РУКОВОДЯЩИЙ ДОКУМЕНТ ОТРАСЛИ

Средства технические телематических служб

Общие технические требования

Минсвязи России

Москва

gost.e-ecolog.ru

Предисловие

1 РАЗРАБОТАН Ассоциацией Документальной Электросвязи (АДЭ), испытательным центром документальной электросвязи (ИЦ ДЭС), испытательным центром "ЦНИИС".
ВНЕСЕН Департаментом электросвязи Министерства РФ по связи и информатизации

2 УТВЕРЖДЕН Министерством Российской Федерации по связи и информатизации

3 ВВЕДЕН В ДЕЙСТВИЕ информационным письмом

от 14.07 2000 г.№ 4-316

4 ВВЕДЕН ВПЕРВЫЕ

Настоящий руководящий документ отрасли не может быть полностью или частично воспроизведен, тиражирован и распространен в качестве официального издания без разрешения Министерства Российской Федерации по связи и информатизации.

РД 45.134-2000

РУКОВОДЯЩИЙ ДОКУМЕНТ ОТРАСЛИ

СОГЛАСОВАНО

Руководитель Департамента электросвязи

В Ю Квицинский

"23" 06 2000 г

УТВЕРЖДАЮ

Первый заместитель министра Российской Федерации по связи и информатизации

Ю А Павленко

"26" 06 2000 г

Лист утверждения

Средства технические телематических служб

Общие технические требования

Председатель исполкома
Ассоциации Документальной Электросвязи — А С Кремер

Начальник группы стандартизации — Ю В Капустин

Директор НТЦ ЦНИИС — В Б Садовский

Руководитель группы испытаний
Испытательного Центра Документальной
Электросвязи (ИЦ ДЭС) — Ю В Загубин

СОДЕРЖАНИЕ

1. Область применения 5
2. Нормативные ссылки.....6
3. Обозначения и сокращения.....8
4. Технические требования.....16
5. Требования к техническому обеспечению.....17
6. Требования к надежности и достоверности.....18
7. Требования к диагностике.....19
8. Требования к электропитанию.....20
9. Требования к электробезопасности и электромагнитной совместимости.....21
10. Требования по устойчивости к климатическим факторам.....22
11. Требования к документации.....23
12. Приложение 1.....24
13. Приложение 2.....40
14. Приложение 3.....53
15. Приложение 4.....89
16. Приложение 5.....115
17. Приложение 6.....153
18. Приложение 7.....161

1. ОБЛАСТЬ ПРИМЕНЕНИЯ

Настоящие общие технические требования предназначен для руководства при проведении сертификационных испытаний и инспекционного контроля в системе "Электросвязь" технических средств телематических служб.

Не все функции. содержащиеся в РД, обязательны для реализации в технических средствах телематических служб, но если эти функции выполняются, то их реализация должна соответствовать требованиям РД.

Настоящие общие технические требования распространяются на технические средства следующих телематических служб:

службы обмена электронными сообщениями в части службы электронной почты (по протоколам SMTP, POP3, IMAP4);

информационные службы в части службы доменных имен (по протоколу DNS), службы доступа к информационным ресурсам (по протоколам HTTP, NNTP, FTP).

Технические требования к техническим средствам факсимильной службы в части службы телефакс определены в РД 45.121-2000.

Технические требования к техническим средствам службы голосовой связи в части службы голосовых сообщений определены в "Общих технических требованиях на оборудование электронных речевых серверов", утвержденных Госкомсвязи России 24.06.98г.

Технические требования к техническим средствам службы голосовой связи в части службы передачи речевых сообщений определены в РД 45.46-99.

2. НОРМАТИВНЫЕ ССЫЛКИ

[1] Рекомендация IETF RFC 821	Простой протокол передачи почты. 1982г. (Simple Mail Transfer Protocol)
[2] Рекомендация IETF RFC 822	Стандарт для формата текстовых сообщений Интернета ARPA., 1982г. (Standart for the Format of ARPA Internet Text Messages.)
[3] Рекомендация IETF RFC 1939	Протокол почтового офиса, 1996г. (Post Office Protocol – Version 3)
[4] Рекомендация IETF RFC 1734	Команда AUTH протокола POP3, 1994г. (POP3. AUTHentication command)
[5] Рекомендация IETF RFC 1321	Алгоритм цифрового сообщения MD5, 1992г. (The MD5 Message-Digest Algorithm)
[6] Рекомендация IETF RFC 1731	Механизм аудентификации протокола IMAP4, 1994г. (IMAP4. Authentication Mechanisms)
[7] Рекомендация IETF RFC 1733	Распределенные модели электронной почты в IMAP4, 1994г. (Distributed Electronic Mail Models in IMAP4)
[8] Рекомендация IETF RFC 2045	MIME (Многоцелевые расширения почты Интернета). Часть 1: Формат тела сообщения Internet.,1996г. (MIME (Multipurpose Internet Mail Extensions). Part One: Format of Internet Message Bodies)
[9] Рекомендация IETF RFC 1700	Выделение номера., 1994г. (Assigned Number)
[10] Рекомендация IETF RFC 1730	Протокол доступа к сообщениям Интернет, 1996г. (Internet Message Access Protocol – version 4)
[11] Рекомендация IETF RFC 2060	Протокол доступа к сообщениям Интернет, 1996г. (Internet Message Access Protocol – version 4rev1)
[12] Рекомендация IETF RFC 1034	Доменные имена.Концепции и возможности., 1987г. (Domain name – concepts and facilities)
[13] Рекомендация IETF RFC 1035	Доменные имена. Реализация и спецификация., 1987г. (Domain name – implementation and specification)
[14] Рекомендация IETF RFC 2068	Протокол передачи гипертекста – HTTP/1.1, 1997г. (Hypertext Transfer Protocol – HTTP/1.1)

[15] Рекомендация IETF RFC 2048	Процедура регистрации типа информации.1996г. (Media Type Registration Procedure)
[16] ISO-8859	Международный стандарт по обработке информации – 8-ми битные однобайтовые наборыкодов графических символов. (International Standart - Information Processing - 8-bit Single Byte Coded Graphic Character Sets.)
[17] Рекомендация IETF RFC 2047	MIME (Многоцелевые расширения почты Интернета). Часть 3: Расширения заголовка для не ASCII-текста.,1996г. (MIME (Multipurpose Internet Mail Extensions) Part Three: Message Header Extensions for Non-ASCII Text)
[18] Рекомендация IETF RFC 1123	Требования для узлов Интернета – применение и поддержка., 1989г. (Requirements for Internet hosts – application and support.)
[19] Рекомендация IETF RFC 1036	Стандарт для обмена USENET сообщениями., 1987г. (Standart for interchange of USENET messages)
[20] Рекомендация IETF RFC 850	Стандарт для обмена USENET сообщениями., 1983г. (Standart for interchange of USENET messages)
[21] Рекомендация IETF RFC 1952	Спецификация формата файла GZIP версии 4.3., 1996г. (GZIP file format specification version 4.3)
[22] Рекомендация IETF RFC 1766	Тэги для идентификации языков., 1995г. (Tags for the identification of languages)
[23] ISO-639	Коды для представления названий языков.,1988г. (Code for representation of names languages)
[24] ISO-3166	Коды стран. (Country codes)
[25] Рекомендация IETF RFC 1864	Поле заголовка Content-MD5.,1995 (The Content-MD5 Header Field)
[26] Рекомендация IETF RFC 977	Протокол передачи сетевых новостей, 1986г. (Network News Transfer Protokol)
[27] Рекомендация IETF RFC 959	Протокол передачи файлов, 1985г. (File Transfer Protocol)
[28] Рекомендация IETF RFC 943	Выделенные числа, 1985г. (Assigned Numbers)
[29] Рекомендация IETF RFC 2138	Служба удаленной аутентификации пользователей подключаемых через телефонную сеть общего пользования (ТфОП) (Remote Authentication Dial In User Service)

3. ОБОЗНАЧЕНИЯ И СОКРАЩЕНИЯ

ASCII - набор символов. определенный в ARPA-Internet Protocol Handbook. В FTP определена нижняя часть восьмибитного кода (старший бит равен нулю)

ccTLD - национальный домен верхнего уровня (Country Code Top Level Domain)

CR - символ возврата каретки

CRLF - символы перехода в начало следующей строки, соответствующие <CR> и <LF>

DAP - протокол доступа к справочнику (Directory Access Protocol)

DNS (Domain Name System) - система доменных имен

DTP (data transfer process) - процесс передачи данных. Устанавливает и управляет соединением данных. Может быть активным и пассивным.

EOF - символ конца файла, определяющий конец передаваемого файла.

EOL - последовательность символов <CR> и <LF>, разделяющая линии, выводимые на печать.

EOR - символ конца записи, определяющий конец передаваемой записи.

FTP - протокол передачи файлов (File Transport Protocol)

HTTP - протокол передачи гипертекста (Hyper Text Transfer Protocol)

IETF - Рабочая группа по инженерным проблемам сети Интернет (Internet Engineering Task Force)

IP - межсетевой протокол (Internet Protocol)

LF - символ перехода на следующую строку

MIME-IMB - формат сообщения, описанный в рекомендации RFC 2045 [8]

Mode (режим передачи данных) - определяет формат данных при передаче, включая EOR и EOF.

NNTP – Network News Transfer Protocol (протокол передачи сетевых новостей)

NUL – специальный атом, представляющий отсутствие отдельного элемента данных, представленного как строка, либо список в скобках, в отличие от пустой строки "" или пустого списка в скобках ().

NULL - строка – строка, состоящая только из символов <CRLF>

NVFS - сетевая виртуальная файловая система. Концепция, определяющая стандартную сетевую файловую систему со стандартными командами и преобразованием имен путей.

NVT - виртуальный терминал сети, согласно определению, данному в описании протокола Telnet.

Page (страница) - структурная единица файла.

Pathname - символьная строка, идентифицирующая файл в файловой системе Конкретный вид зависит от файловой системы.

PI - интерпретатор протокола. Стороны клиента и сервера реализуют PI клиента и PI сервера.

POP3 - протокол обмена почтовой информацией (Post Office Protocol)

RADIUS - протокол аутентификации пользователей в соответствии с RFC 2138 [29] [(Remote Authentication Dial In User Service)

Record (запись) - структурная единица последовательного файла. Структура записей поддерживается FTP, но файл не должен состоять из структуры записей.

RFC - обозначение документа IETF (Request For Comments)

RR (Resource Records) - набор информации о ресурсе, связанный с отдельным доменным именем

SMTP - простой протокол передачи почты (Simple Mail Transport Protocol)

SNMP - протокол управления сетью на базе TCP/IP (Simple Network Management Protocol)

SP - символ пробела

TCP - транспортный протокол (Transport Control Protocol)

TCP/IP - стек протоколов межсетевого взаимодействия (Transmission Control Protocol/Internet Protocol)

Type (тип представления данных). Тип определяет преобразования при хранении данных и передаче данных.

UDP (User Datagramm Protocol) - протокол пользовательских датаграмм

UID - уникальный идентификатор почтового сообщения

URI - Universal Resource Identifier (универсальный идентификатор ресурса)

URL - Uniform Resource Locators (унифицированный указатель ресурсов)

АВ-терминал - аудио-видео терминал

Авторитетные данные - данные, полученные от авторитетного сервера

Авторитетный сервер - сервер, хранящий полную информацию о зоне

Агент клиентский - клиент, инициирующий запрос (броузер, редактор, робот или другое средство)

Адресат - клиент, которому предназначается почтовое сообщение

АП - агент пользователя

АПС - агент передачи сообщений

Атом - структура данных, состоящая из одного или более символов, не являющихся специальными.

Валидатор - элемент протокола, используемый для определения, является ли данная позиция кэша эквивалентной копией сущности.

Вариант - каждое из отдельных представлений ресурса, связанное с представлением в данный момент.

Возраст - время, прошедшее с момента отправки или успешной проверки актуальности ответа.

Группа новостей - имя, идентифицирующее группу клиентов, которым будет доставлена данная статья

Группа распространения - имя, идентифицирующее группу клиентов, которым будет доставлена данная статья в дополнение к клиентам группы новостей

Данные электронной почты - последовательность произвольной длины, состоящая из символов кода ASCII. удовлетворяющая формату почтового сообщения в соответствии с RFC 822[2].

Домен - иерархически структурированный глобальный адрес компьютера узла сети в виде строки символов

Заголовок сообщения HTTP - 1. набор строк между первой строкой сообщения (start-line) и пустой строкой, отделяющей заголовок от тела сообщения. 2. - строка (несколько строк), содержащая выражение, задающее значение для отдельного поля заголовка сообщения.
Идентификатор валидности - уникальное значение, выделяемое для каждого почтового ящика. При удалении почтового ящика и создании почтового ящика с таким же именем, идентификатор валидности нового почтового ящика должен быть отличным от предыдущего.

Идентификатор уникальный почтового сообщения (UID сообщения) - 32-х битовый номер, выделяемый для каждого почтового сообщения и используемый совместно с уникальным идентификатором валидности. UID и идентификатор валидности вместе занимают 64 бита. Значение итогового составного идентификатора является гарантированно уникальным для каждого почтового сообщения в данном почтовом ящике.

ИС - информационная служба

ИСО - Международная организация по стандартизации (International Organization for Standartization)

Канал передачи - полнодуплексное соединение между передатчиком SMTP и приемником SMTP, используемое для обмена командами, ответами и текстом почтовых сообщений.

"Клеевые" записи - записи, содержащие ссылку на авторитетый сервер подзоны

Клиент - программа, устанавливающая соединение с целью получения услуги некоторого вида, определенного соответствующим протоколом. Причиной, источником запуска такой программы, может выступать процесс на компьютере или действия человека.

Команда - сообщение NNTP, направляемое от клиента к серверу

Конверт сообщения – заголовок почтового сообщения.

Кэш - местное хранилище сообщений ответов, а также подсистема, управляющая хранением и удалением сообщений. Как сервер, так и клиент могут содержать кэш, хотя кэш не может использоваться на сервере, выполняющим функции тоннеля.

"Лист" - элемент иерархического графа, дерева, не имеющего выходящих дуг

Литерал – основная форма строки. Представляет последовательность из 0 или более октетов (включая символы <CR> и <LF>), которой предшествует счетчик октетов. Формат счетчика октетов: открывающаяся фигурная скобка "{", число октетов, закрывающаяся фигурная скобка "}", <CRLF>. В случае, когда литерал посылается от клиента серверу, клиент должен ждать получения запроса продолжения команды перед отправлением данных (и остатка команды). Даже если счетчик октетов равен 0, клиент, передающий буквенную строку, должен ждать получения команды запроса продолжения.

МПС - служба межперсональных сообщений

МСЭ - Международный союз электросвязи

МСЭ-Т - Сектор стандартизации электросвязи МСЭ

Новости электронные (news) - вид информации, периодически распространяемой в виде электронных сообщений большому количеству клиентов по сети передачи данных.

Номер порядковый почтового сообщения - относительный номер сообщения в почтовом ящике. Сообщения в почтовом ящике должны располагаться по возрастанию значения UID. Два соседних порядковых номера сообщения должны отличаться точно на 1. Порядковые номера сообщений могут изменяться в течение сессии.

Остаток (имени, команды) - последний элемент (группа элементов) структуры

Ответ - сообщение NNTP, направляемое от сервера клиенту

Ответ актуальный - ответ, который не устарел, не утратил актуальности

Ответ отрицательный - ответ со значением индикатора статуса "-ERR"

Ответ первичный - ответ, который пришел от сервера-источника. Ответ также является первичным, если его актуальность была проверена непосредственно сервером-источником.

Ответ положительный - ответ со значением индикатора статуса "+OK"

Ответ устаревший - ответ, у которого истек срок его актуальности

Отправитель - клиент, инициировавший отправку почтового сообщения

Передатчик SMTP - процесс, осуществляющий передачу электронной почты. Передатчик SMTP инициирует соединение транспортного уровня.

Представление - сущность, включенная в ответ, являющийся предметом согласования содержимого. Может быть множество различных представлений, ассоциированных с отдельным статусом ответа.

Приемник SMTP - процесс, осуществляющий прием электронной почты.

Прокси - программа-посредник, выполняющая функции сервера и клиента с целью выполнения запросов от имени других клиентов.

Процесс сервера FTP (сервер) - процесс, выполняющий функции передачи файлов совместно с процессом клиента FTP и, возможно, другим сервером. Функционально процесс сервера можно разделить на процесс интерпретатора протокола (PI) и процесс передачи данных (DTP).

"Разрез" - точка разделения иерархисеского графа (дерева) на два составляющих иерархических графа (поддерева)

РД - руководящий документ

Режим стойких соединений - режим работы сервера, при котором он обрабатывает несколько запросов клиента, не разрывая соединения TCP с данным клиентом.

Ресурс - объект сетевых данных или служба, которая может быть идентифицирована посредством URI.

САК - служба аудиоконференций

СВК - служба видеоконференций

СГС - служба голосовых сообщений

Сервер - **1.** - прикладная программа, принимающая соединения с целью обслуживания запросов путем отправки ответов. Использование этого термина относится только к текущему конкретному соединению, так как может быть программа, способная выполнять функции и клиента и сервера. **2.** - процесс, выполняющий функции доступа к электронной почте совместно с процессом клиента

Сервер-источник - сервер, на котором находится или создается данный ресурс.

серверов. В отличие от прокси, шлюз принимает запросы так, как если бы он был сервером-источником для запрошенного ресурса.

Сессия - набор процедур обмена, происходящих по открытому соединению транспортного уровня

Система новостей USENET - способ организации доставки электронных новостей, а также набор аппаратно-программного обеспечения, реализующего данный способ. При данном способе доставки электронные новости доставляются клиенту специальными средствами в моменты времени, определяемые стороной клиента.

СКА - сервер контроля и авторизации

Слово - последовательность печатных символов

Согласование содержимого - механизм для выбора соответствующего представления при обслуживании запроса.

Соединение данных - полнодуплексное соединение, по которому в определенном режиме (mode) передаются данные определенного типа (type).

Сообщение - информация, состоящая из структурированной последовательности октетов, удовлетворяющая синтаксису сообщения HTTP и передаваемая по соединению

Сообщение NNTP - сообщение, передаваемое по каналу передачи протокола нижнего уровня, используемого протоколом NNTP.

Список в скобках - структура данных, представляющая собой последовательность элементов данных, разделенных пробелами и заключенных в скобки. Список в скобках может, в свою очередь, содержать другие списки в скобках. При этом несколько уровней скобок показывают вложенность. Пустой список представляется как () - список в скобках, не содержащий членов.

Список рассылки (mailing list) - способ организации доставки электронных новостей, а также набор аппаратно-программного обеспечения, реализующего данный способ. При данном способе доставки электронные новости доставляются клиенту средствами электронной почты в моменты времени, определяемые серверной стороной списка рассылки.

СПРИ - служба передачи речевой информации

СПС - система передачи сообщений

Срок актуальности - промежуток времени между генерацией ответа и окончанием срока истечения

Срок истечения точный - время, по истечении которого сервер-источник считает, что сущность не должна больше выдаваться кэшем без дальнейшей проверки актуальности.

Срок истечения эвристический - срок истечения точный, устанавливаемый кэшем.

Статья - сообщение электронных новостей

СТК - служба телеконференций

Строка бинарная – это любая строка с символами NUL.

Строка в кавычках – форма строки, представляющая собой последовательность из 0 или более семибитных символов, кроме символов <CR> и <LF>, с символом двойной кавычки <"> с каждой стороны.

Сущность - информация, передаваемая в виде полезной нагрузки запроса или ответа. Сущность состоит из метаинформации в форме полей заголовка сущности и содержимого в форме тела сущности.

СХИ - система хранения информации

Тег – короткая строка, состоящая из буквенно-цифровой информации, используемая в качестве идентификатора команды.

ТМ службы - телематические службы

Тоннель - промежуточная программа, работающая как безусловный ретранслятор между двумя соединениями. Будучи установленным, активный тоннель не рассматривается как часть взаимодействия по HTTP, хотя тоннель может быть установлен вследствие запроса HTTP. Тоннель перестает существовать, когда оба соединения закрываются.

Транзакция - набор процедур обмена, требуемый для того, чтобы одно почтовое сообщение было передано одному или нескольким получателям

ТС - сеанс телеконференцсвязи

ТфОП - телефонная (сеть) общего пользования

Узел сети (узел) - компьютер, подключенный к сети, на котором запущен процесс SMTP, либо присутствуют почтовые ящики

Указатель конца данных почты - специальная последовательность символов, указывающая на конец данных электронной почты. Состоит из последовательности символов: CR, LF, символа точка ("."), CR, LF.

УПОР - устройство пакетной обработки речи

Управляющее соединение - соединение между PI клиента и PI сервера для обмена командами и ответами.

УТС DNS - узел телематических служб, реализующий функции сервера DNS.

