, April 2003

48. Myers J., Rose M. The Content-MD5 Header Field. RFC 1864, October 1995.

49. Troost R., Dorner S., Moore K. Communicating Presentation Information in Internet Messages: The Content-Disposition Header. RFC 2183, August 1997.

50. Alvestrand H. Tags for the Identification of Languages. RFC 3066, January 2001.

51. Myers J. IMAP4 ACL extension. RFC 2086, January 1997.

52. Myers J. IMAP4 QUOTA extension. RFC 2087, January 1997.
53. Leiba B. IMAP4 IDLE command. RFC 2177, June 1997
54. Newman C. IMAP URL Scheme. RFC 2192, September 1997.
55. Berners-Lee T., Masinter L., McCahill M. Uniform Resource Locators (URL). RFC 1738, December 1994.
56. Gahrns M. IMAP4 Mailbox Referrals. RFC 2193, September 1997.
57. Gahrns M. IMAP4 Login Referrals. RFC 2221, October 1997.
58. Gahrns M., Newman C. IMAP4 Namespace. RFC 2342, May 1998.
59. Myers J. IMAP4 UIDPLUS extension. RFC 2359, June 1998.
60. Showalter T. IMAP4 ID extension. RFC 2971, October 2000.