УТС FTP - узел телематических служб, реализующий функции сервера FTP.

УФС - узел факсимильной связи

ХС - хранилище сообщений

Шлюз - сервер, который работает как промежуточный для некоторых других

ЭП - электронная почта

Ящик электронной почты (почтовый ящик) - набор символов, идентифицирующий клиента, которому отправляется почта. Обычно состоит из спецификации клиента и узла. Дополнительно, под данным термином понимают абстрактный "контейнер", в котором хранятся сообщения электронной почты.

4. ТЕХНИЧЕСКИЕ ТРЕБОВАНИЯ

Технические требования к техническим средствам службы обмена электронными сообщениями в части службы электронной почты по протоколу SMTP приведены в Приложении 1.

Технические требования к техническим средствам службы обмена электронными сообщениями в части службы электронной почты по протоколу POP3 приведены в Приложении 2.

Технические требования к техническим средствам службы обмена электронными сообщениями в части службы электронной почты по протоколу IMAP4 приведены в Приложении 3.

Технические требования к техническим средствам информационных служб в части службы доменных имен по протоколу DNS приведены в Приложении 4.

Технические требования к техническим средствам службы доступа к информационным ресурсам по протоколу HTTP приведены в Приложении 5.

Технические требования к техническим средствам службы доступа к информационным ресурсам по протоколу NNTP приведены в Приложении 6.

Технические требования к техническим средствам службы доступа к информационным ресурсам по протоколу FTP приведены в Приложении 7

5. ТРЕБОВАНИЯ К ТЕХНИЧЕСКОМУ ОБЕСПЕЧЕНИЮ.

Используемые при создании ТС телематической службы средства вычислительной техники должны иметь сертификаты системы ГОСТ Р, подтверждающие соответствие Российским стандартам на средства вычислительной техники, эксплуатируемые в производственных помещениях.

6. ТРЕБОВАНИЯ К НАДЕЖНОСТИ И ДОСТОВЕРНОСТИ.

ТС телематических служб должны быть рассчитаны на круглосуточную работу без постоянного присутствия персонала и технического обслуживания.
Надежность хранения информации в системе должна обеспечиваться применением аппаратно-программных методов организации данных с применением стандартных носителей.

7. ТРЕБОВАНИЯ К ДИАГНОСТИКЕ.

Диагностика аппаратной части и системного программного обеспечения ТС телематических служб должна производиться средствами, поставляемыми предприятиями-изготовителями средств вычислительной техники и программного обеспечения. Указанные средства должны включать тестовое ПО комплекса технических средств телематических служб, обеспечивающее проверку работоспособности ТС телематических служб и диагностику.

Средства диагностики сервера (узла) телематических служб не должны нарушать целостность и корректность данных.

8. ТРЕБОВАНИЯ К ЭЛЕКТРОПИТАНИЮ.

Система должна быть работоспособной при электропитании оборудования системы от источников бесперебойного электропитания, обеспечивающих на выходе напряжение 220 В с частотой 50 Гц и допустимыми отклонениями напряжения от минус 15% до +10% и частоты ± 5 Гц.

В случае пропадания электропитания источники гарантированного питания должны обеспечить работоспособность аппаратуры сервера (узла) телематических служб в течение не менее 5 минут для выполнения корректного закрытия системы и выполнения процедур, обеспечивающих сохранность информации.

9. ТРЕБОВАНИЯ К ЭЛЕКТРОБЕЗОПАСНОСТИ И ЭЛЕКТРОМАГНИТНОЙ СОВМЕСТИМОСТИ.

Технические средства телематических служб должны отвечать общим требованиям электрической и механической безопасности, требованиям электромагнитной совместимости и должны иметь соответствующий сертификат соответствия.

Конструкция и монтаж аппаратных средств системы должны исключать возможность прикосновения обслуживающего персонала к токоведущим частям. Компьютеры и периферийные устройства, входящие в состав ТС телематических служб должны быть подключены к защитному заземлению (занулению).

10. ТРЕБОВАНИЯ ПО УСТОЙЧИВОСТИ К КЛИМАТИЧЕСКИМ ФАКТОРАМ.

ТС телематических служб должен оставаться работоспособным при температуре окружающего воздуха от 5 до 40 град. С и относительной влажности от 20 до 80 % (без конденсата).

ТС телематических служб должен сохранять свои параметры во всем диапазоне рабочих температур при изменении напряжения первичного источника электропитания в допустимых пределах.

11. ТРЕБОВАНИЯ К ДОКУМЕНТАЦИИ.

В состав документации на ТС телематических служб должны входить следующие обязательные документы

Технические условия;

Комплект эксплуатационной документации.

Технические условия на ТС телематических служб должны быть выполнены на русском языке и соответствовать требованиям настоящих ОТТ.

Комплект эксплуатационной документации должен быть выполнен на русском языке и должен содержать:

- общее описание, включая контрольный пример;
- руководства администратора и оператора.

ПРИЛОЖЕНИЕ 1

ТЕХНИЧЕСКИЕ ТРЕБОВАНИЯ К ТЕХНИЧЕСКИМ СРЕДСТВАМ СЛУЖБЫ ЭЛЕКТРОННОЙ ПОЧТЫ ПО ПРОТОКОЛУ SMTP

1. ОБЛАСТЬ ПРИМЕНЕНИЯ

Настоящее приложение описывает технические требования к ТС службы ЭП по протоколу SMTP в соответствии с RFC 821 [1].

В приложении приведены передача сообщений электронной почты другим серверам электронной почты по протоколу SMTP по сети передачи данных в соответствии с адресом получателя, промежуточное временное накопление сообщений для дальнейшей передачи, а также доставка сообщений в локальный ящик электронной почты в соответствии с указанным именем ящика.

Не все функции, содержащиеся в данном приложении, обязательны для ТС служб ЭП по протоколу SMTP, но если они выполняются, то их реализация должна соответствовать настоящему приложению.

2. ФУНКЦИОНАЛЬНЫЕ ТРЕБОВАНИЯ К SMTP

2.1. Соединения

2.1.1. Протокол нижнего уровня

При использовании TCP для организации соединения клиента и сервера должен использоваться порт 25. При кодировании сообщений SMTP должно учитываться, что соединение TCP поддерживает длину байта 8 бит. Семибитные символы сообщений SMTP должны быть выровнены вправо, а старший бит октета установлен в 0.

2.1.2. Установление соединений.

В результате запроса клиента передатчик SMTP устанавливает дуплексное соединение с приемником SMTP.

Протокол SMTP должен предоставлять механизм передачи почты непосредственно с узла передающего клиента на узел получающего клиента при условии, что эти два узла соединены единой транспортной службой.

Протокол SMTP должен предоставлять механизм передачи почты путем пересылки между одним и более серверами SMTP, если два узла клиентов не соединены единой транспортной службой.

2.2. Взаимодействие

По запросу клиента передатчик SMTP устанавливает дуплексное соединение транспортного уровня с приемником SMTP. Приемник SMTP может быть либо промежуточным узлом, либо оконечным узлом адресата. Приемник и передатчик обмениваются командами и ответами.

После установления соединения транспортного уровня приемник должен выдать ответ приветствия 220.

Первой командой в сессии должна быть команда HELO

Последней командой сессии должна быть команда QUIT.

Элементы взаимодействия по протоколу SMTP приведены в п.8.

3. СООБЩЕНИЯ

Сообщения SMTP. посылаемые передатчиком SMTP приемнику SMTP, называются командами. Сообщения SMTP, посылаемые приемником SMTP передатчику SMTP, называются ответами.

Команды и ответы состоят из символов кода ASCII.

3.1. Команды

Командами являются символьные строки, заканчивающиеся <CRLF>. Команды состоят из кода команды и последующего поля аргументов. Коды команды и аргументы должны быть разделены одним или более пробелами. Регистр символов кода команды и названий параметров, таких как "to:" или "from:", не является существенным. Регистр аргументов прямого и обратного пути является существенным. Поле аргумента состоит из строки символов переменной длины, заканчивающееся <CRLF>.

3.1.1. Перечень команд

Перечень команд SMTP приведен в табл. 1

Таблица 1

Перечень команд SMTP

Команда	HELO <domain>
Аргументы	domain - имя узла передатчика SMTP
Описание	Используется для идентификации передатчика SMTP приемником SMTP.
Действия с буферами	Все таблицы состояний и буферы очищены.

Команда	MAIL FROM:<reverse-path>
Аргументы	reverce-path – обратный путь. Состоит из списка узлов и почтового ящика отправителя.
Описание	Указывает на передачу почты. Наличие списка узлов в обратном пути показывает, что данное почтовое сообщение было переслано через каждый из указанных узлов. Данный список используется в качестве маршрута для пересылки недоставленной почты отправителю. При каждой пересылке пересылающий узел добавляет свое имя в начало списка. Если узел имеет несколько имен, должно использоваться имя, известное в системе назначения.
Действия с буферами	Очищаются буферы обратного пути, буферы прямого пути, буфер данных почты. В буфер обратного пути помещается данные аргумента команды.

Команда	RCPT TO <forward-path>
Аргументы	forward-path - прямой путь. Состоит из списка узлов и почтового ящика адресата
Описание	Идентифицирует индивидуального получателя данных почты. Несколько получателей определяются использованием множества команд RCPT. Наличие списка узлов в прямом пути указывает, что почтовое сообщение должно быть передано следующему узлу из списка. Если приемник SMTP не поддерживает функцию пересылки, он должен выдать ответ 550 (неизвестный локальный клиент). При передаче почтового сообщения передающий узел должен удалить свое имя из списка прямого пути. При достижении почтовым сообщением оконечного адресата (при этом прямой путь будет содержать только имя почтового ящика) приемник SMTP должен поместить почтовое сообщение в почтовый ящик с именем, указанным в прямом пути.
Действия с буферами	Аргумент прямого пути добавляется в буфер прямого пути

Команда	DATA
Аргументы	-
Описание	Приемник обрабатывает строки, следующие за этой командой как данные почты, направляемой от передатчика. Полученные почтовые данные добавляются к буферу данных. Почтовые данные должны заканчиваться последовательностью "<CRLF>.<CRLF>". После окончания получения почтовых данных сервер начинает их обработку с использованием информации из буфера обратного пути, прямого пути и буфера почтовых данных. По окончании выполнения данной команды эти буферы должны быть очищены. В случае удачного выполнения команды приемник должен выдать ответ OK. Когда приемник SMTP получает почтовое сообщение для пересылки или для окончательной доставки, он должен вставлять в начало почтовых данных линию штампа времени. В штампе времени должны указываться: узел-отправитель, узел - получатель (приемник данной команды), дата и время получения сообщения. Когда приемник SMTP выполняет окончательную доставку почтового сообщения, он должен вставлять в начало почтовых данных информацию о линии обратного пути. Вставляемая информация должна быть взята из аргумента "обратный путь" команды MAIL. В случае, если пересылка почты выполнена только частично (только части указанных адресатов), сервер SMTP должен выдать ответ OK и извещения о непересланных сообщениях. Может быть либо одно извещение с перечнем всех неудачных адресатов, либо для каждого неудачного адресата должно быть выслано отдельное извещение. Все извещения о недоставке должны посылаться с помощью команды MAIL.
Действия с буферами	Буферы обратного пути, прямого пути и буфер данных сбрасываются.

Команда	SEND FROM:<reverse-path>
Аргументы	reverse-path - обратный путь
Описание	Используется для инициации транзакции, в которой почта доставляется одному или более терминалам. Обратный путь может состоять из необязательного списка узлов и имени почтового ящика отправителя. Если присутствует список узлов, он указывает на узлы, через которые пересылалось данное почтовое сообщение. Данный список используется для посылки отправителю извещений о недоставке.
Действия с буферами	Буферы обратного пути, прямого пути и буфер данных сбрасываются. Информация из аргумента обратного пути вставляется в буфер обратного пути.

Команда	SOML FROM:<reverse-path>
Аргументы	reverse-path - обратный путь
Описание	Используется для инициации транзакции, в которой почта доставляется одному или более терминалам или почтовым ящикам. Данные почты для каждого адресата доставляются на терминал, если он активен, или в почтовый ящик в противном случае. Назначение аргумента аналогично команде SEND.
Действия с буферами	Буферы обратного пути, прямого пути и буфер данных сбрасываются. Информация из аргумента обратного пути вставляется в буфер обратного пути.

Команда	SAML FROM:<reverse-path>
Аргументы	Reverse-path - обратный путь
Описание	Используется для инициации транзакции, в которой почта доставляется одному или более терминалам и почтовым ящикам. Данные почты для каждого адресата доставляются на терминал, если он активен, и обязательно в почтовый ящик. Назначение аргумента аналогично команде SEND.
Действия с буферами	Буферы обратного пути, прямого пути и буфер данных сбрасываются. Информация из аргумента обратного пути вставляется в буфер обратного пути.

Команда	RSET
Аргументы	-
Описание	Показывает, что текущая транзакция должна быть прекращена, все запомненные данные уничтожены, все буферы очищены. Приемник должен ответить OK.
Действия с буферами	Все сохраненные данные уничтожаются, все буферы сбрасываются.

Команда	VRFY <string>
Аргументы	string – предполагаемое имя клиента
Описание	Данная команда просит приемник подтвердить, что аргумент идентифицирует клиента. Если аргумент содержит имя клиента, приемник должен выдать ответ с полным именем клиента, если оно известно, и полным именем почтового ящика.
Действия с буферами	-

Команда	EXPN <string>
Аргументы	string – предполагаемый идентификатор списка рассылки
Описание	Данная команда просит приемник подтвердить, что аргумент идентифицирует список рассылки. Если аргумент содержит список рассылки, приемник должен выдать многострочный ответ с перечнем полных имен клиентов, если они известны, и полных имен почтовых ящиков, занесенных в данных список рассылки.
Действия с буферами	-

Команда	HELP [<string>]
Аргументы	string - имя команды
Описание	По данной команде приемник должен выдать ответ с полезной для передатчика информацией.
Действия с буферами	-

Команда	NOOP
Аргументы	-
Описание	Нет операции. Приемник должен выдать ответ OK.
Действия с буферами	-

Команда	QUIT
Аргументы	-
Описание	Приемник должен выдать ответ OK и закрыть соединение.
Действия с буферами	Сброс всех данных и буферов.

Команда	TURN
Аргументы	-
Описание	Приемник должен либо выдать ответ OK и взять на себя роль передатчика, либо выдать ответ отказа 502 и остаться в роли приемника. Если обмен ролями произошел, процесс, ставший приемником высылает ответ приветствия 220.
Действия с буферами	Сброс всех данных и буферов.

3.1.2. Синтаксис команд определен в п.5.

3.1.3. Команды: HELO, MAIL, RCPT, DATA, RSET, NOOP, QUIT должны быть реализованы обязательно.

3.1.4. Обеспечение прозрачности передачи данных в команде DATA

При посылке передатчиком данных почты каждую последовательность "<CRLF>." (0x0D 0x0A 0x2E) передатчик должен заменять на "<CRLF>.."(0x0D 0x0A 0x2E 0x2E). Приемник должен выполнять обратное преобразование. Указатель конца почтовых данных этому преобразованию не подвергается.

3.2. Ответы

3.2.1. Код ответа

Ответ SMTP состоит из трехзначного кода ответа (передаваемого как три символа), за которым следует текст.

Значения номера ответа:

первая цифра

1	Положительный предварительный ответ
2	Положительный окончательный ответ .
3	Положительный промежуточный ответ
4	Временный отрицательный окончательный ответ
5	Постоянный отрицательный окончательный ответ

вторая цифра

0	Ошибки синтаксиса
1	Запрос информации
2	О состоянии соединения
3	не определен
4	не определен
5	О состоянии почтовой системы

третья цифра позволяет сделать более точное разделение значений ответов по функциональным категориям, определенным второй цифрой.

Ответ сервера может состоять из одной или нескольких строк.
Однострочный ответ состоит из:
трехзначного номера ответа, передаваемого как три символа,
символа <SP>,
текста,
символа <CRLF> .
Многострочный ответ состоит из:
трехзначного номера ответа, передаваемого как три символа,
символа "-"
текста первой строки
символа <CRLF>

трехзначного номера ответа, передаваемого как три символа,
символа "-"
текста второй строки
символа <CRLF>

.....
трехзначного номера ответа, передаваемого как три символа,
символа <SP>,
текста последней строки,
символа <CRLF> .

Список кодов ответов приведен в табл. 2. Для всех ответов, кроме 110, текст ответа не обязательно должен соответствовать приведенному в табл. 2.

Таблица 2

Список кодов ответов

Код	Текст
211	Системный статус или ответ системной помощи
214	Ответ помощи
220	<домен> Служба готова
221	<домен> Служба закрывает соединение
250	Запрошенное действие выполнено успешно
251	Клиент не локальный, направлено в <прямой путь>
354	Начинаю получение почтовых данных. Конец при <CRLF>.<CRLF>
421	<домен> Служба не доступна, закрываю соединение
450	Запрошенное действие не принято. Почтовый ящик недоступен (например, занят)
451	Запрошенное действие прервано. Локальная ошибка выполнения.
452	Запрошенное действие не принято. Недостаточно памяти.
500	Синтаксическая ошибка, команда не распознана
501	Синтаксическая ошибка в параметре или аргументах
502	Команда не реализована
503	Неправильная последовательность команд.
504	Аргумент команды не реализован
550	Запрошенное действие не принято. Почтовый ящик не доступен (например, не найден)
551	Клиент не локальный. Пожалуйста, попробуйте <прямой путь>
552	Запрошенное действие прервано. Превышен лимит памяти.
553	Запрошенное действие не принято. Неправильное имя почтового ящика.
554	Ошибка транзакции.

3.3. Порядок команд и ответов

Первой командой в сессии должна быть команда HELO. Если аргумент команды HELO является неприемлемым, должен быть выдан ответ 501 и приемник SMTP должен остаться в прежнем состоянии.

Команды NOOP, HELP, EXPN, VRFY могут использоваться в любое время в течении сессии.

Команды MAIL, SEND, SOML, SAML начинают транзакцию. Если аргумент команды начала транзакции является неприемлемым, приемник должен выдать ответ 501 и остаться в прежнем состоянии. Во время транзакции должны использоваться команды в следующей последовательности: одна или несколько команд RCPT, одна команда DATA. Транзакция может быть прервана командой RSET. В течение сессии может быть 0, 1 или более транзакций. Если во время транзакции команды выдаются с нарушением указанного порядка, приемник должен выдать ответ 503 и остаться в прежнем состоянии.

Последней командой сессии должна быть команда QUIT. Команда QUIT может быть выдана в любое время в течение сессии.

На каждую команду должен выдаваться точно один ответ.

В п.6 и п.7 определяются допустимые последовательности команд и ответов.

3.4. Ограничения на размер элементов сообщений SMTP

Ограничения на размер элементов сообщений SMTP приведены в табл. 3

Таблица 3

Ограничения на размер элементов сообщений SMTP.

Обозначение элемента	Элемент	Максимальный размер
User	имя клиента	64 символов
Domain	имя домена	64 символов
Path	обратный путь или прямой путь	256 символов
Command line	Строка команды включая символы <CRLF>	512 символов
reply line	Строка ответа включая код ответа и символы <CRLF>	512 символов
text line	Строка данных почты, включая символы <CRLF>, но не считая символы точки, добавленные для обеспечения прозрачности	1000 символов
Recipient buffer	Емкость буфера адресатов	100 адресатов

4. ОПИСАНИЕ СИНТАКСИСА КОМАНД И ОТВЕТОВ

<HELO> ::= "HELO" 1*<SP> <domain> <CRLF>

<MAIL> ::= "MAIL" 1*<SP> "FROM:" <reverse-path> <CRLF>

<RCPT> ::= "RCPT" 1*<SP> "TO:" <forward-path> <CRLF>

<DATE> ::= "DATA" <CRLF>

<RSET> ::= "RSET" <CRLF>

<SEND> ::= "SEND" 1*<SP> "FROM:" <reverse-path> <CRLF>

<SOML> ::= "SOML" 1*<SP> "FROM:" <reverse-path> <CRLF>

<SAML> ::= "SAML" 1*<SP> "FROM:" <reverse-path> <CRLF>

<VRFY> ::= "VRFY" 1*<SP> <string> <CRLF>

<EXPN> ::= "EXPN" 1*<SP> <string> <CRLF>

<HELP> ::= "HELP" [1*<SP> <string>] <CRLF>

<NOOP> ::= "NOOP" <CRLF>

<QUIT> ::= "QUIT" <CRLF>

<TURN> ::= "TURN" <CRLF>

<reverse-path> ::= <path>

<forward-path> ::= <path>

<path> ::= "<" [<a-d-l> ":"] <mailbox> ">"

<a-d-l> ::= <at-domain> | <at-domain> "," <a-d-l>

<at-domain> ::= "@" <domain>

<domain> ::= <element> | <element> "." <domain>

<element> ::= <name> | "#" <number> | "[" <dotnum> "]"

<mailbox> ::= <local-part> "@" <domain>

<local-part> ::= <dot-string> | <quoted-string>

<name> ::= <a> <ldh-str> <let-dig>

<ldh-str> ::= <let-dig-hyp> | <let-dig-hyp> <ldh-str>

<let-dig> ::= <a> | <d>

<let-dig-hyp> ::= <a> | <d> | "-"

<dot-string> ::= <string> | <string> "." <dot-string>

<string> ::= <char> | <char> <string>

<quoted-string> ::= """ <qtext> """

<qtext> ::= "\" <x> | "\" <x> <qtext> | <q> | <q> <qtext>

<char> ::= <c> | "\" <x>

<dotnum> ::= <snum> "." <snum> "." <snum> "." <snum>

<number> ::= <d> | <d> <number>

<CRLF> ::= <CR> <LF>

<CR> ::= символ возврата каретки (код ASCII 13)

<LF> ::= символ следующей строки (код ASCII 10)

<SP> ::= символ пробела (код ASCII - 32)

<snum> ::= одна. две или три десятичные цифры, представляющие
десятичное число в диапазоне от 0 до 255

<a> ::= любой из 52 алфавитных строчных и прописных
символа от A до Z и от a до z

<c> ::= любой из 128 символов ASCII кроме
<special> or <SP>

<d> ::= любая из 10 цифр от 0 до 9

<q> ::= любой из 128 символов ASCII кроме <CR>,
<LF>, кавычек ("), или (\)

<x> ::= любой из 128 символов ASCII

<special> ::= "<" | ">" | "(" | ")" | "[" | "]" | "\" | "."
| "," | ";" | ":" | "@" """ | управляющие
символы (коды ASCII от 0 до 31 включительно, а так же
127)

Примечание 1: символ "\" указывает на то, что следующий за ним специальный символ интерпретируется "буквально", а не в соответствии с обычной интерпретацией.

Примечание 2: для именования узлов используются две дополнительные числовые формы. Первая форма состоит из символа "#", за которым следует целое десятичное число, являющееся адресом узла. Вторая форма состоит из 4-х целых десятичных чисел, разделенных точками и заключенных в квадратные скобки. Вторая форма соответствует адресу IP.

<return-path-line> ::= "Return-Path:" <SP><reverse-path><CRLF>

<time-stamp-line> ::= "Received:" <SP> <stamp> <CRLF>

<stamp> ::= <from-domain> <by-domain> <opt-info> ";"
<daytime>

<from-domain> ::= "FROM" <SP> <domain> <SP>

<by-domain> ::= "BY" <SP> <domain> <SP>

<opt-info> ::= [<via>] [<with>] [<id>] [<for>]

<via> ::= "VIA" <SP> <link> <SP>

<with> ::= "WITH" <SP> <protocol> <SP>

<id> ::= "ID" <SP>'<string> <SP>

<for> ::= "FOR" <SP> <path> <SP>

<link> ::= Стандартные имена соединений (links),
зарегистрированные в организации Network
Information Center

<protocol> ::= Стандартные имена протоколов, зарегистрированные
в организации Network Information Center

<daytime> ::= <SP> <date> <SP> <time>

<date> ::= <dd> <SP> <mon> <SP> <yy>

<time> ::= <hh> ":" <mm> ":" <ss> <SP> <zone>

<dd> ::= однозначное или двузначное десятичное число,
обозначающее день в диапазоне от 1 до 31

<mon> ::= "JAN" | "FEB" | "MAR" | "APR" | "MAY" | "JUN" |
"JUL" | "AUG" | "SEP" | "OCT" | "NOV" | "DEC"

<yy> ::= двузначное десятичное число, обозначающее год столетия
в диапазоне от 00 до 99

<hh> ::= двузначное десятичное число, обозначающее часы дня в
диапазоне от 00 до 23

<mm> ::= двузначное десятичное число, обозначающее минуты часа в
диапазоне от 00 до 59

<ss> ::= двузначное десятичное число, обозначающее секунды
минуты в диапазоне от 00 до 59

<zone> ::= "UT" для Универсального Времени (по умолчанию)
или другого часового пояса в соответствии с RFC 822[2]

5. ПОСЛЕДОВАТЕЛЬНОСТЬ КОМАНД И ОТВЕТОВ

Используются следующие сокращения:

I - промежуточный положительный ответ
S - успешное выполнение
F - неудача
E - ошибка

Установление соединения
S: 220
F: 421
HELO
S: 250
E: 500, 501, 504, 421
MAIL

S: 250
F: 552, 451, 452
E: 500, 501, 421

RCPT
S: 250, 251
F: 550, 551, 552. 553, 450, 451, 452
E: 500, 501, 503. 421

DATA
I: 354 -> data -> S: 250
F: 552, 554, 451, 452
F: 451, 554
E: 500, 501, 503, 421

RSET
S: 250
E: 500, 501, 504, 421

SEND
S: 250
F: 552, 451, 452
E: 500, 501, 502, 421

SOML
S: 250
F: 552, 451, 452
E: 500, 501, 502, 421

SAML
S: 250
F: 552, 451, 452
E: 500, 501, 502, 421

VRFY
S: 250, 251
F: 550, 551, 553
E: 500, 501, 502, 504, 421

EXPN
S: 250
F: 550
E: 500, 501, 502, 504, 421

HELP
S: 211, 214
E: 500, 501, 502, 504, 421

NOOP
S: 250
E: 500, 421

QUIT
S: 221
E: 500

TURN
S: 250
F: 502
E: 500, 503

6. ДИАГРАММЫ СОСТОЯНИЙ СЕРВЕРА SMTP

Диаграмма состояний сервера SMTP для команд HELO, MAIL, RCPT, RSET, SEND, SOML, SAML, VRFY. EXPN, HELP, NOOP, QUIT, TURN приведены на рис.1. Диаграмма состояний сервера SMTP для команды DATA приведена на рис.2.

Используемые сокращения на рис.1 и рис.2:

B - Начало
S - Успешное выполнение
F - Неудача
E - Ошибка
W - Ожидание ответа
data - серия линий с данными почты, посылаемых от передатчика приемнику.
M - Среднее состояние
Цифрами обозначены возможные номера ответов, что соответствует первой цифре трехзначного кода ответа, как это описано в пункте 4.2.1.

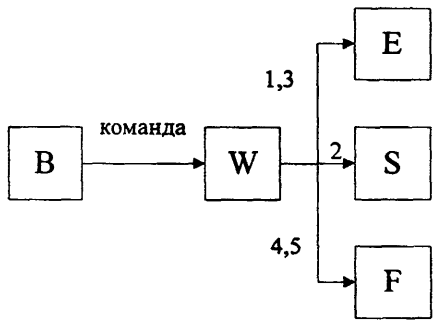


Рис. 1 Диаграмма состояний сервера SMTP для команд HELO, MAIL, RCPT, RSET, SEND, SOML, SAML, VRFY, EXPN, HELP, NOOP, QUIT, TURN.

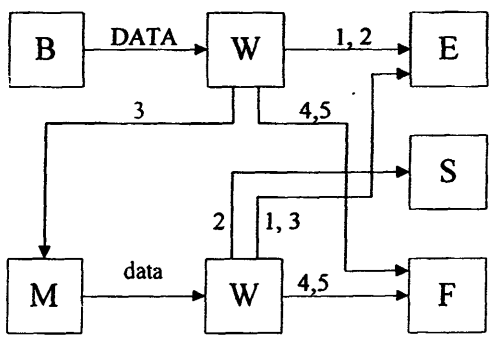


Рис. 2 Диаграмма состояний сервера SMTP для команды DATA.

7. ОПИСАНИЕ ПРОЦЕДУР SMTP

На рис. 3. показана схема соединений при взаимодействии SMTP.

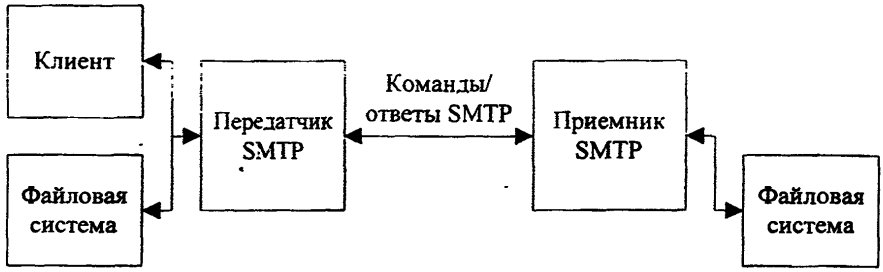


Рис. 3. Схема соединений при взаимодействии SMTP.

Рассматривают следующие процедуры SMTP во время взаимодействия SMTP:

- транзакция.
- направление.
- верификация.
- доставка в почтовый ящик
- доставка на терминал клиента,
- открытие и закрытие соединения

7.1. Открытие и закрытие соединения

Для открытия соединения используется команда HELO. Этой командой идентифицируется узел передатчика SMTP.

HELO <домен>

Для закрытия соединения используется команда: QUIT

Во время соединения приемник и передатчик могут поменяться ролями. Для этого используется команда TURN. Инициатором обмена ролями должен быть передатчик. Для отказа обмена используется ответ 502. Данная команда не должна использоваться в случае, если в качестве протокола транспортного уровня используется TCP.

7.2. Транзакция.

В результате запроса клиента передатчик SMTP устанавливает дуплексное соединение с получателем SMTP. Приемником SMTP может быть либо адресат, либо промежуточный пункт.

Сразу после установления соединения передатчик SMTP посылает команду MAIL, указывающую отправителя почты. Если приемник SMTP может принять почту, он отвечает OK. После этого передатчик SMTP посылает команду RCPT, указывающую получателя почты. Если приемник SMTP может принять почту для данного получателя, он отвечает

OK. Если нет, он высылает ответ, отклоняющий данного получателя (но не всю транзакцию). Передатчик SMTP и приемник SMTP могут согласовать нескольких получателей. После завершения согласования получателей, передатчик SMTP отправляет данные почты, заканчивающиеся определенной последовательностью. Если приемник SMTP успешно обработал полученные данные, он отвечает OK.

Таким образом транзакция состоит из трех шагов, что отображено на табл. 4.

Таблица 4

Шаги транзакции

Шаг транзакции	Команда	Описание шага транзакции
1.	MAIL from:<обратный путь>	На данном шаге происходит идентификация отправителя, сброс всех таблиц состояния и буферов. Устанавливается обратный путь, используемый для передачи сообщений об ошибках. Если приемник SMTP принимает данную команду, от выдает ответ 250 OK.
2.	RCPT to: <прямой путь 1> .... RCPT to: <прямой путь N>	На данном шаге идентифицируется один или больше адресатов. Каждая команда RCPT идентифицирует одного адресата. Если приемник SMTP принимает команду RCPT, он выдает ответ 250 OK. Если адресат неизвестен, приемник SMTP выдает ответ 550 Failure. Описанный шаг повторяется для каждого адресата.
3.	DATA	Если приемник SMTP принимает данную команду, он выдает ответ 354 Intermediate. Следующие за командой DATA строки приемник SMTP воспринимает как текст почтового сообщения. После того, как все строки приняты и запомнены, приемник SMTP выдает ответ 250 OK.

Пример процедуры Транзакция:

S: MAIL FROM:<Smith@Alpha.ARPA>
R: 250 OK

S: RCPT TO:<Jones@Beta.ARPA>
R: 250 OK

S: RCPT TO:<Green@Beta.ARPA>
R: 550 No such user here

S: RCPT TO:<Brown@Beta.ARPA>
R: 250 OK

S: DATA
R: 354 Start mail input; end with <CRLF>.<CRLF>
S: Blah blah blah...
S: ...etc. etc. etc.
S: <CRLF>.<CRLF>

R: 250 OK

7.3. Направление

В случаях, когда информация аргумента <прямой путь> команды RCPT оказывается некорректной, но приемник SMTP знает адресат, должны использоваться следующие ответы:

251 Клиент не локальный. Сообщение переслано в <прямой путь>

551 Клиент не локальный. Попробуйте <прямой путь>

В первом случае приемник SMTP сам пересылает сообщение и затем информирует клиента в том, что в следующий раз он должен использовать указанный в ответе прямой путь.

Во втором случае клиенту будет возвращена ошибка, и клиент сам должен повторить отправку сообщения с указанием нового прямого пути.

7.4. Верификация

Для предоставления возможностей верификации пользователей и списка рассылки существуют команды VRFY и EXPN.

По команде VRFY приемник по введенному имени клиента выдает полное имя клиента и имя его почтового ящика.

По команде EXPN приемник по идентификатору списка рассылки выдает многострочный ответ, содержащий полные имена клиентов и имена их почтовых ящиков, включенных в данный список рассылки.

7.5. Доставка в почтовый ящик и доставка на терминал клиента

Доставка почты на терминал клиента осуществляется командой

SEND from:<обратный путь>

Если адресат неактивен или не принимает почтовое сообщение, отправитель получит ответ 450.

Команда SOML позволяет доставить почту на терминал клиента, если он активен и принимает сообщения, или оставить в почтовом ящике, в противном случае.

Команда SAML позволяет доставить почту на терминал клиента (если он активен) и поместить ее в почтовый ящик.

7.6. Пересылка

При пересылке сообщений сервер SMTP добавляет свой идентификатор в обратный путь, удаляет первый элемент прямого пути в случае, если он совпадает с его идентификатором. Сообщение пересылается на узел, соответствующий первому элементу прямого пути.

Если сервер SMTP взял на себя задачу пересылки почты и позднее обнаружил, что почта по какой-либо причине не может быть доставлена, он должен составить извещение о невозможности доставить почту и выслать его отправителю. Сервер SMTP не должен высылать извещения о проблемах пересылки извещений. Для этого при отправке извещения используется команда с нулевым обратным путем.

ПРИЛОЖЕНИЕ 2

ТЕХНИЧЕСКИЕ ТРЕБОВАНИЯ К ТЕХНИЧЕСКИМ СРЕДСТВАМ СЛУЖБЫ ЭЛЕКТРОННОЙ ПОЧТЫ ПО ПРОТОКОЛУ POP3

1. ОБЛАСТЬ ПРИМЕНЕНИЯ

Настоящее приложение описывает технические требования к ТС службы ЭП по протоколу POP3 в соответствии с RFC 1939 [3] и RFC 1734[4].

В приложении приведен удаленный доступ пользователей по сети передачи данных общего пользования к функциям сервера электронной почты .

Не все функции. содержащиеся в данном приложении, обязательны для ТС служб ЭП по протоколу POP3, но если они выполняются, то их реализация должна соответствовать настоящему приложению.

2. ФУНКЦИОНАЛЬНЫЕ ТРЕБОВАНИЯ К ВЗАИМОДЕЙСТВИЮ КЛИЕНТА POP3 И СЕРВЕРА POP3.

2.1. Соединения

2.1.1. Протокол нижнего уровня

Для организации соединения клиента и сервера должен использоваться протокол TCP. На стороне сервера должен использоваться порт 110.

2.1.2. Общая структура протокола

Сессия протокола POP3 состоит из установления соединения клиент/сервер, начального приветствия от сервера и взаимодействия клиент/сервер. В ходе сессии сервер и клиент обмениваются сообщениями. Сообщения разделяются на команды клиента и ответы сервера. В ответы сервера включаются данные, передаваемые сервером клиенту и результаты выполнения команд. Все сообщения передаются клиентом и сервером в форме строк, заканчивающимися символом <CRLF>.

2.2. Взаимодействие

2.2.1. Состояния сервера

В сервере должны быть реализованы следующие состояния:

AUTHORIZATION
TRANSACTION
UPDATE

Типичная последовательность переходов состояний сервера показана на рис. 1.

2.2.1.1. Сервер должен находиться в состоянии AUTHORIZATION после установления соединения TCP и выдачи им ответа приветствия.

Если процесс идентификации прошел успешно (т.е. клиент может получить доступ к соответствующему почтовому ящику), сервер открывает почтовый ящик и переходит в состояние TRANSACTION. При этом все метки удаления почтовых сообщений сброшены.

Если происходит ошибка при открытии почтового ящика, сервер отвечает отрицательным сообщением. После отрицательного ответа сервер может закрыть соединение. В случае, если соединение не закрыто, в сервере должны быть реализованы повторная идентификация и команда QUIT.

После того, как сервер открыл почтовый ящик, для каждого почтового сообщения должен быть выделен номер, начиная с 1. Номера должны последовательно возрастать на 1. В командах и ответах номера почтовых сообщений должны быть представлены в десятичном формате.

В состоянии AUTHORIZATION должны быть реализованы команды как минимум одного из механизмов идентификации, приведенных в п. 3.3.2., и команда QUIT.

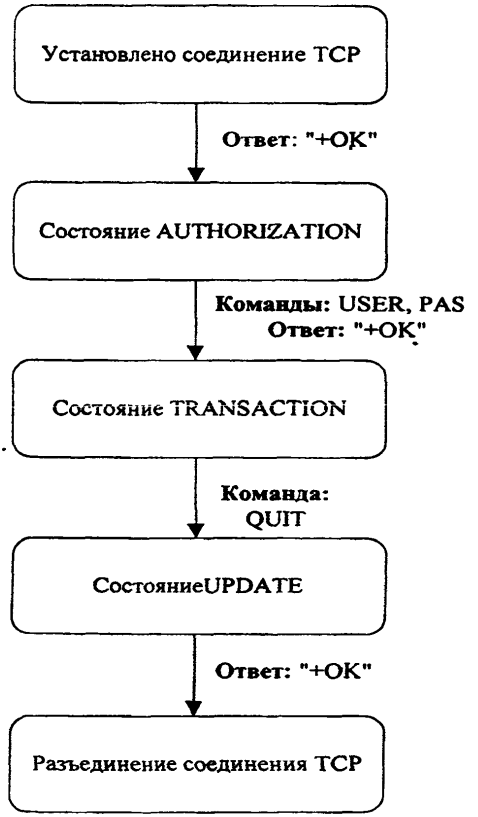


Рис. 1 Типичная последовательность состояния сервера

2.2.1.2. Состояние TRANSACTION

Сервер переходит в состояние TRANSACTION в случае успешной идентификации клиента и успешного открытия почтового ящика.

Если сервер получает команду QUIT, он должен перейти в состояние UPDATE. В состоянии TRANSACTION должны быть реализованы следующие команды: STAT, LIST, RETR, DELE, NOOP, RSET, QUIT

2.2.1.3. Состояние UPDATE

Сервер переходит в состояние UPDATE в случае получения команды QUIT в состоянии TRANSACTION. В состоянии UPDATE (и только в этом состоянии) сервер удаляет из почтового ящика сообщения, помеченные как удаленные.

2.2.2. Механизмы идентификации клиента

В сервере могут быть реализованы следующие механизмы идентификации клиента:
- с помощью команд USER и PASS
- с помощью команды APOP
- с помощью команды AUTH (дополнительный механизм идентификации)

2.2.2.1. На сервере должен быть реализован механизм идентификации с помощью команд USER и PASS. Команда PASS должна следовать непосредственно за командой USER. Формат команд USER и PASS описан в п. 4.1. настоящего Руководящего Документа.

2.2.2.2. Механизм идентификации с помощью команды APOP позволяет избежать посылки пароля по сети. Сервер, в котором реализована команда APOP, должен в сообщение приветствия включить метку времени. Должна обеспечиваться уникальность метки времени для каждого сообщения приветствия, выдаваемого сервером.

Клиент должен запомнить последнюю присланную ему метку времени и использовать ее в команде APOP.

Формат команды APOP: APOP <name> <digest>

Более подробно формат команды описан в п. 4.1. настоящего Руководящего Документа.

Значение параметра name такое же, как в команде NAME. Параметр digest должен вычисляться согласно алгоритму MD5 в соответствии с RFC 1321[5], применяемому к метке времени и следующей за ней строке пароля. Параметр digest должен иметь формат 16-октетного числа в 16-ричном формате, представленного в ASCII - коде, используя строчные буквы.

Получая команду APOP, сервер должен проверить соответствие параметра digest. В случае соответствия выдается положительный ответ, и сервер переходит в состояние TRANSACTION. В случае несоответствия сервер выдает отрицательный ответ и остается в состоянии AUTHORIZATION.

2.2.2.3. Механизм идентификации с помощью команды AUTH.

Механизмы идентификации и защиты, используемые командой AUTH в протоколе POP3 должны быть аналогичны используемым в протоколе IMAP4 и соответствовать RFC 1731[6], 1734[4].

Формат команды AUTH описан в п. 4.1. настоящего Руководящего Документа. Команда передает серверу параметр с указанием, какой механизм идентификации должен быть использован. Если в сервере поддерживается запрошенный механизм, сервер производит обмен сообщениями с клиентом для идентификации пользователя. Также может быть выполнено согласование механизма защиты для последующих сессий.

В случае, если команда AUTH не поддерживается, сервер должен послать отрицательный ответ.

Обмен сообщениями протокола идентификации состоит из серий вызовов, исходящих от сервера, и ответов клиента, являющихся специфичными для каждого механизма идентификации. В качестве вызова сервера используется ответ "готов". Вызов сервера должен иметь формат строки в коде BASE64 с первым символом "+". Ответ клиента должен иметь формат строки в коде BASE64. Ответ клиента "*" вызывает прерывание процедуры идентификации и последующий отрицательный ответ сервера. Если установлен механизм защиты, он применяется ко всем последующим пересылкам данных по соединению. Механизм защиты должен активизироваться сразу после получения символа <CRLF> положительного ответа сервера после обмена сообщениями протокола идентификации. При активном механизме защиты потоки символов команд и ответов преобразуются в зашифрованные буферы. Каждый буфер передается как поток октетов, которому предшествует 4-х октетное поле длины последующих данных. Максимальная длина шифрованного буфера определяется механизмом защиты.

2.2.3. В сервере может быть реализован таймер разъединения по отсутствию активности. Установленное время таймера разъединения по отсутствию активности должно быть как минимум 10 минут. Получение любой команды должно сбрасывать таймер. При закрытии сессии по отсутствию активности сервер:

- не должен входить в состояние UPDATE,
- не должен удалять сообщения,
- не должен посылать ответ клиенту.

3. ПЕРЕЧЕНЬ И СТРУКТУРА СООБЩЕНИЙ

Все сообщения протокола POP3 должны быть стандартными текстовыми сообщениями в соответствии с RFC822 [2].

3.1. Команды

Сообщения, посылаемые от клиента серверу, называются командами.

3.1.1. Структура команд

Команды состоят из печатных символов кода ASCII. Структура команды:

ключевое слово
один или более аргументов (могут отсутствовать)
символ <CRLF>

Ключевое слово и аргументы, а также аргументы между собой разделяются одним символом <SP> (пробел).

Максимальная длина аргумента составляет 40 символов.

Длина ключевого слова составляет 3 или 4 символа.

3.1.2. Перечень команд

Перечень команд приведен в табл. 1.

Таблица 1

Перечень команд

Команда	STAT
Аргументы	-
Описание	Информация о статусе почтового ящика.
Возможные ответы	+OK <статус>
Разрешенные состояния	TRANSACTION

Команда	LIST [<msg>]
Аргументы	msg - номер сообщения. Не должен ссылаться на удаленное сообщение.
Описание	Если указан аргумент, сервер высылает положительный однострочный ответ с информацией о данном сообщении. Если нет сообщения с таким номером, выдается отрицательный ответ. Если аргумент не указан, сервер высылает положительный многострочный ответ с информацией обо всех сообщениях в почтовом ящике. В случае, если почтовый ящик пуст, выдается положительный ответ, состоящий только из ".CRLF".
Возможные ответы	+OK <скан-список> -ERR
Разрешенные состояния	TRANSACTION

Команда	RETR <msg>
Аргументы	msg - номер сообщения
Описание	Сервер высылает положительный многострочный ответ с сообщением, соответствующим указанному номеру.
Возможные ответы	+OK <верх сообщения> -ERR
Разрешенные состояния	TRANSACTION

Команда	DELE <msg>
Аргументы	msg - номер сообщения
Описание	Сервер помечает сообщение как удаленное. Команда с аргументом, указывающим на несуществующее сообщение, либо на уже удаленное сообщение, вызывает отрицательный ответ.
Возможные ответы	+OK -ERR
Разрешенные состояния	TRANSACTION

Команда	NOOP
Аргументы	-
Описание	Нет операции
Возможные ответы	+OK
Разрешенные состояния	TRANSACTION
Команда	RSET
Аргументы	-
Описание	Сбрасываются все метки удаленных сообщений
Возможные ответы	+OK
Разрешенные состояния	TRANSACTION

Команда	QUIT
Аргументы	-
Описание	Удаление всех сообщений, отмеченных как удаленные из почтового ящика, закрытие почтового ящика и разрыв соединения TCP. Если при удалении сообщений возникает ошибка, выдается сообщение об ошибке.
Возможные ответы	+OK -ERR
Разрешенные состояния	TRANSACTION
Команда	QUIT
Аргументы	-
Описание	Разрыв соединения TCP.
Возможные ответы	+OK
Разрешенные состояния	AUTHORIZATION

Команда	TOP <msg> <n>
Аргументы	msg - номер сообщения n - число строк (положительное чмсло либо равное 0)
Описание	Сервер выдает положительный многострочный ответ, включающий: заголовок сообщения, пустую строку, указанное число строк сообщения.
Возможные ответы	+OK <ответ> -ERR
Разрешенные состояния	AUTHORIZATION

Команда	UIDL [<msg>]
Аргументы	msg - номер сообщения
Описание	В случае, если аргумент задан, сервер выдает однострочный положительный ответ "unique-id listening" для почтового сообщения с указанным номером. Если аргумент не задан, сервер выдает многострочный положительный ответ, состоящий из - первой строки +OK, - строк "unique-id listening" для каждого сообщения в почтовом ящике. Отрицательный ответ выдается в случае отсутствия сообщения с указанным номером.
Возможные ответы	+OK <список uid> -ERR
Разрешенные состояния	TRANSACTION

Команда	USER <name>
Аргументы	name - идентификатор почтового ящика
Описание	Идентификатор почтового ящика. Используется в механизме идентификации по комбинации команд USER PASS. Положительный ответ выдается, если существует почтовый ящик с указанным именем, отрицательный - если такой почтовый ящик отсутствует.
Возможные ответы	+OK -ERR
Разрешенные состояния	AUTHORIZATION

Команда	PASS <string>
Аргументы	string - пароль почтового ящика
Описание	Пароль почтового ящика Используется в механизме идентификации по комбинации команд USER PASS. Положительный ответ выдается, если клиенту позволен доступ к соответствующему почтовому ящику, и отрицательный - в противном случае.
Возможные ответы	+OK -ERR
Разрешенные состояния	AUTHORIZATION, после выполнения команды USER

Команда	APOP <name> <digest>
Аргументы	name - идентификатор почтового ящика digest - строка MD5
Описание	Пароль почтового ящика Используется в механизме идентификации по методу "разделяемого секрета". Положительный ответ выдается, если клиенту позволен доступ к соответствующему почтовому ящику, и отрицательный - в противном случае.
Возможные ответы	+OK -ERR
Разрешенные состояния	AUTHORIZATION

Команда	AUTH <str>
Аргументы	str - идентификатор идентификационного механизма в соответствии с RFC 1731[6]
Описание	Команда начала процедуры идентификации по механизму IMAP4
Возможные ответы	+OK -ERR
Разрешенные состояния	AUTHORIZATION

3.1.3. Обязательными для реализации являются команды: QUIT, STAT, LIST, RETR, DELE, NOOP, RSET, USER, PASS.

3.2. Ответы

Сообщения, посылаемые от сервера клиенту называются ответами.

3.2.1. Формат ответа

Ответы состоят из отображаемых символов кода ASCII. Каждый ответ заканчивается символом <CRLF>. Максимальная длина ответа, включая символы <CRLF>, составляет 512 символов.

Ответ состоит из индикатора статуса и последующего текста ответа. Индикатор статуса может принимать значения либо "+OK" либо "-ERR". Буквы в индикаторе статуса должны быть заглавными. Ответы с индикатором статуса "+OK" называются положительными, ответы с индикатором статуса "-ERR" называются отрицательными.

Содержание текста ответа зависит от типа ответа. Для ответов некоторых типов структура текста ответа является существенной. Данные типы ответов приведены в табл. 2. Структура и тексты ответов, не перечисленных в табл. 2 несущественны. Смысловое содержание текста ответа должно соответствовать логическому значению ответа. Исключением является ответ "готов" сервера при идентификации клиента по команде AUTH.

Ответы могут быть однострочными и многострочными. В многострочных ответах строки отделяются символами <CRLF>. Последняя строка многострочного ответа состоит из заключительного октета (с десятичным кодом 46 (".")) и символов <CRLF>. Если строка многострочного ответа начинается с символа "," , в начало этой строки добавляется еще один символ ".". Таким образом критерием конца многострочного ответа является

последовательность "CRLF.CRLF". Оконечная последовательность ".CLRF" не считается частью многострочного ответа.

На нереализованные или нераспознанные команды сервер должен отвечать отрицательным ответом. На команды, неразрешенные в данном состоянии, сервер должен отвечать отрицательным ответом.

3.2.2. Список ответов с фиксированной структурой текста ответа.

Список ответов с фиксированной структурой текста ответа приведен в табл. 2

Таблица 2

Список ответов с фиксированной структурой текста ответа

Ответ:	Приветствие
Структура	+OK +OK <timestamp>
Описание	Сообщение приветствия. Метка времени timestamp должна соответствовать RFC822[2] и должна включаться в сообщение, когда реализована команда APOP.

Ответ:	Статус
Структура	+OK <nn> <mm> nn - количество сообщений в почтовом ящике mm - размер почтового ящика в октетах
Описание	Статус почтового ящика

Ответ:	скан-список
Структура	+OK <n> <m> ; для однострочного ответа n - номер сообщения m - точный размер сообщения --- +OK ; для многострочного ответа <n> <m> ... <n> <m> . n - номер сообщения m – точный размер сообщения
Описание	Данные о сообщении. Точный размер сообщения должен соответствовать размеру передаваемого сообщения, который может отличаться от размера хранимого сообщения.

Ответ:	Сообщение
Структура	+OK <line1> ... <linen> . line1 .. linen - строки почтового сообщения, передаваемые в соответствии с правилами составления многострочного сообщения.
Описание	Почтовое сообщение.

Ответ:	верх сообщения
Структура	+OK <header1> ... <headern> <line1> ... <linem> . header1 .. headern - строки заголовка почтового сообщения; line1 .. linem - начальные m строк тела почтового сообщения. При передаче строк заголовка и тела почтового сообщения должны использоваться правила составления многострочного сообщения.
Описание	Верхняя часть почтового сообщения

Ответ:	список uid
Структура	+OK <n> <uidm> ; для однострочного ответа n - номер сообщения uidm - уникальный идентификатор сообщения --- +OK ; для многострочного ответа <n1> <uidm1> ... <nn> <uidmn> . n - номер сообщения uidm – уникальный идентификатор сообщения
Описание	Уникальный идентификатор почтового сообщения.

Ответ:	"готов"
Структура	+<строка BASE64>
Описание	Вызов сервера после подачи пользователем команды AUTH

4. СИНТАКСИС КОМАНД И ОТВЕТОВ

В приведенном ниже определении синтаксиса команд и ответов сервера POP3 используется расширенная форма Наура-Бекуса, приведенная в рекомендации RFC 822[2]. Замечание: в качестве разделителя в конструкции "#" используется один пробел и не используются запятые. В случае противоречия в приведенных определениях должно использоваться правило, указанное выше по списку. Различие между строчными и прописными символами алфавита является несущественным.

```
; Команда
<command> ::=     "STAT" <CRLF>
                    | "LIST" [ <SP> <msg> ] <CLRF>
                    | "RETR" <SP> <msg> <CLRF>
                    | "DELE" <SP> <msg> <CLRF>
                    | "NOOP" <CLRF>
                    | "RSET" <CLRF>
                    | "QUIT" <CLRF>
                    | "TOP"  <SP> <msg> <SP> <number> <CLRF>
                    | "UIDL" [ <SP> <msg> ] <CLRF>
                    | "USER" <SP> <name> <CLRF>
                    | "PASS" <SP> <string> <CLRF>
                    | "APOP" <SP> <name> <SP> <digest>
                    | "AUTH" <SP> <str>
; Примечание 1: регистр символов в ключевых словах "STAT", "LIST", ...,
; "AUTH" не имеет значения

; Ответ
<answer>  ::=  <single_line> ; Однострочный ответ
         | <multi_line>  ; Многострочный ответ

<single_line> ::=  <greeting>          ; Приветствие
                    | <status>         ; Статус
                    | <simple_scan_list> ; Однострочный ответ
                                          ; "скан-список"
          | <simple_uid_list>  ; Однострочный ответ
                                          ; "список идентификаторов"
                    | <simple_OK>      ; Простой положительный ответ
                    | <simple_ERR>     ; Простой отрицательный ответ
                    | <ready>          ; Ответ сервера на команду AUTH


<multi_line> ::=  <scan_list>  ; Многострочный скан-список
          | <message>    ; Сообщение
          | <message_top> ; Верхняя часть сообщения
          | <uid_list>   ; Список идентификаторов сообщений



; Примечание 2: длина строк ответа должна составлять не более 512
; символов включая <CRLF>
```

```
<greeting>        ::= "+OK" [ <timestamp> ]

<status>          ::= "+OK" <SP> <number> ; Количество сообщений
                                  : в почтовом ящике
                      <SP> <number> ; Размер почтового ящика
                      <CLRF>

<simple_scan_list>  ::= "+OK" <SP> <number> ; Номер сообщения
                      <SP> <number> ; Размер сообщения
                      <CLRF>

<simple_uid_list>  ::= "+OK" <SP> <number> ; Номер сообщения
                      <SP> <number> ; Идентификатор сообщения
                      <CLRF>

<simple_OK>        ::= "+OK" <text> <CLRF>  ; произвольный текст

<simple_ERR>  ::= "-ERR" <text> <CLRF> ; произвольный текст


<ready>      ::= "+" <SP> <строка BASE64> ; произвольный текст


<scan_list>  ::= "+OK" <CRLF>
            1* ( <number>      ; Номер сообщения
               <SP> <number> ; Размер сообщения
              <CRLF>)
            "." <CRLF>

<message>    ::= "+OK" <CRLF>
            1* (<line> <CRLF>)  ; Строки почтового сообщения
            "." <CRLF>

<message_top> ::= "+OK> <CRLF>
            1* (<line> <CRLF>)  ; Строки заголовка
            <CRLF>
            1* (<line> <CRLF>)  ; Начальные строки сообщения
            "." <CRLF>

<uid_list>   ::= "+OK" <CRLF>
            1* ( <number>      ; Номер сообщения
               <SP> <number>   ; Идентификатор сообщения
               <CRLF> )
            "." <CRLF>

<timestamp>   ::= <метка времени в соответствии с RFC 822[2]>

<msg>    ::=  <number>

<name>   ::=  <string>

<line>   ::=  2*( PRINT_ASCII | <SP> )
```

| (EXCEPT_POINT_ASCII | <SP>)

<text> ::= *(PRINT_ASCII | <SP>)

<string> ::= 1*40 <PRINT_ASCII>

<number> ::= 1*40 <DIGIT>

<CRLF> ::= <CR> <LF>

<CR> ::= символ возврата каретки (код ASCII 13)

<LF> ::= символ следующей строки (код ASCII 10)

<SP> ::= символ пробела (код ASCII - 32)

<DIGIT> ::= <любая цифра ASCII "0".."9">

<PRINT_ASCII> ::= <печатный символ ASCII>

<EXCEPT_POINT> ::= <печатный символ ASCII, кроме символа ".">

<str> : : = <идентификатор в соответствии RFC1731[6]>

5. ТРЕБОВАНИЯ К РЕАЛИЗАЦИИ СЕРВЕРА

5.1. Механизм ограничения объема хранящихся сообщений

В сервере может быть реализован механизм ограничения объема хранящихся сообщений.

5.2. Механизм удаления сообщений, хранящихся дольше установленного срока

В сервере может быть реализован механизм удаления сообщений, хранящихся дольше установленного срока.

ПРИЛОЖЕНИЕ 3

ТЕХНИЧЕСКИЕ ТРЕБОВАНИЯ К ТЕХНИЧЕСКИМ СРЕДСТВАМ СЛУЖБЫ ЭЛЕКТРОННОЙ ПОЧТЫ ПО ПРОТОКОЛУ IMAP4

1. ОБЛАСТЬ ПРИМЕНЕНИЯ

Настоящее приложение описывает технические требования к ТС службы ЭП по протоколу IMAP4 в соответствии с RFC 822 [2], RFC 1733[7] и RFC 2045 [8].

В приложении приведены операции создания, удаления и переименования почтовых ящиков, проверки на наличие новых сообщений, удаления сообщений после прочтения, установки и снятия флагов, разбора сообщений в соответствии со стандартами RFC 822 [2] и RFC 2045 [8], поиска, избирательной выдачи атрибутов и текста сообщений, а также их частей. Доступ к сообщениям организован с использованием либо последовательных номеров сообщений (относительная позиция сообщения в почтовом ящике), либо уникальных идентификаторов (постоянных, последовательно увеличивающихся значений, выделяемых для каждого сообщения).

Не все функции, содержащиеся в данном приложении, обязательны для ТС служб ЭП по протоколу IMAP4, но если они выполняются, то их реализация должна соответствовать настоящему приложению.

2. ФУНКЦИОНАЛЬНЫЕ ТРЕБОВАНИЯ К ВЗАИМОДЕЙСТВИЮ КЛИЕНТА IMAP4 И СЕРВЕРА IMAP4

2.1. Соединения

2.1.1. Протокол нижнего уровня

Протокол IMAP4 должен использовать протокол нижнего уровня, предоставляющий прозрачную надежную доставку потока данных. При использовании протоколом IMAP4 в качестве протокола нижнего уровня TCP, должен использоваться порт 143.

2.1.2. Общая структура протокола

Сессия протокола IMAP4 состоит из установления соединения клиент/сервер, начального приветствия от сервера и взаимодействия клиент/сервер. В ходе сессии сервер и клиент обмениваются сообщениями. Сообщения разделяются на команды клиента, и ответы сервера. В ответы сервера включаются данные, передаваемые сервером клиенту, и результаты выполнения команд. Все сообщения передаются клиентом и сервером в форме строк, заканчивающихся символами <CRLF>.

2.1.3. Соответствие команд и ответов

Команда клиента начинает выполнение операции. В начале каждой команды клиента стоит тег. Для различных команд теги, генерируемые клиентом, должны быть различны.

Возможны ответы сервера, не содержащие тег. Ответ сервера с тегом показывает результат выполнения определенной команды клиента. Значение тега в ответе с тегом должно быть идентично значению тега соответствующей команды клиента.

В ответах без тега вместо тега должен следовать символ "*" или символ "+".

2.2. Состояния

Сервер может находиться в одном из 4-х состояний:

Non-Authenticated ("не идентифицирован");
Authenticated ("идентифицирован");
Selected ("выбран");
Logout ("разъединение").

Для каждого состояния существует свой набор разрешенных команд. На запрещенную команду сервер должен выдавать ответ результата выполнения команды BAD или NO. Диаграмма типовых переходов состояний приведена на рис. 1.

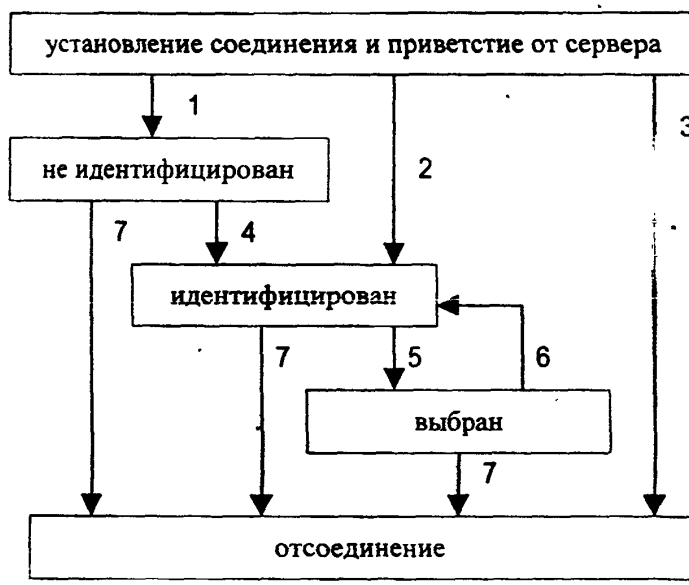


Рисунок 1. Диаграмма типовых переходов состояний

На рис. 1 цифрами обозначены:
1 - соединение без преидентификации (приветствие OK),
2 - соединение с преидентификацией (приветствие PREAUTH),
3 - отвергнутое соединение (приветствие BYE),
4 - успешное выполнение команды LOGIN или AUTHENTICATE,
5 - успешное выполнение команды SELECT или EXAMINE,
6 - команда CLOSE или неудачное выполнение команд SELECT или EXAMINE,
7 - команда LOGOUT, сервер закрыт или соединение закрыто.

2.2.1. Состояние "не идентифицирован"

Сервер входит в состояние "не идентифицирован" после установления соединения, если только соединение не является преидентифицированным.

2.2.2. Состояние "идентифицирован"

В состоянии "идентифицирован" команды работы с сообщениями становятся разрешенными только после того, как клиент выбрал почтовый ящик для доступа. В это состояние сервер входит в случае установления преидентифицированного соединения или после того, как при работе с выбранным почтовым ящиком произошла ошибка.

2.2.3. Состояние "выбран"
Сервер входит в это состояние после того, как почтовый ящик успешно выбран.

2.2.4. Состояние "отсоединено"
В состоянии "отсоединено" идет процесс завершения соединения, после чего сервер закрывает соединение. Сервер может перейти в данное состояние по запросу клиента или по внутреннему одностороннему решению сервера.

2.3. Элементы функционирования сервера

2.3.1. Наименование почтовых ящиков

Имя почтового ящика INBOX является специальным именем, зарезервированным для "первичного почтового ящика данного пользователя на данном сервере".

2.3.2. Иерархическое именование почтовых ящиков

При необходимости экспортировать иерархические имена почтовых ящиков, имена почтовых ящиков должны быть иерархически выстроены слева направо с использованием единственного символа, разделяющего уровни иерархии. Один и тот же разделитель иерархии должен использоваться для всех уровней иерархии внутри одного родительского имени.

2.3.3. Размер почтового ящика и обновления статуса сообщений.

Сервер должен посылать данные о текущем размере почтового ящика автоматически, если наблюдается изменение размера почтового ящика при выполнении команды. Сервер должен посылать данные об изменении флагов сообщений автоматически без запроса клиента на обновление именно этих данных.

2.3.4. Ответы при отсутствии выполняемых в данное время команд.

Сервер может посылать ответы без тегов (кроме EXPUNGE) при отсутствии выполняемых в данное время команд. При реализации подобной возможности в сервере должны быть предусмотрены меры контроля трафика. Должна быть реализована либо проверка, что размер данных не превышает доступный размер окна транспортного протокола нижнего уровня, либо должны использоваться неблокирующие записи.

2.3.5. Таймер автоотсоединения

Если в сервере реализован таймер автоотсоединения, установленное время таймера автоотсоединения должно быть как минимум 30 минут. Получение любой команды от клиента должно сбрасывать таймер автоотсоединения.

2.3.6. Выполнение нескольких команд

От клиента не требуется ждать ответа о результатах выполнения команды перед тем, как отправить следующую команду. Точно так же сервер может не ждать завершения выполнения текущей команды для начала выполнения следующей, кроме тех случаев, когда при одновременном выполнении команд могут возникнуть неоднозначности. В этом случае сервер должен выполнять команды в той последовательности, в которой они получены.

Однако, все ответы на запросы продолжения команды и продолжения команды должны быть согласованы перед инициализацией следующей команды.

2.3.7. Идентификация

Команды AUTHENTICATE и LOGIN запускают процесс идентификации в состоянии Non-authenticated. В сервере может быть реализован неидентифицированный доступ к определенным почтовым ящикам. Для этого должна использоваться команда LOGIN с идентификатором пользователя "anonimus". Наличие пароля является необходимым.

2.3.8. Атрибуты почтового ящика

Должны быть реализованы следующие атрибуты почтового ящика:
Noinferiors - порожденные уровни не существуют и не могут быть созданы;
Noselect - невозможно использовать данное имя в качестве выбираемого почтового ящика;
Marked - почтовый ящик отмечен сервером, как почтовый ящик, в который были добавлены новые сообщения;
Unmarked - после того, как почтовый ящик был выбран последний раз, в него не были добавлены новые сообщения.

3. ПЕРЕЧЕНЬ И СТРУКТУРА СООБЩЕНИЙ ПРОТОКОЛА IMAP4

3.1. Форматы данных, используемые в сообщениях

Протокол IMAP4 использует текстовые команды и ответы. Данные могут быть представлены в одной из 5 форм:

атом;
число;
строка;
список в скобках;
NIL.

Формальное определение форматов данных приведено в Приложении 5.2.

3.1.1. Строки.

Должны быть реализованы два вида строки: литерал и строка в кавычках.

3.1.2. Восьмибитные и бинарные строки.

Восьмибитная текстовая и бинарная почта поддерживается с помощью кодирования MIME-IMB (RFC 2045 [8]). Возможно использование литералов для передачи восьитбитовых или многооктетных символов, но только тогда, когда определен набор символов CHARSET (RFC 1700 [9]).

Несмотря на существование кодирования BINARY для тела сообщения, некодированные бинарные строки запрещены.

Реализации должны кодировать бинарные данные в текстовую форму, такую как BASE64, перед выполнением передачи. Строка, включающая символы CTL, также может считаться бинарной.

3.2. Атрибуты почтовых сообщений

3.2.1. Уникальный номер почтового сообщения (UID), порядковый номер почтового сообщения и идентификатор валидности

Длина уникального идентификатора должна составлять 32 бита.
Длина идентификатора валидности должна составлять 32 бита.
UID должны выделяться в строго возрастающем порядке для вновь поступающих сообщений. Два соседних UID могут отличаться более, чем на 1. UID могут сохраняться неизменными для различных сессий. В случае, если UID не сохраняется для различных сессий, вновь выделяемые UID должны быть больше, чем UID, использованные предыдущими сессиями.

При удалении почтового ящика и создании почтового ящика с таким же именем, идентификатор валидности нового почтового ящика должен быть отличным от предыдущего.

Два соседних порядковых номера сообщения должны отличаться точно на 1. Порядковые номера сообщений могут изменяться в течение сессии.

3.2.2. Флаги почтового сообщения

Должны быть реализованы флаги:

\Seen
\Answered
\Flagged
\Deleted
\Draft
\Recent

Могут быть реализованы дополнительные флаги.

3.3. Команды

Сообщения, направляемые от клиента серверу, называются командами.
Формат команд приведен в п.5.
Список команд с описанием назначения приведен в табл.1.

Таблица 1

Список команд.

Команда	CAPABILITY
Аргументы	-
Описание	Запрашивает список возможностей, поддерживаемых сервером. Сервер должен ответить единственным ответом CAPABILITY без тега с указанием "IMAP4rev1", в качестве одной из возможностей в списке, а затем ответом OK с тегом. Выдаваемый список не должен зависеть от состояния или клиента.
Возможные ответы без тега	Обязательный ответ без тега: CAPABILITY
Возможные ответы с тегом	OK - команда выполнена BAD - неизвестная команда или ошибка в аргументах
Разрешенные состояния	Любое состояние

Команда	NOOP
Аргументы	-
Описание	Нет операции.
Возможные ответы без тега	-
Возможные ответы с тегом	OK - команда выполнена BAD - неизвестная команда или ошибка в аргументах
Разрешенные состояния	Любое состояние

Команда	LOGOUT
Аргументы	-
Описание	Команда информирует сервер, что клиент хочет закрыть соединение.
Возможные ответы без тега	Обязательный ответ без тега: BYE
Возможные ответы с тегом	OK - команда выполнена BAD - неизвестная команда или ошибка в аргументах
Разрешенные состояния	Любое состояние

Команда	AUTHENTICATE <am>
Аргументы	am - название механизма идентификации
Описание	По этой команде сервер начинает процесс идентификации в соответствии с указанным механизмом по RFC 1731 [6]. Сервер может также согласовать механизм дополнительной защиты для последующих взаимодействий протокола. Если механизм идентификации не поддерживается, сервер должен ответить NO и отклонить команду. При успешной идентификации выдается сообщение OK, и сервер переходит в состояние Authenticated.
Возможные ответы без тега	Могут быть запрошены данные продолжения
Возможные ответы с тегом	OK - идентификация выполнена NO - идентификация не выполнена, механизм идентификации не поддерживается BAD - неизвестная команда или ошибка в аргументах
Разрешенные состояния	Not-Authenticated

Команда	LOGIN <user> <pass>
Аргументы	user - идентификатор клиента pass – пароль
Описание	Идентификация клиента и пересылка серверу незакодированного текстового пароля.
Возможные ответы без тега	
Возможные ответы с тегом	OK - идентификация выполнена, состояние authenticated NO - идентификация не выполнена BAD - неизвестная команда или ошибка в аргументах
Разрешенные состояния	Not-Authenticated
Команда	SELECT <mn>
Аргументы	mn - имя почтового ящика
Описание	Команда выбирает почтовый ящик для доступа к его сообщениям. Если клиенту разрешено модифицировать почтовый ящик, сервер может в ответе с тегом OK перед текстом указать код ответа "[READ-WRITE]" Если клиенту не разрешено модифицировать почтовый ящик, сервер должен в ответе с тегом OK перед текстом указать код ответа "[READ-ONLY]"
Возможные ответы без тега	Обязательные ответы без тега: FLAGS, EXISTS, RECENT Необязательные ответы OK без тега: UNSEEN, PERMANENTFLAGS
Возможные ответы с тегом	OK - выбор выполнен, состояние selected NO - ошибка выбора, состояние authenticated BAD – неизвестная команда или ошибка в аргументах
Разрешенные состояния	Authenticated, Selected

Команда	EXAMINE <mn>
Аргументы	mn - имя почтового ящика
Описание	Команда EXAMINE является идентичной команде SELECT. Выбранный почтовый ящик всегда открывается без разрешения модификации (READ-ONLY).
Возможные ответы без тега	Обязательные: FLAGS, EXISTS, RECENT Необязательные ответы OK: UNSEEN, PERMANENTFLAGS
Возможные ответы с тегом	OK - команда выполнена, состояние selected NO - ошибка выбора, состояние authenticated BAD – неизвестная команда или ошибка в аргументах
Разрешенное состояния	Authenticated, Selected

Команда	CREATE <mn>
Аргументы	mn - имя почтового ящика
Описание	Создание почтового ящика с заданным именем. В случае. если клиент намерен создать иерархический почтовый ящик, после имени почтового ящика может следовать символ отделения уровня иерархии. Сервер, для которого не требуется подобный символ, должен его игнорировать. Если символ отделения уровня иерархии появляется внутри имени почтового ящика, сервер может создавать почтовые ящики (несколько почтовых ящиков) таким образом, чтобы их структура соответствовала заданному имени. Например, при попытке создать ящик "foo/bar/zap" (символ "/" является отделителем иерархии) сервер создает иерархические ящики foo/, foo/bar и foo/bar/zap. Если новый почтовый ящик создается с таким же именем, как и ранее удаленный почтовый ящик, сервер должен выделить для нового ящика идентификационный номер больший, чем номера предыдущих ящиков с таким же именем, если только новый ящик не имеет другое уникальное значение валидности идентификатора.
Возможные ответы без тега	-
Возможные ответы с тегом	OK - команда выполнена NO - ошибка создания BAD – неизвестная команда или ошибка в аргументах
Разрешенное состояния	Authenticated, Selected

Команда	DELETE <mn>
Аргументы	mn - имя почтового ящика
Описание	Удаление почтового ящика. Команда не должна удалять ящики низших иерархических имен при удалении ящика высшего иерархического имени. (Удаление foo не должно удалить foo/bar). Является ошибкой попытка удаления ящика высших иерархических имен, клиентом почтового ящика имеющего низшие иерархические имена и атрибут Noselect. Команда может удалять ящики высшего иерархического имени без атрибута /Noselect. При этом все сообщения из данного почтового ящика удаляются и устанавливается атрибут /Noselect.
Возможные ответы без тега	-
Возможные ответы с тегом	OK - команда выполнена NO - ошибка удаления BAD – неизвестная команда или ошибка в аргументах
Разрешенные состояния	Authenticated, Selected

Команда	RENAME <exmn> <newmn>
Аргументы	exmn – существующее имя с почтового ящика newmn - новое имя почтового ящика
Описание	Изменяет имя почтового ящика. Ошибкой является попытка переименования несуществующего почтового ящика или попытка назначения существующего имени. Низшие иерархические имена должны переименовываться. При переименовании почтового ящика INBOX должен быть создан почтовый ящик с новым именем и в него перенесены все сообщения из ящика INBOX, а ящик INBOX должен после выполнения команды остаться пустым. При поддержке сервером низших иерархических имен в ящике INBOX после команды переименования INBOX низшие иерархические имена должны оставаться неизмененными.
Возможные ответы без тега	-
Возможные ответы с тегом	OK - команда выполнена NO - ошибка удаления BAD – неизвестная команда или ошибка в аргументах
Разрешенные состояния	Authenticated, Selected

Команда	SUBSCRIBE <mb>
Аргументы	mb - почтовый ящик
Описание	Добавляет заданное имя почтового ящика к набору активных (подписанных) почтовых ящиков Сервер может выполнять проверку существования почтового ящика перед занесением его в активный список. Сервер не должен в одностороннем порядке удалять имя почтового ящика из списка активных в случае, если ящика с таким именем больше не существует.
Возможные ответы без тега	-
Возможные ответы с тегом	OK - команда выполнена NO - ошибка выполнения команды BAD - неизвестная команда или ошибка в аргументах
Разрешенные состояния	Authenticated, Selected

Команда	UNSUBSCRIBE <mb>
Аргументы	mb - почтовый ящик
Описание	Удаляет заданное имя почтового ящика из списка активных (подписанных) почтовых ящиков.
Возможные ответы без тега	
Возможные ответы с тегом	OK - команда выполнена NO - ошибка выполнения команды BAD - неизвестная команда или ошибка в аргументах
Разрешенные состояния	Authenticated, Selected

Команда	LIST <rn> <mn>
Аргументы	rn - ссылочное имя mn - шаблон имени почтового ящика
Описание	Возвращает подмножество имен полного множества имен, доступных для клиента и удовлетворяющих сочетанию ссылочного имени и шаблона Почтового ящика. Пустое ссылочное имя показывает, что необходимо взять имя почтового ящика, выбранного командой SELECT. Возвращаемые имена почтовых ящиков должны удовлетворять шаблону. При пустом шаблоне имени почтового ящика должно быть возвращено корневое имя от ссылочного име-ни.

Возможные ответы без тега	LIST
Возможные ответы с тегом	OK - команда выполнена NO - ошибка выполнения команды BAD – неизвестная команда или ошибка в аргументах
Разрешенные состояния	Authenticated, Selected

Команда	LSUB <rn> <mn>
Аргументы	rn - ссылочное имя mn - шаблон имени почтового ящика
Описание	Возвращает подмножество имен полного множества имен, занесенных в список активных для клиента и удовлетворяющих сочетанию ссылочного имени и шаблона почтового ящика. Сервер может проверять существование почтовых ящиков, соответствующих возвращаемым именам, и помечать несуществующие ящики флагом \Noselect. Сервер не должен удалять из возвращаемого списка имена, соответствующие несуществующим почтовым ящикам.
Возможные ответы без тега	LSUB
Возможные ответы с тегом	OK - команда выполнена NO - ошибка выполнения команды BAD – неизвестная команда или ошибка в аргументах
Разрешенные состояния	Authenticated, Selected

Команда	STATUS <mn> <data>
Аргументы	mn - имя почтового ящика data - имена пунктов данных статуса
Описание	Команда статуса запрашивает статус указанного почтового ящика. Она не должна влиять на состояние сообщений в указанном ящике (в частности, не должна сбрасывать флаг \Recent). Команда позволяет проверить статус почтового ящика, отличного от выбранного командой SELECT. Имена пунктов данных статуса могут быть: MESSAGES – количество сообщений в почтовом ящике RECENT – количество сообщений с выставленным флагом \Recent UIDNEXT – следующее значение UID, которое будет выделено новому сообщению в данном почтовом ящике. UIDVALIDITY – значение валидности уникального идентификатора для почтового ящика. UNSEEN - количество сообщений, в которых не установлен флаг \Seen .
Возможные ответы без тега	STATUS

Возможные ответы с тегом	OK – команда выполнена NO – ошибка выполнения команды BAD - неизвестная команда или ошибка в аргументах
Разрешенные состояния	Authenticated, Selected

Команда	APPEND <mn> [<flags>] [<date>/<time>] <ml>
Аргументы	mn - имя почтового ящика flags - список в скобках флагов date - строка дата/время ml - литерал сообщения
Описание	Команда добавляет аргумент ml в качестве нового сообщения в конец обозначенного почтового ящика. Аргумент ml должен иметь формат, согласно RFC 822 [2]. В сообщении разрешены восьмибитные символы. Если сервер не может правильно сохранить восьмибитные символы, то в сервере должно быть осуществлено преобразование восьмибитных символов в семибитный код, соответствующий MIME-IMB, а также выполнено и обратное преобразование. Список флагов результирующего сообщения должен соответствовать аргументу flags. По умолчанию список флагов результирующего сообщения должен быть пуст. Если указан аргумент date, внутренняя дата сообщения должна быть установлена согласно указанному значению. По умолчанию устанавливается текущая дата и время. Частичное добавление в случае возникновения ошибки запрещено. При отсутствии указанного почтового ящика сервер не должен автоматически его создавать. В этом случае, а также в случае если нет никаких причин, по которым почтовый ящик с указанным именем не может быть создан, сервер должен в ответ с тегом NO включить код ответа [TRYCREATE].
Возможные ответы без тега	-
Возможные ответы с тегом	OK - команда выполнена NO - ошибка выполнения команды BAD - неизвестная команда или ошибка в аргументах
Разрешенные состояния	Authenticated, Selected

Команда	CHECK
Аргументы	-
Описание	Команда запрашивает контрольную точку текущего почтового ящика. Если команда не реализована, она должна выполняться подобно NOOP.
Возможные ответы без тега	-

Возможные ответы с тегом	OK - команда выполнена BAD - неизвестная команда или ошибка в аргументах
Разрешенные состояния	Selected
Команда	CLOSE
Аргументы	-
Описание	Команда удаляет из текущего выбранного почтового ящика все сообщения, имеющие флаг \Deleted, и переводит сервер в состояние authenticated. Команда игнорируется без выдачи сообщения об ошибке, если почтовый ящик открыт в режиме READ-ONLY.
Возможные ответы без тега	-
Возможные ответы с тегом	OK - команда выполнена, состояние authenticated NO - ошибка закрытия BAD - неизвестная команда или ошибка в аргументах
Разрешенные состояния	Selected

Команда	EXPUNGE
Аргументы	-
Описание	Команда удаляет из текущего выбранного почтового ящика все сообщения, имеющие флаг \Deleted. Перед выдачей ответа OK для каждого удаленного сообщения высылается ответ EXPUNGE.
Возможные ответы без тега	EXPUNGE
Возможные ответы с тегом	OK - команда выполнена NO - ошибка выполнения команды BAD - неизвестная команда или ошибка в аргументах
Разрешенные состояния	Selected

Команда	SEARCH [<CHARSET>] <sc>
Аргументы	CHARSET - спецификация набора символов sc - критерий поиска (несколько критериев)
Описание	Выполняет поиск сообщений, удовлетворяющих критерию поиска. Критерий поиска состоит из одного или более ключей поиска. При наличии нескольких ключей поиска, результат является функцией пересечения (AND) всех сообщений, удовлетворяющих ключам. Ключом поиска является список в скобках ключей поиска. Сервер может исключить из области поиска части тела сообщения с типами содержимого, отличными от типов TEXT или MESSAGE. CHARSET характеризует кодовый набор строк, из которых состоит критерий поиска. Обязательной является поддержка ASCII. Если сервер не поддерживает

	запрошенный кодовый набор, он должен ответить NO с тегом. Во всех ключах поиска, использующих в качестве условия строку, считается, что ключ удовлетворен, если строка ключа является подстрокой поля сообщения. Должно использоваться сравнение, независимое от регистра символов Определены следующие ключи: <message set> - список номеров сообщений ALL - все сообщения в почтовом ящике ANSWERED - сообщения с флагом \Answered BCC <строка> - сообщение, содержащее указанную строку в поле BCC конверта BEFORE <дата> - сообщения, внутренняя дата которых является более ранней, чем указанная. BODY <строка> - сообщение, содержащее указанную строку в теле сообщения CC <строка> - сообщение, содержащее указанную строку в поле BCC конверта DELETED - сообщения с флагом \Deleted DRAFT - сообщения с флагом \Draft FLAGGED - сообщения с флагом \Flagged FROM <строка> - сообщение, содержащее указанную строку в поле From конверта HEADER <имя поля> <строка> - сообщение, указанное поле заголовка которого, содержит указанную строку KEYWORD <флаг> - Сообщения с указанным набором ключевых слов LARGER <n> - Сообщения с размером по RFC-822 [2], большим чем n октет NEW - сообщения с флагом \Recent, но без флага \Seen (= RECENT UNSEEN) NOT <ключ поиска> = сообщения, не удовлетворяющие указанному ключу поиска OLD - сообщения без флага \Recent ON <дата> - сообщения с внутренней датой, входящей в указанный промежуток OR <ключ поиска1> <ключ поиска1> - сообщения, удовлетворяющие любому ключу поиска RECENT - сообщения с флагом \Recent SEEN - сообщения с флагом \Seen SENTBEFORE <дата> - сообщения с датой заголовка более ранней, чем указанная SENTON <дата> - сообщения с датой заголовка, соответствующей указанной SENTSINCE <дата> - сообщения с датой заголовка, более поздней или соответствующей указанной SINCE <дата> - сообщения с внутренней датой, более поздней или соответствующей указанной SMALLER <n> - сообщения с размером по RFC-822 [2], меньшим чем n октет SUBJECT <строка> - сообщение, содержащее указанную строку в поле Subject конверта TEXT <строка> - сообщение, содержащее указанную строку в заголовке или теле сообщения TO <строка> - сообщение, содержащее указанную строку в поле TO конверта

	UID <message set> - сообщение с UID, равным указанному UNANSWERED - сообщения без флага \Answered UNDELETED - сообщения без флага \Deleted UNDRAFT - сообщения без флага \Draft UNFLAGGED - сообщения без флага \Flagged UNKEYWORD <флаг> - сообщения, в которых нет указанных ключевых слов UNSEEN - сообщения без флага \Seen
Возможные ответы без тега	SEARCH
Возможные ответы с тегом	OK - команда выполнена NO - ошибка выполнения команды BAD - неизвестная команда или ошибка в аргументах
Разрешенные состояния	Selected

Команда	FETCH <ms> <items>
Аргументы	ms - набор сообщений items - имена элементов данных сообщеня
Описание	Запрос данных, связанных с сообщением в почтовом ящике. Элементами запрашиваемых данных могут быть единственный атом или список в скобках. Могут быть реализованы элементы: ALL - эквивалентно набору FLAGS INTERNALDATE RFC822.SIZE ENVELOPE BODY - нерасширяемая форма BODYSTRUCTURE BODY [<секция>]<<частичный>> - текст отдельной секции тела сообщения. Секция - набор определителей части. Определитель части – номер части или одно из следующих значений: HEADER, HEADER.FIELDS, HEADER.FIELDS.NOT, MIME, TEXT. Пустой аргумент секции ссылается на сообщение целиком, включая заголовок. Аргумент <частичный> позволяет выбрать произвольный отрезок выбранной части с точностью до октета. Если <частичный> указывает вне текста, возвращается пустая строка. Если указываемый отрезок частично выходит за пределы текста, возвращается накрываемый отрезок текста. При выполнении устанавливается флаг \Seen. BODY.PEEK [<секция>]<<частичный>> - подобно TEXT, но при выполнении флаг \Seen не устанавливается. BODYSTRUCTURE - структура сообщения согласно RFC 2045 [8]. ENVELOPE - Структура конверта сообщения. FAST - эквивалентно набору FLAGS INTERDATE RFC822.SIZE FLAGS - флаги, установленные для данного сообщения FULL - эквивалентно набору FLAGS INTERDATE RFC822.SIZE ENVELOPE BODY INTERNALDATE - внутренняя дата сообщения RFC822 - эквивалентно BODY[], хотя синтаксис возвращаемого ответа отличен RFC822.HEADER - эквивалентно BODY.PEEK[HEADER], хотя синтаксис возвращаемого ответа отличен RFC822.SIZE - размер сообщения по RFC 822 [2]

	RFC822.TEXT - эквивалентно BODY[TEXT], хотя синтаксис возвращаемого ответа отличен UID – UID сообщения
Возможные ответы без тега	FETCH
Возможные ответы с тегом	OK – команда выполнена NO – ошибка выполнения команды BAD - неизвестная команда или ошибка в аргументах
Разрешенные состояния	Selected

Команда	STORE <ms> <md> <value>
Аргументы	ms - набор сообщений md - имя элемента данных почтового сообщения value - значение элемента данных почтового сообщения
Описание	Изменяет данные, связанные с сообщением в почтовом ящике. Обновленные значения данных возвращаются ответом FETCH. Если после имени элемента данных стоит слово .SILENT, ответ без тега FETCH не должен высылаться сервером. Должны быть реализованы аргументы: FLAGS <список флагов> - Удаляет старый список флагов сообщения и создает новый список флагов FLAGS.SILENT <список флагов> +FLAGS <список флагов> - добавляет указанные флаги к существующему списку +FLAGS.SILENT <список флагов> -FLAGS <список флагов> - Удаляет указанные флаги из списка флагов сообщения -FLAGS.SILENT <список флагов>
Возможные ответы без тега	FETCH
Возможные ответы с тегом	OK - команда выполнена NO - ошибка закрытия BAD - неизвестная команда или ошибка в аргументах
Разрешенные состояния	Selected

Команда	COPY <ms> <mn>
Аргументы	Ms - набор сообщений Mn - имя почтового ящика
Описание	Команда копирует определенное сообщение в конец указанного ящика адресата. Флаги и внутренняя дата должны быть сохранены. Если указанного почтового ящика не существует, сервер может создать почтовый ящик автоматически. Если сервер не создал почтовый ящик, но может создать ящик с таким именем, он должен отправить ответ NO с тегом и кодом ответа [TRYCREATE]. В случае неудачного выполнения команды, сервер должен вернуть почтовый ящик в состояние, идентичное тому, что было до выполнения команды.

Возможные ответы без тега	-
Возможные ответы с тегом	OK - команда выполнена NO - ошибка закрытия BAD - неизвестная команда или ошибка в аргументах
Разрешенные состояния	Selected

Команда	UID <cn> <ca>
Аргументы	cn - имя команды ca - аргументы команды
Описание	Команда UID имеет две формы. В первой форме первым аргументом является команда COPY, FETCH или STORE, а вторым аргументом - аргументы указанной команды. Однако в аргументе набора сообщений нужно указывать не последовательные номера сообщений, а UID. Во второй форме команда UID отрабатывает команду SEARCH с соответствующими аргументами. Аргументы интерпретируются так же, как и в команде SEARCH, однако номера, возвращаемые в ответе SEARCH, будут UID вместо порядковых номеров сообщений.
Возможные ответы без тега	FETCH, SEARCH
Возможные ответы с тегом	OK - команда выполнена NO - ошибка закрытия BAD – неизвестная команда или ошибка в аргументах
Разрешенные состояния	Selected

3.4. Ответы

Сообщения, направляемые от сервера клиенту, называются ответами.
Существуют три формы ответов сервера:

ответы статуса
данные сервера
запрос продолжения команды

Ответы статуса могут быть:

с тегом
без тега

Ответы с тегом показывают результат выполнения команды клиента: статус OK, NO, BAD.

В ответе без тега на месте тега находится символ "*". Ответами статуса без тега могут быть ответ приветствия или ответы статуса, не показывающие результаты выполнения какой-либо команды.

В запросах продолжения команды вместо тега находится символ "+". Эти запросы посылаются клиенту, чтобы показать, что неполная команда принята, и сервер готов принять остаток команды.

3.4.1. Ответы статуса

Ответами статуса являются: OK, BAD, NO, PREAUTH, BYE.
Ответы OK, BAD, NO могут быть с тегом и без тега.
Ответы PREAUTH и BYE всегда с тегом.
Ответы статуса могут включать код ответа. Код ответа состоит из данных внутри квадратных скобок в форме атома. Возможны аргументы, отделенные пробелом. Список кодов ответа приведен в табл. 2. Все дополнительные коды ответов, реализованные в сервере, должны начинаться с буквы X.

Таблица 2

Список кодов ответа.

Код ответа:	Описание
ALERT	Текст, предназначенный для чтения человеком-клиентом, составленный таким образом, чтобы привлечь внимание клиента.
NEWNAME	Ошибка выполнения команд SELECT или EXAMINE в связи с тем, что почтовый ящик с указанным именем больше не существует, так как был переименован.
PARSE	Текст, предназначенный для чтения человеком-клиентом и показывающий ошибку разбора заголовка RFC822 [2] или заголовка MIME-IMB сообщения.
PERMANENTFLAGS	За данным кодом следует список в скобках, содержащий список флагов, которые клиент может изменить. Список PERMANENTFLAGS может включать в себя специальный флаг *, говорящий о возможности создания новых ключевых слов путем попыток сохранить данные флаги в почтовом ящике.
READ-ONLY	Почтовый ящик выбран в режиме только для чтения.
READ-WRITE	Почтовый ящик выбран как доступный для изменений.
TRYCREATE	Ошибка выполнения команд APPEND или COPY в связи с тем, что указанный почтовый ящик назначения не существует.
UIDVALIDITY	За данным кодом следует десятичное число, указывающее значение валидности UID.
UNSEEN	За данным кодом следует десятичное число, показывающее номер первого сообщения без флага \Seen.

В табл. 3 приведен список ответов статуса. Все сообщения статуса должны содержать текст, предназначенный для чтения человеком-клиентом и содержащий информацию, соответствующую назначению сообщения.

Таблица 3

Список ответов статуса.

Ответ:	Описание
OK	Указывает на информационное сообщение, поступившее от сервера. Если присутствует тег, ответ указывает на успешное выполнение соответствующей команды. Ответ без тега может использоваться в качестве приветствия.

NO	Указывает на сообщение об ошибке работы сервера. Если присутствует тег, ответ указывает на ошибку выполнения соответствующей команды. Сообщение без тега является предупреждением и не указывает на то, что команда не выполнена.
BAD	Указывает на сообщение ошибки, поступившее от сервера. . Если присутствует тег, ответ указывает на ошибку протокольного уровня в команде клиента, на которую указывает тег.
PREAUTH	Может использоваться в качестве приветствия. Указывает на установление преидентифицированного соединения.
BYE	Указывает на процесс закрытия соединения. Посылается сервером в случаях: - нормального закрытия соединения по команде LOGOUT - в случае внезапного закрытия соединения сервером (например, в случае останова сервера) - в случае разрыва соединения по таймеру неактивности - при установлении соединения в качестве приветствия, показывающего, что сервер не желает.установить соединение в клиентом.

3.4.2. Ответы статуса сервера и почтового ящика

С помощью данных ответов сервер передает данные клиенту. В табл. 4 приведен список ответов статуса сервера и почтового ящика

Таблица 4

Список ответов статуса сервера и почтового ящика.

Ответ:	Описание
CAPABILITY	Ответ на команду CAPABILITY. Выдается перечень возможностей, поддерживаемых сервером. Должен быть включен атом "IMAP4rev1". Возможность "AUTH=" показывает, что сервер поддерживает данный механизм идентификации. Имена, данные в списке ответа, должны быть либо стандартными, либо начинаться с буквы X.
LIST	Ответ на команду LIST. Выдается одно имя, удовлетворяющее спецификации команды LIST. Может быть несколько ответов LIST на команду LIST. В ответе LIST указывается разделитель иерархии. Если почтовый ящик неиерархический, в качестве разделителя иерархии должен стоять NUL.
LSUB	Ответ на команду LSUB. Выдается одно имя, удовлетворяющее спецификации команды. Может быть несколько ответов на команду.
STATUS	Ответ на команду STATUS. Возвращает имя почтового ящика, удовлетворяющее спецификации команды, и запрошенную информацию статуса.
SEARCH	Ответ на команду SEARCH. Возвращает список номеров сообщений, удовлетворяющих критерию поиска. Для команды SEARCH выдаются порядковые номера, для команды UID SEARCH - UID.
FLAGS	Ответ на команду SELECT или EXAMINE. Выдает список в скобках, содержащий флаги, которые можно использовать в данном почтовом ящике.

3.4.3. Ответы размера почтового ящика

С помощью данных ответов сервер передает клиенту информацию об изменениях размера почтового ящика. В табл.5 приведен список ответов размера почтового ящика.

Таблица 5

Список ответов размера почтового ящика.

Ответ:	Описание
EXISTS	Указывает количество сообщений в почтовом ящике. Выдается в ответ на команды SELECT или EXAMINE, или когда меняется размер почтового ящика.
RECENT	Указывает количество сообщений в почтовом ящике с установленным флагом \Recent. Выдается в ответ на команды SELECT или EXAMINE, или когда меняется размер почтового ящика.

3.4.4. Ответы статуса сообщений

С помощью данных ответов сервер передает клиенту данные сообщений. В табл. 6 приведен список ответов статуса сообщений. Сразу за символом "*" находится последовательный номер сообщения.

3.4.5. Ответ запроса продолжения команды

Данный ответ показывает, что сервер готов принять продолжение команды от клиента. Данный ответ используется в команде AUTHORIZATION для выполнения передачи данных сервера клиенту и запроса дополнительных данных клиента. Данный ответ может также использоваться для любой команды в случае, если аргументом является литерал. Клиент не должен высылать октеты текста литерала, пока сервер не выдаст запрос. Остаток команды, включая <CRLF>, следует за октетами литерала. В случае, если присутствуют дополнительные аргументы, за литералом должен следовать пробел и эти аргументы.

Таблица 6

Список ответов статуса сообщений.

Ответ:	Описание
EXPUNGE	Указывает, что определенный последовательный номер сообщения был удален почтового ящика. Последовательные номера сообщений в данном почтовом ящике немедленно уменьшаются на 1. В случае, если удаляются несколько сообщений, выдается соответствующее количество ответов EXPUNGE. Корректировка номеров оставшихся сообщений производится после каждого ответа EXPUNGE. Таким образом, номера, выдаваемые ответами EXPUNGE, п удалении нескольких сообщений не будут соответствовать номерам, указанным команде EXPUNGE. Ответ EXPUNGE не должен посылаться во время выполнения команд FETCH, STORE, SEARCH.

FETCH	Выдает клиенту данные о сообщении. Данные передаются в виде пар: имя элемента данных и его значение. Данный ответ может быть послан при выполнении команд FETCH или STORE или без команды. Определены элементы данных: BODY – форма BODYSTRUCTURE без данных расширения BODY [<секция>]<<начальный октет>> - текст отдельной секции тела сообщения, начиная с начального октета. Данные могут быть в 8-битовом формате, если в список параметров данной секции включен идентификатор [CHARSET]. 7-битные и 8-битные символы запрещены в заголовках. Нетекстовые данные должны передаваться в форме BASE64. BODYSTRUCTURE - список в скобках, описывающий структуру сообщения согласно MIME-IMB. Порядок описания данных: Данные расширения части сложного тела: Body parameter parenthesized list Body discription Body language Первичные поля части простого тела: Body type Body subtype Body patameter parenthesized list Body id Body description Body encoding Body size Данные расширения части простого тела: Body MD5 Body disposition Body language ENVELOPE - список в скобках, описывающий структуру конверта сообщения. Порядок полей должен быть следующим: date, subject, from, sender, reply-to, to, cc, bcc, in-reply-to, message-id. Порядок полей списка адреса: personal name, at-domain-list, mailbox name, host name. Все поля с отсутствующими значениями должны иметь значения NIL. Сервер должен определить поля reply-to и sender из поля from. FLAGS – список в скобках флагов, установленных для данного сообщения INTERNALDATE - строка, представляющая внутреннюю дату сообщения RFC822 – эквивалентно BODY[] RFC822.HEADER - эквивалентно BODY.PEEK[HEADER] RFC822.SIZE – числовое выражение, показывающее размер сообщения по RFC 822 [2] RFC822.TEXT – эквивалентно BODY[TEXT] UID – числовое выражение, показывающее UID сообщения

4.СИНТАКСИС КОМАНД И ОТВЕТОВ

В приведенном ниже определении синтаксиса команд и ответов сервера IMAP4 используется расширенная форма Наура-Бекуса, приведенная в рекомендации RFC 822 [2].

Замечание: в качестве разделителя в конструкции "#" используется один пробел и не используются запятые. В случае противоречия в приведенных определениях должно использоваться правило, указанное выше по списку. Различие между строчными и прописными символами алфавита является несущественным.

```
<">          ::= <ASCII quote mark, 0x22>
                     ;; метка ограничителя ASCII, код 0x22

address      ::= "(" addr_name SPACE addr_adl SPACE addr_mailbox
             SPACE addr_host ")"

addr_adl     ::= nstring
             ;; Holds route from [RFC-822[2]] route-addr if
             ;; non-NIL
                     ;; (Содержит маршрутизацию от [RFC-822[2]] адреса
                     маршрутизации,
                     ;; если не-NIL)
addr_host    ::= nstring
             ;; NIL indicates [RFC-822[2]] group syntax.
             ;; Otherwise, holds [RFC-822[2]] domain name
                     ;; (NIL указывает [RFC-822[2]] синтаксис группы.
                     ;; Иначе, содержит имя домена по [RFC-822[2]])
addr_mailbox ::= nstring
             ;; NIL indicates end of [RFC-822[2]] group; if
             ;; non-NIL and addr_host is NIL, holds
             ;; [RFC-822[2]] group name.
             ;; Otherwise, holds [RFC-822[2]] local-part
                     ;; (NIL указывает на конец групы по [RFC-822[2]];
                     ;; Если не-NIL и addr_host - NIL, то содержит имя группы
                     ;; по [RFC-822[2]]. Иначе, содержит локальную часть.)
addr_name    ::= nstring
             ;; Holds phrase from [RFC-822[2]] mailbox if
             ;; non-NIL
                     ;; (Содержит фразу из [RFC-822[2]] почтового ящика, если не-NIL)
alpha        ::= "A" / "B" / "C" / "D" / "E" / "F" / "G" / "H" /
             "I" / "J" / "K" / "L" / "M" / "N" / "O" / "P" /
             "Q" / "R" / "S" / "T" / "U" / "V" / "W" / "X" /
             "Y" / "Z" /
             "a" / "b" / "c" / "d" / "e" / "f" / "g" / "h" /
             "i" / "j" / "k" / "l" / "m" / "n" / "o" / "p" /
             "q" / "r" / "s" / "t" / "u" / "v" / "w" / "x" /
             "y" / "z"
             ;; Case-sensitive (Есть зависимость от регистра букв)
             ;; (65. - 90.), (97. - 122.)

append       ::= "APPEND" SPACE mailbox [SPACE flag_list]
             [SPACE date_time] SPACE literal

astring      ::= atom / string

atom         ::= 1*ATOM_CHAR
```

```
ATOM_CHAR       ::= <any CHAR except atom_specials>
                    ;; любое CHAR, кроме atom_specials

atom_specials  ::= "("  ")"  "{" / SPACE / CTL / list_wildcards
            quoted_specials

authenticate   ::= "AUTHENTICATE" SPACE auth_type *(CRLF base64)

auth_type      ::= atom
                    ;; Defined by [IMAP-AUTH] (Определяется документом [IMAP-
AUTH])

base64         ::= *(4base64_char) [base64_terminal]

base64_char    ::= alpha / digit / "+" / "/"

base64_terminal ::= (2base64_char "==") / (3base64_char "=")

body           ::= "(" body_type_1part / body_type_mpart ")"

body_extension  ::= nstring / number / "(" 1#body_extension ")"
            ;; Future expansion.  Client implementations
            ;; MUST accept body_extension fields.  Server
            ;; implementations MUST NOT generate
            ;; body_extension fields except as defined by
            ;; future standard or standards-track
            ;; revisions of this specification.
                    ;; (Будущее перспективное расширение. Клиентские реализации
                    ;; должны иметь доступ к полям body_extension. Реализации
                    ;; сервера не должны осуществлять
                    ;; генерацию полей body_extension, за
                    ;; исключением случаев, определенных вводимыми стандартами
                    ;; или поправками - версиями ревизий - данной спецификации)

body_ext_1part  ::= body_fld_md5 [SPACE body_fld_dsp
            [SPACE body_fld_lang
            [SPACE 1#body_extension]]]
            ;; MUST NOT be returned on non-extensible
            ;; "BODY" fetch
                    ;; (не должны возвращаться на нерасширенный вызов BODY )

body_ext_mpart  ::= body_fld_param
            [SPACE body_fld_dsp SPACE body_fld_lang
            [SPACE 1#body_extension]]
            ;; MUST NOT be returned on non-extensible
            ;; "BODY" fetch
                    ;; (не должны возвращаться на нерасширенный вызов BODY )

body_fields    ::= body_fld_param SPACE body_fld_id SPACE
            body_fld_desc SPACE body_fld_enc SPACE
            body_fld_octets
```

```
body_fld_desc   ::= nstring

body_fld_dsp    ::= "(" string SPACE body_fld_param ")" / nil

body_fld_enc    ::= (<"> ("7BIT" / "8BIT" / "BINARY" / "BASE64"/
                "QUOTED-PRINTABLE") <">) / string

body_fld_id     ::= nstring

body_fld_lang   ::= nstring  "(" 1#string ")"

body_fld_lines  ::= number

body_fld_md5    ::= nstring

body_fld_octets ::= number

body_fld_param  ::= "(" 1#(string SPACE string) ")" / nil

body_type_1part ::= (body_type_basic / body_type_msg / body_type_text)
                [SPACE body_ext_1part]

body_type_basic ::= media_basic SPACE body_fields
                ;; MESSAGE subtype MUST NOT be "RFC822"
                        ;; (Под-тип сообщения не должен определяться RFC822[2])

body_type_mpart ::= 1*body SPACE media_subtype
                [SPACE body_ext_mpart]

body_type_msg   ::= media_message SPACE body_fields SPACE envelope
                SPACE body SPACE body_fld_lines

body_type_text  ::= media_text SPACE body_fields SPACE body_fld_lines

capability      ::= "AUTH=" auth_type / atom
                ;; New capabilities MUST begin with "X" or be
                ;; registered with IANA as standard or
                ;; standards-track
                        ;; (Новые возможности не должны начинаться с "X" или
                        ;; регистрироваться с именем IANA в качестве
                        ;; стандартного или стандартной записью)

capability_data ::= "CAPABILITY" SPACE [1#capability SPACE]
                "IMAP4rev1" [SPACE 1#capability]
                ;; IMAP4rev1 servers which offer RFC1730[10], RFC2060[11]
                ;; compatibility MUST list "IMAP4" as the first
                ;; capability.
                        ;; IMAP4revq серверы, которые предлагают
                        ;; совместимость по RFC1730[10] и RFC2060[11], должны определить
                        ;; список "IMAP4" в качестве первоначально-определяемой
                        ;; совместимости
```

```
CHAR            ::= <any 7-bit US-ASCII character except NUL,
                0x01 - 0x7f>

CHAR8           ::= <any 8-bit octet except NUL, 0x01 - 0xff>

command         ::= tag SPACE (command_any / command_auth /
                command_nonauth / command_select) CRLF
                ;; Modal based on state (Форма базируется на утверждении)

command_any     ::= "CAPABILITY" / "LOGOUT" / "NOOP" / x_command
                ;; Valid in all states (Годится в любых состояниях)

command_auth    ::= append / create / delete / examine / list /
                lsub / rename / select / status / subscribe /
                unsubscribe
                ;; Valid only in Authenticated or Selected state
                        ;; (Годится только в режиме Аутоидентификации и Выбора)

command_nonauth ::= login / authenticate
                ;; Valid only when in Non-Authenticated state
                        ;; (Годится только в режиме Не-Аутоидентификации)

command_select  ::= "CHECK" / "CLOSE" / "EXPUNGE" /
                copy / fetch / store / uid / search
                ;; Valid only when in Selected state
                        ;; (Годится только в режиме Выбора)

continue_req    ::= "+" SPACE (resp_text / base64)

copy            ::= "COPY" SPACE set SPACE mailbox

CR              ::= <ASCII CR, carriage return, 0x0D>

create          ::= "CREATE" SPACE mailbox
                ;; Use of INBOX gives a NO error
                        ;; (Использование INBOX дает ошибку NO)

CRLF            ::= CR LF

CTL             ::= <any ASCII control character and DEL,
                  0x00 - 0x1f, 0x7f>

date            ::= date_text / <"> date_text <">

date_day        ::= 1*2digit
                ;; Day of month (День месяца)

date_day_fixed  ::= (SPACE digit) / 2digit
                ;; Fixed-format version of date_day
                        ;; (Версия формата даты/дня)

date_month      ::= "Jan" / "Feb" / "Mar" / "Apr" / "May" / "Jun" /
```

```
                "Jul" / "Aug"  "Sep" / "Oct" / "Nov" / "Dec"

date_text       ::= date_day "-" date_month "-" date_year

date_year       ::= 4digit

date_time       ::= <"> date_day_fixed "-" date_month "-" date_year
                SPACE time SPACE zone <">


delete          ::= "DELETE" SPACE mailbox
                ;; Use of INBOX gives a NO error
                        ;; (Использование INBOX дает ошибку NO)

digit           ::= "0" / digit_nz

digit_nz        ::= "1" / "2" / "3" / "4" / "5" / "6" / "7" / "8" /
                "9"
                ;; (48. - 57.)

envelope        ::= "(" env_date SPACE env_subject SPACE env_from
                SPACE env_sender SPACE env_reply_to SPACE env_to
                SPACE env_cc SPACE env_bcc SPACE env_in_reply_to
                SPACE env_message_id ")"

env_bcc         ::= "(" 1*address ")" / nil

env_cc          ::= "(" 1*address ")" / nil

env_date        ::= nstring

env_from        ::= "(" 1*address ")" / nil

env_in_reply_to ::= nstring

env_message_id  ::= nstring

env_reply_to    ::= "(" 1*address ")" / nil

env_sender      ::= "(" 1*address ")" / nil

env_subject     ::= nstring

env_to          ::= "(" 1*address ")" / nil

examine         ::= "EXAMINE" SPACE mailbox

fetch           ::= "FETCH" SPACE set SPACE ("ALL" / "FULL" /
                "FAST" / fetch_att / "(" 1=fetch_att ")")

fetch_att       ::= "ENVELOPE" / "FLAGS" / "INTERNALDATE" /
                "RFC822" [".HEADER" / ".SIZE" / ".TEXT"] /
```

```
            "BODY" ["STRUCTURE"] / "UID" /
            "BODY" [".PEEK"] section
            ["<" number "." nz_number ">"]

flag        ::= "\Answered" / "\Flagged" / "\Deleted" /
            "\Seen" / "\Draft" / flag_keyword / flag_extension

flag_extension  ::= "\" atom
            ;; Future expansion.  Client implementations
            ;; MUST accept flag_extension flags.  Server
            ;; implementations MUST NOT generate
            ;; flag_extension flags except as defined by
            ;; future standard or standards-track
            ;; revisions of this specification.
                        ;; (Будущее перспективное расширение. Клиентские реализации
                        ;; должны иметь доступ к флагам flag_extension. Реализации
                        ;; сервера не должны осуществлять
                        ;; генерацию флагов flag_extension, за
                        ;; исключением случаев, определенных вводимыми стандартами
                        ;; или поправками - версиями ревизий - данной спецификации)

flag_keyword    ::= atom

flag_list       ::= "(" #flag ")"

greeting        ::= "*" SPACE (resp_cond_auth / resp_cond_bye) CRLF

header_fld_name ::= astring

header_list     ::= "(" 1#header_fld_name ")"

HTAB            ::= <ASCII HT, horizontal tab, 0x9>

LF              ::= <ASCII LF, line feed, 0x0A>

linear-white-space ::= 1*([CRLF] LWSP-char)

list            ::= "LIST" SPACE mailbox SPACE list_mailbox

list_mailbox    ::= 1*(ATOM_CHAR / list_wildcards) / string

list_wildcards  ::= "%" / "*"

literal         ::= "{" number "}" CRLF *CHAR8
            ;; Number represents the number of CHAR8 octets
                        ;; (Number представляет число октетов CHAR8)

login           ::= "LOGIN" SPACE userid SPACE password

lsub            ::= "LSUB" SPACE mailbox SPACE list_mailbox

LWSP-char       ::= SPACE / HTAB
```

```
mailbox         ::= "INBOX" astring
                ;; INBOX is case-insensitive.  All case variants of
                ;; INBOX (e g  "iNbOx") MUST be interpreted as INBOX
                ;; not as an astring.  Refer to section 5.1 for
                ;; further semantic details of mailbox names.
                        :: (INBOX зависит от ргистра букв. Все варианты случаев
                        :: INBOX (напрмер "iNbOx") должны быть интерпретированы
                        ;; и не astring. См. раздел 5.1 для
                        ;; более подробного рассмотрения имен почтовых яиков.)

mailbox_data    ::= "FLAGS" SPACE flag_list /
                "LIST" SPACE mailbox_list /
                "LSUB" SPACE mailbox_list /
                "MAILBOX" SPACE text /
                "SEARCH" [SPACE 1#nz_number] /
                "STATUS" SPACE mailbox SPACE
                "(" #<status_att number ")" /
                number SPACE "EXISTS" / number SPACE "RECENT"

mailbox_list    ::= "(" #("\Marked" / "\Noinferiors" /
                "\Noselect" / "\Unmarked" / flag_extension) ")"
                SPACE (<"> QUOTED_CHAR <"> / nil) SPACE mailbox

media_basic     ::= (<"> ("APPLICATION" / "AUDIO" / "IMAGE" /
                "MESSAGE" / "VIDEO") <">) / string)
                SPACE media_subtype
                ;; Defined in [MIME-IMT] (Определены в [MIME-IMT])

media_message   ::= <"> "MESSAGE" <"> SPACE <"> "RFC822" <">
                ;; Defined in [MIME-IMT] (Определены в [MIME-IMT])

media_subtype   ::= string
                ;; Defined in [MIME-IMT] (Определены в [MIME-IMT])


media_text      ::= <"> "TEXT" <"> SPACE media_subtype
                ;; Defined in [MIME-IMT] (Определены в [MIME-IMT])

message_data    ::= nz_number SPACE ("EXPUNGE" /
                        ("FETCH" SPACE msg_att))

msg_att         ::= "(" 1#("ENVELOPE" SPACE envelope /
                "FLAGS" SPACE "(" #(flag / "\Recent") ")" /
                "INTERNALDATE" SPACE date_time /
                "RFC822" [".HEADER" / ".TEXT"] SPACE nstring /
                "RFC822.SIZE" SPACE number /
                "BODY" ["STRUCTURE"] SPACE body /
                "BODY" section ["<" number ">"] SPACE nstring /
                "UID" SPACE uniqueid) ")"

nil             ::= "NIL"
```

```
nstring         ::= string  nil

number          ::= 1*digit
                ;; Unsigned 32-bit integer (Целые без знака.32 бита)
                ;; (0 <= n < 4,294,967,296)

nz_number       ::= digit_nz *digit
                ;; Non-zero unsigned 32-bit integer (Ненулевые целые без
                        ;; знака, 32 бита)

                ;; (0 < n < 4,294,967,296)

password        ::= astring

quoted          ::= <"> *QUOTED_CHAR <">

QUOTED_CHAR     ::= <any TEXT_CHAR except quoted_specials> /
                "\" quoted_specials

quoted_specials ::= <"> / "\"

rename          ::= "RENAME" SPACE mailbox SPACE mailbox
                ;; Use of INBOX as a destination gives a NO error
                        ;; (Использование INBOX, выдающего сообщение об отсутствии
                        ;; ошибок как пункта назначения)

response        ::= *(continue_req / response_data) response_done

response_data   ::= "*" SPACE (resp_cond_state / resp_cond_bye /
                mailbox_data / message_data / capability_data)
                CRLF

response_done   ::= response_tagged / response_fatal

response_fatal  ::= "*" SPACE resp_cond_bye CRLF
                ;; Server closes connection immediately
                        ;; (Сервер немедленно закрывает соединение)

response_tagged ::= tag SPACE resp_cond_state CRLF

resp_cond_auth  ::= ("OK" / "PREAUTH") SPACE resp_text
                ;; Authentication condition (Условие аутоидентификации)

resp_cond_bye   ::= "BYE" SPACE resp_text

resp_cond_state ::= ("OK" / "NO" / "BAD") SPACE resp_text
                ;; Status condition (Условие статуса)

resp_text       ::= ["[" resp_text_code "]" SPACE] (text_mime2 / text)
                ;; text SHOULD NOT begin with "[" or "="
                        ;; (текст не следует начинать с символа "[" или "=")
```

```
resp_text_code  ::= "ALERT" / "PARSE" /
            "PERMANENTFLAGS" SPACE "(" #(flag / "\*") ")" /
            "READ-ONLY" / "READ-WRITE" / "TRYCREATE" /
            "UIDVALIDITY" SPACE nz_number /
            "UNSEEN" SPACE nz_number /
            atom [SPACE 1*<any TEXT_CHAR except "]">]

search      ::= "SEARCH" SPACE ["CHARSET" SPACE astring SPACE]
            1#search_key
            ;; [CHARSET] MUST be registered with IANA
                        ;; ( [CHARSET] должен быть зарегистрирован с именем IANA)

search_key    ::= "ALL" / "ANSWERED" / "BCC" SPACE astring /
            "BEFORE" SPACE date / "BODY" SPACE astring /
            "CC" SPACE astring / "DELETED" / "FLAGGED" /
            "FROM" SPACE astring /
            "KEYWORD" SPACE flag_keyword / "NEW" / "OLD" /
            "ON" SPACE date / "RECENT" / "SEEN" /
            "SINCE" SPACE date / "SUBJECT" SPACE astring /
            "TEXT" SPACE astring / "TO" SPACE astring /
            "UNANSWERED" / "UNDELETED" / "UNFLAGGED" /
            "UNKEYWORD" SPACE flag_keyword / "UNSEEN" /
            ;; Above this line were in [IMAP2]
                        ;; (Ранее эта строка была в [IMAP2])
            "DRAFT" /
            "HEADER" SPACE header_fld_name SPACE astring /
            "LARGER" SPACE number / "NOT" SPACE search_key /
            "OR" SPACE search_key SPACE search_key /
            "SENTBEFORE" SPACE date / "SENTON" SPACE date /
            "SENTSINCE" SPACE date / "SMALLER" SPACE number /
            "UID" SPACE set / "UNDRAFT" / set /
           ."(" 1#search_key ")"

section     ::= "[" [section_text / (nz_number *["." nz_number]
            ["." (section_text / "MIME")])] "]"

section_text  ::= "HEADER" / "HEADER.FIELDS" [".NOT"]
            SPACE header_list / "TEXT"

select      ::= "SELECT" SPACE mailbox

sequence_num  ::= nz_number / "*"
            ;; * is the largest number in use.  For message
            ;; sequence numbers, it is the number of messages
            ;; in the mailbox.  For unique identifiers, it is
            ;; the unique identifier of the last message in
            ;; the mailbox.
                        ;; (* -наибольше используемое число. Для последовательного
                        ;; подсчета сообщений - это число сообщений в почтовом ящике.
                        ;; Для уникального идентификатора - это уникальный иденти-
                        ;; фикатор последнего сообщения в почтовом ящике)
```

```
set             ::= sequence_num  (sequence_num ":" sequence_num) /
                    (set "," set)
                    ;; Identifies a set of messages.  For message
                    ;; sequence numbers, these are consecutive
                    ;; numbers from 1 to the number of messages in
                    ;; the mailbox
                    ;; Comma delimits individual numbers, colon
                    ;; delimits between two numbers inclusive.
                    ;; Example: 2,4:7,9,12:* is 2,4,5,6,7,9,12,13,
                    ;; 14,15 for a mailbox with 15 messages.
                                :: (Определяет набор сообщений. Для подсчета последовательных
                                :: сообщений это - последовательные числа от 1 до числа
                                :: сообщений в почтовом ящике. Запятая разделяет отдельные
                                :: числа, двоеточие указывает наличие промежуточных чисел.
                                :: Например, 2,4:7,9,12:* это 2,4,5,6,7,9,12,13,14,15 для
                                :: почтового ящика с 15-ью сообщениями)

SPACE           ::= <ASCII SP, space, 0x20>

status          ::= "STATUS" SPACE mailbox SPACE "(" 1#status_att ")"

status_att      ::= "MESSAGES" / "RECENT" / "UIDNEXT" /
                    "UIDVALIDITY" / "UNSEEN"

store           ::= "STORE" SPACE set SPACE store_att_flags

store_att_flags ::= (["+" / "-"] "FLAGS" [".SILENT"]) SPACE
                    (flag_list / #flag)

string          ::= quoted / literal

subscribe       ::= "SUBSCRIBE" SPACE mailbox

tag             ::= 1*<any ATOM_CHAR except "+">

text            ::= 1*TEXT_CHAR

text_mime2      ::= "=?" <charset> "?" <encoding> "?"
                    <encoded-text> "?="
                    ;; Syntax defined in [MIME-HDRS]
                                ;;(Синтаксис определен в [MIME-IMT])


TEXT_CHAR       ::= <any CHAR except CR and LF>

time            ::= 2digit ":" 2digit ":" 2digit
                    ;; Hours minutes seconds (Часы минуты секунды)

uid             ::= "UID" SPACE (copy / fetch / search / store)
                    ;; Unique identifiers used instead of message
                    ;; sequence numbers
```

;; (Уникальные идентификаторы, используемые вместо
;; номера последовательных сообщений)

uniqueid ::= nz_number
;; Strictly ascending (Строго по восходящей)

unsubscribe ::= "UNSUBSCRIBE" SPACE mailbox

usend ::= astring

x_command ::= "X" atom <experimental command arguments>

zone ::= ("+" / "-") 4digit
;; Signed four-digit value of hhmm representing
;; hours and minutes west of Greenwich (that is,
;; (the amount that the given time differs from
;; Universal Time). Subtracting the timezone
;; from the given time will give the UT form.
;; The Universal Time zone is "+0000".
;; (4-х значное величина hhmm со знаком, отображающая
;; часы и минуты по Гринвичу, то есть значение, которое
;; дает отличие времени от значения Универсального
;; времени). Вычитание временной зоны из данного времени
;; даст значение UT-формы - формы Универсального времени.
;; Зона Универсального времени - это "+0000".)

5. СТРУКТУРА ПОЧТОВОГО СООБЩЕНИЯ В СООТВЕТСТВИИ С RFC [illegible]

Сообщение состоит из заголовка и необязательного тела сообщения. Тело сообщения является простым набором линий, содержащих символы ASCII. Тело сообщения отделено от заголовка NULL-строкой (строкой, состоящей только из символов <CRLF>). Заголовок сообщения состоит из полей заголовка. Каждое поле состоит из имени поля, двоеточия - символа ":" и значения поля. Значение поля может быть разбито на части символами <CRLF>. При этом символы <CRLF> должны немедленно следовать за символом <LWSP>. Далее поля заголовка описываются в предположении, что из их значений выброшены символы <CRLF>.

Приведено определение структуры сообщения с использованием формы Наура-Бекуса. В определениях используются элементы, определенные в п. 5.

<"> = <ASCII quote mark> ; (42, 34.)

addr-spec = local-part "@" domain ; глобальный адрес

address = mailbox ; один адресат
/ group ; именованный список

ALPHA = <любой алфавитный символ ASCII>
; (101-132, 65.- 90.)
; (141-172, 97.-122.)

```
atom      = 1*<любой CHAR кроме specials, SPACE и CTL>


authentic  =  "From"     ":"  mailbox  ; Единственный автор
        / ( "Sender"    ":"  mailbox  ; Настоящий отправитель
          "From"      ":" 1#mailbox) ; Много авторов

CHAR      = <любой символ ASCII>    ; (  0-177,  0.-127.)
CTL       = <любой символ ASCII control        ; (  0- 37,  0.- 31.)
          и DEL>                        ; (   177,    127.)
comment    = "(" *(ctext / quoted-pair / comment) ")"


CR        = <ASCII CR, возврат каретки>  ; (    15,     13.)

CRLF      = CR LF

ctext     = <any CHAR excluding "(",    ; => may be folded
          ")", "\" & CR, & including
           linear-white-space>

date      = 1*2DIGIT month 2DIGIT       ; day month year
                                  ; e.g. 20 Jun 82

date-time  = [ day "," ] date time      ; dd mm yy
                                  ; hh:mm:ss zzz

dates     =  orig-date                  ; Отправлено
          [ resent-date ]               ; Переслано

day       = "Mon"  / "Tue" /  "Wed"  / "Thu"
          / "Fri"  / "Sat" /  "Sun"

delimiters  =  specials / linear-white-space / comment

destination =  "To"         ":" 1#address      ; Первичный
            / "Resent-To"   ":" 1#address
            / "cc"          ":" 1#address  ; Вторичный
            / "Resent-cc"   ":" 1#address
            / "bcc"         ":"  #address  ; Слепая пересылка
            / "Resent-bcc"  ":"  #address

DIGIT     = <any ASCII decimal digit>   ; ( 60- 71, 48.- 57.)

domain    = sub-domain *("." sub-domain)

domain-literal =  "[" *(dtext / quoted-pair) "]"

domain-ref  =  atom                     , символическая ссылка

dtext     = <любой CHAR кроме "[",    ; => may be folded
```

```
                "]", "\" и CR, и включая
                linear-white-space>

fields      =   dates               ; Требуется Дата создания
                source              ; id автора и один
                1*destination       ; адрес. Другие поля
                *optional-field     ; факультативные


group       =  phrase ":" [#mailbox] ";"


hour        =  2DIGIT ":" 2DIGIT [":" 2DIGIT]

HTAB        =  <ASCII HT, horizontal-tab>  ; (    11,      9.)

LF          =  <ASCII LF, linefeed>        ; (    12,     10.)

linear-white-space =  1*([CRLF] LWSP-char)  ;


local-part  =  word *("." word)          ; протоколом не интерпретируется

LWSP-char   =  SPACE / HTAB


mailbox     =  addr-spec                 ; простой адрес
            /  phrase route-addr         ; name & addr-spec

message     =  fields *( CRLF *text )    ; Все, что следует после
                                         ; первой пустой строки -
                                         ; это тело сообщения

month       =  "Jan"  /  "Feb" /  "Mar"  /  "Apr"
            /  "May"  /  "Jun" /  "Jul"  /  "Aug"
            /  "Sep"  /  "Oct" /  "Nov"  /  "Dec"

msg-id      =  "<" addr-spec ">"         ; Уникальный идент. сообщения

optional-field =
                / "Message-ID"       ":"  msg-id
                / "Resent-Message-ID" ":"  msg-id
                / "In-Reply-To"      ":"  *(phrase / msg-id)
                / "References"       ":"  *(phrase / msg-id)
                / "Keywords"         ":"  #phrase
                / "Subject"          ":"  *text
                / "Comments"         ":"  *text
                / "Encrypted"        ":"  1#2word
                / extension-field
                / user-defined-field         ; может быть пустым
```

```
orig-date  = "Date"       " "  date-time

originator =  authentic                ; идентифицированный адрес
              [ "Reply-To"  "." 1#address] )


phrase     = 1*word

quoted-pair =  "\" CHAR

quoted-string = <"> *(qtext/quoted-pair) <">;

qtext      = <любой CHAR кроме <">,    ; => may be folded
             "\" и CR, и включая
              linear-white-space>

received   = "Received"    ":"          ; один на
             ["from" domain]            ; оправлюящий узел
             ["by"  domain]             ; получающий узел
             ["via"  atom]              ; физический путь
             *("with" atom)             ; протокол соединения/почты
             ["id"  msg-id]             ; идентификатор сообщения
                                        ; получателя
             ["for"  addr-spec]         ; начальная форма
              ";"   date-time           ; время получения


resent     =  resent-authentic
              [ "Resent-Reply-To"  ":" 1#address] )

resent-authentic =
             =  "Resent-From"      ":"  mailbox
             / ( "Resent-Sender"   ":"  mailbox
                 "Resent-From"     ":" 1#mailbox  )

resent-date =  "Resent-Date" ":"  date-time

return     =  "Return-path" ":" route-addr ; обратный адрес


route-addr  =  "<" [route] addr-spec ">"

route      =  1#("@" domain) ":"          ; path-relative
                                          ; case-preserved

source     = [  trace ]                   ; сетевые узлы,
             originator                   ; переславшие сообщение
           [  resent ]                    ;
```

```
SPACE      = <ASCII SP. пробел>        ; (    40,    32.)

specials   = "("   ")" / "<"   ">" / "@"  ; Для использования
           / "," / ":" / ":" / " "   <">  ; внутри слова должен быть
           / "." / "[" / "]"              ; в кавычках

sub-domain  =  domain-ref / domain-literal

text       = <any CHAR. including bare   ; => atoms, specials.
             CR & bare LF. but NOT       ; comments и
             including CRLF>             ; quoted-strings
                                         ; не распознаются

time       =  hour zone                  ; ANSI и Military
                                         ; 00:00:00 - 23:59:59

trace      =   return                    ; путь к отправителю
             1*received                  ; тэги получения

word       =  atom / quoted-string


zone       =  "UT"  / "GMT"              ; Время по Гринвичу
                                         ; North American : UT
           / "EST" / "EDT"               ; Eastern:  - 5/ - 4
           / "CST" / "CDT"               ; Central:  - 6/ - 5
           / "MST" / "MDT"               ; Mountain: - 7/ - 6
           / "PST" / "PDT"               ; Pacific:  - 8/ - 7
           / 1ALPHA                      ; Military: Z = UT;
                                         ; A:-1; (J not used)
                                         ; M:-12; N:+1; Y:+12
           / ( ("+" / "-") 4DIGIT )      ; Местное время
```

ПРИЛОЖЕНИЕ 4

ТЕХНИЧЕСКИЕ ТРЕБОВАНИЯ К ТЕХНИЧЕСКИМ СРЕДСТВАМ СЛУЖБЫ ДОМЕННЫХ ИМЕН ПРОТОКОЛУ DNS

1. ОБЛАСТЬ ПРИМЕНЕНИЯ

Настоящее приложение описывает технические требования к ТС службы доменных имен по протоколу DNS в соответствии с RFC 1034 [12] и RFC 1035 [13].
В приложении приведены процедуры обращения клиента к распределенной базе данных доменных имен с целью получения информации, связанной с указанным доменным именем, а также описывает функционирование распределенной базы данных доменых имен.

Не все функции. содержащиеся в данном приложении, обязательны для ТС служб Iдоменных имен по протоколу DNS, но если они выполняются, то их реализация должна соответствовать настоящему приложению.

2. ФУНКЦИОНАЛЬНЫЕ ТРЕБОВАНИЯ К СЕРВЕРУ DNS

2.1. Соединения

2.1.1. Протокол нижнего уровня

Сообщения DNS должны передаваться в датаграммах UDP или с использованием виртуального соединения TCP.

Контрольные файлы DNS должны передаваться с использованием протоколов надежной передачи файлов по сети передачи данных.

2.1.2. Требования к использованию протокола UDP

При использовании протокола UDP на сервере должен использоваться порт с десятичным номером 53.

При использовании UDP максимальный размер сообщения DNS должен составлять 512 байт. Более длинные сообщения должны усекаться с установкой бита TC заголовка сообщения (см. п. 4.1.2.). Протокол UDP не должен использоваться для пересылки зоны.

2.1.3. Требования к использованию протокола TCP

При использовании протокола TCP на сервере должен использоваться порт с десятичным номером 53.

При пересылке сообщений по соединению TCP сообщениям должно предшествовать поле длины размером 2 байта, содержащее значение длины сообщения без учета данного дополнительного поля.

При использовании TCP каждый семибитный символ, передаваемый по соединению TCP, должен передаваться в отдельном октете. При этом символ должен быть выровнен вправо, а старший бит октета установлен в 0.

2.1.4. Установление соединения и закрытие соединения

При обмене сообщениями DNS в случае использования виртуального соединения соединение должно инициироваться стороной клиента.

3. ПЕРЕЧЕНЬ И СТРУКТУРА ДАННЫХ, ПЕРЕДАВАЕМЫХ ПО ПРОТОКОЛУ DNS

Данные должны передаваться либо в форме сообщений DNS, либо в форме контрольных файлов. Обмен данными в форме сообщений DNS должен происходить как последовательность запросов DNS и ответов DNS.
Сервер DNS должен поддерживать как клиентскую, так и серверную часть протокола DNS.

3.1. Формат сообщений DNS

3.1.1. Формат сообщений DNS должен соответствовать п. 5.4.1.

3.1.2. Заголовок сообщения DNS должен состоять из полей: ID, QR, OPCODE, AA, TC, RD, RA, Z, RCODE, QDCOUNT, ANCOUNT, NSCOUNT, ARCOUNT.

3.1.3. Формат записей ресурсов в сообщениях DNS должен соответствовать п. 5.3.2.

3.1.4. Должны поддерживаться следующие виды сообщений DNS:
запрос;
ответ.

3.1.5. Должен поддерживаться стандартный тип запроса. Порядок обработки запроса должен соответствовать п. 5.4.2.1.

3.1.6. Если поддерживается инверсный тип запроса, порядок его обработки должен соответствовать п. 5.4.2.2.

3.1.7. Должны поддерживаться следующие коды ответа:

0 - нет ошибки;
1 - ошибка формата запроса;
2 - внутренняя ошибка сервера;
3 - ошибка имени;
4 - данный вид запроса не реализован;
5 - отказ выполнения операции;

3.1.8. Если поддерживаются рекурсивные запросы, порядок согласования выполнения рекурсивного запроса должен соответствовать п. 5.5.5., а алгоритм обработки рекурсивного запроса должен соответствовать п. 5.5.2.

3.1.9. При выдаче сервером ответа в поле ID заголовка ответа должен содержаться идентификационный номер, совпадающий с идентификационным номером соответствующего запроса.

3.1.10. При выдаче ответа авторитетным сервером в поле AA заголовка ответа должна быть установлена

3.1.11. При усечении сообщения в поле TC заголовка сообщения должна устанавливаться 1.

3.1.12. Если в сервере реализована обработка инверсных запросов, формат запроса и ответа должен соответствовать п. 5.4.2.2.

3.2. Кодирование данных в сообщениях DNS

При сравнении любых элементов данных протокола DNS не должно делаться различия между строчными и прописными символами. Символьная форма доменного имени должна соответствовать 5.2.1.2., а двоичная форма доменного имени должна соответствовать 5.2.1.1.

3.3. Сжатие сообщений

Если сервер DNS поддерживает сжатие сообщений, механизм сжатия должен соответствовать п. 5.4.3.

3.4. Контрольные файлы

В сервере должна быть реализована загрузка информации о доменном дереве из управляющего файла в формате, соответствующем п. 5.4.4.

Должны быть исключены ситуации, когда загрузка контрольного файла осуществляется не полностью. При ошибке загрузки контрольного файла сервер имен не должен использовать информацию из данного контрольного файла.

После загрузки контрольного файла записи RR, выдаваемые в ответах сервера, должны соответствовать записям загруженного контрольного файла, если только не поступила информация об изменении зоны .

4. ТРЕБОВАНИЯ К РЕАЛИЗАЦИИ СЕРВЕРА DNS

4.1. Общие требования к реализации сервера DNS

4.1.1. УТС DNS должен обеспечивать функции сервера DNS и функции клиента DNS

4.1.2. При реализации кэширования отрицательных ответов к дополнительной секции авторитетного ответа должна добавляться запись RR типа SOA. Описание типа SOA дано в п. 5.3.2.1.6.

4.1.3. При реализации разрешающей системы и проведения поиска данных по псевдониму, должна быть выполнена цепочка запросов до получения требуемых данных, либо до выяснения ошибочного состояния. Данное требования не распространяется на запросы общего поиска.

4.1.4. Должны поддерживаться записи RR с шаблонными именами владельца согласно п. 5.3.4.

4.1.5. Должен поддерживаться субдомен IN-ADDR.ARPA. согласно п. 5.2.4.

4.2. Требования к размеру элементов протокола

Максимальный размер элементов протокола должен соответствовать табл. 1

Таблица 1

Максимальный размер элементов протокола.

Элементы		Максимальный размер
Labels	Метки	63 октета
Names	Имена	255 октетов
TTL	время жизни	Максимальное положительное значение 32-х битового целого со знаком
UDP messages	пакеты UDP	512 октетов

4.3. Реализация вторичного сервера зоны

4.3.1. Если сервер поддерживает функции вторичного сервера зоны DNS, временные параметры REFRESH, RETRY, EXPIRE процедуры отслеживания изменений зоны на первичном сервере должны соответствовать значениям, установленным в записи RR SOA для данной зоны.

4.3.2. Получение измененной зоны должно выполняться при помощи запроса AXFR. Команда AXFR должна выдаваться первичному серверу при обнаружении увеличения параметра SERIAL записи RR SOA для данной зоны на первичном сервере.

4.4. Реализация первичного сервера зоны

Если сервер поддерживает функции первичного сервера зоны DNS, при получении запроса AXFR он должен выдать ответы, содержащие все записи RR, относящиеся к зоне, указанной в запросе. Причем первое и последнее сообщения ответа должны содержать данные для верхнего авторитетного узла в зоне.

5.ОПИСАНИЕ СИСТЕМЫ ДОМЕННЫХ ИМЕН

5.1. Структура системы доменных имен

5.1.1. Основные компоненты системы доменных имен.

В DNS входят три основные компоненты:

- **пространство доменных имен.** Каждый узел и лист древовидного пространства доменных имен указывает на некоторый набор данных. Операции выполнения запроса

можно рассматривать как попытку выделить определенные типы данных из отдельного набора.

- **серверы имен** Сервером имен называется серверное программное обеспечение, хранящее информацию о структуре доменного дерева и соответствующие наборы данных. Сервер имен может кэшировать структуру и данные любой части доменного дерева, но, как правило, отдельный сервер имен содержит полную информацию о каком-либо подмножестве доменного пространства и указатели на другие серверы имен, с помощью которых можно найти информацию о любой части доменного дерева. Если сервер обладает полной информацией о какой-либо части доменного дерева (о зоне доменного дерева), его называют авторитетным сервером (AUTHORITY) данной зоны, а информацию об этой зоне, хранимую на данном сервере - авторитетной информацией.

разрешающая система. Разрешающая система обеспечивает доступ к распределенной базе данных доменных имен, расположенной на множестве серверов имен. Разрешающая система позволяет продолжить запрос клиента в случае отсутствия на конкретном сервере запрашиваемой информации.

С точки зрения клиента доменное пространство состоит из одного дерева, причем доступна любая его часть. С точки зрения разрешающей системы доменная система состоит из неизвестного количества серверов имен. С точки зрения сервера имен доменная система состоит из отдельных наборов локальной информации, называемых зонами. Сервер должен одновременно обслуживать запросы от нескольких разрешающих систем.
Предполагается, что в доменной системе все данные находятся в контрольных файлах (master files), распределенных по узлам сети, использующим доменную систему. Изменения в данные контрольных файлов заносятся системными администраторами. Контрольные файлы имеют стандартный текстовый формат, поэтому различные узлы могут обмениваться контрольными файлами с использованием любых средств передачи файлов. Серверы имен являются хранилищем информации, составляющей распределенную базу данных имен домена. База данных разделена на секции, называемые зонами. Зоны распределены между серверами имен. Для повышения надежности, одна зона должна содержаться на нескольких серверах имен. Один сервер имен обычно поддерживает одну или несколько зон. Он имеет авторитетную информацию о достаточно небольшом участке пространства имен и также может иметь неавторитетную кэшированную информацию о некоторых других частях дерева. Кэшированная информация используется местной разрешающей системой. Сервер имен помечает ответы таким образом, чтобы запросчик мог различать авторитетные данные от неавторитетных. Периодически сервер имен делает проверку актуальности хранимых им данных, и, в случае, если хранимые данные не актуальны, сервер получает копию обновленной зоны из контрольных файлов, хранящихся локально или на другом сервере имен.

5.1.2. Основные конфигурации взаимодействия сервера, клиента и разрешающей системы в системе DNS.

Клиентом называется процесс, использующий функции системы доменных имен. Сервером имен называется процесс, выполняющий совместно с процессом клиента функции доступа к системе доменных имен. Внешний сервер имен - сервер имен, который взаимодействует с клиентом через разрешающую систему другого сервера.
Как правило клиент выполняет функции либо часть функций разрешающей системы, что дает ему потенциальную возможность доступа к системе доменных имен путем обращений к нерекурсивным серверам имен. Клиент, у которого не реализованы функции разрешающей системы, при взаимодействии с нерекурсивным сервером имен будет иметь возможность доступа только к той части доменного пространства, которая хранится на данном сервере имен.

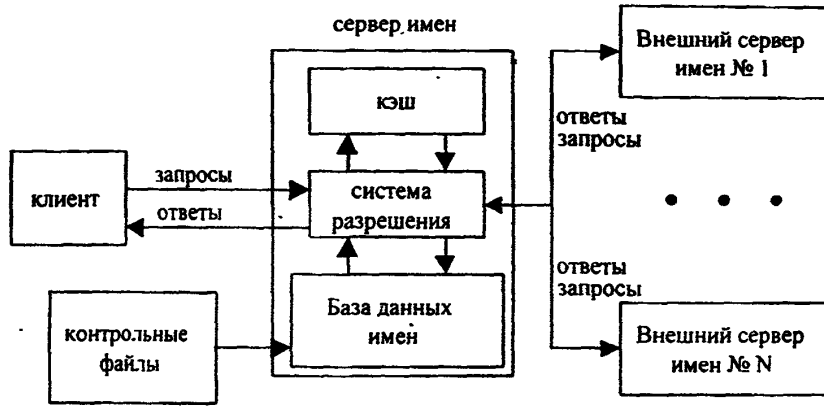


Рисунок. 1. Доступ клиента к пространству доменных имен через рекурсивный сервер имен

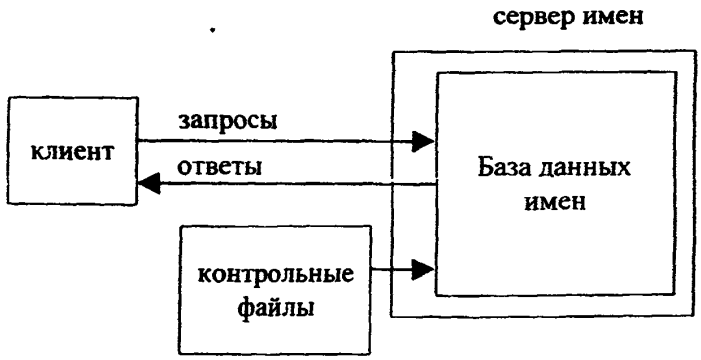


Рисунок. 2. Доступ клиента к пространству доменных имен через нерекурсивный сервер имен

5.2. Структура пространства доменных имен

Пространство доменных имен является древовидной структурой Каждый узел и лист дерева соответствует определенному набору данных о ресурсах. Этот набор данных может быть пустым. Доменная система одинаково использует внутренние узлы и листья дерева, поэтому далее листья называются тоже узлами дерева. Каждому узлу дерева присвоена метка длиной от 0 до 63 символов. Длина метки корневого узла должна быть равна 0.

Имя домена состоит из меток, находящихся на пути от данного узла к корню доменного дерева и идентифицирует узел доменного дерева. Набор данных о ресурсе, связанный с отдельным доменным именем, содержится в одной или более записях ресурса (RR). Порядок следования RR в наборе для одного доменного имени является несущественным.

Домен А называют субдоменом другого домена В, если А содержится в домене В. Кроме разделения на субдомены рассматривают разделение доменов на классы и зоны. Разделение домена на классы осуществляется в соответствии со значением поля CLASS записей RR, в которых хранится информация о домене. Разделение на зоны рассматривается в п. 6.2.3.

Замечание: Необходимо различать понятия "узел доменного дерева" и "узел сети передачи данных". Узел сети передачи данных может иметь несколько сетевых адресов, которым будут соответствовать несколько узлов доменного дерева. Узел сети передачи данных может вообще не иметь доменного имени и соответствующего узла доменного дерева.

5.2.1. Доменные имена

Доменное имя узла - это список меток, находящихся на пути от данного узла к корню доменного дерева. Метки, составляющие доменное имя, должны располагаться слева направо, от метки, наиболее удаленной от корня, к метке, наиболее близкой к корню. При хранении доменных имен запоминается регистр символов, но при любых сравнениях меток регистр не учитывается. Доменное имя может быть представлено в двух формах: в символьной форме и в двоичной форме.

5.2.1.1. Внутренняя форма представления доменного имени

Каждая метка хранится в виде одного октета длины, за которым следует некоторое количество октетов. содержащих символы метки. Так как каждое доменное имя заканчивается нулевой меткой, обозначающей корень. представление каждого доменного имени заканчивается октетом длины, содержащим 0. Октет длины содержит два старших бита, установленных в 0. Оставшиеся 6 битов содержат значение длины поля символов метки от 0 до 63. Общая длина доменного имени (сумма всех октетов символов и октетов длин) не должна превышать 255 октетов.

Внутренняя форма представления доменного имени должна соответствовать следующему определению:

Все символы должны быть закодированы в ASCII

```
domain-name ::= [ subdomain ] nul-label
subdomain ::= label / (subdomain  label)
label ::= length letter [ [ ldh-str ] let-dig ] ; максимальная длина label
                                                ; составляет 63 символа
length ::= 2(0bit) len-val
len-val ::= 6(Xbit)          ; 6-битное значение длины соответствующей
```

```
                    ; метки label
nul-label ::= 8(0bit)
character-string ::= s_length *256(char)      ; символьная строка
s_length ::= 8(Xbit)                          ; длина символьной строки в
                                              ; октетах

ldh-str ::= let-dig-hyp / (let-dig-hyp ldh-str)
let-dig-hyp ::= let-dig / "-"
let-dig ::= letter / digit
letter ::= <любая из 52 алфавитных символов кода ASCII
          от A до  Z и от a до z>
char ::= <любой символ кода ASCII>
digit ::= <любая из 10 цифр от  0 до  9>
Xbit ::= <бит>
0bit ::= <бит, установленный в 0>
unsigned_int32 ::= 32(Xbit) ; 32-разрядное целое без знака
```

5.2.1.2. Печатная форма представления доменного имени

В печатной форме доменные имена представляются как список меток, разделенных одной точкой.

Все символы должны кодироваться в ASCII.

```
domain ::= subdomain / " "
subdomain ::= label / (subdomain "." label)
label ::= letter [ [ ldh-str ] let-dig ] ; максимальная длина label
                                        ; составляет 63 символа

ldh-str ::= let-dig-hyp / (let-dig-hyp ldh-str)
let-dig-hyp ::= let-dig / "-"
let-dig ::= letter / digit
letter ::= <любая из 52 алфавитных символов кода ASCII
          от A до  Z и от a до z>
digit ::= <любая из 10 цифр от  0 до  9>
```

В печатной форме различают полное доменное имя, имеющее точку на правом конце и неполное (относительное) доменное имя без точки на правом конце. Неполные имена используются относительно хорошо известного источника, либо относительно списка доменов, используемых в качестве поискового списка.

5.2.2. Псевдонимы и канонические имена

Узлы и другие ресурсы часто имеют несколько имен. Как правило одно из набора эквивалентных имен называют каноническим или первичным, а остальные - псевдонимами. Псевдоним присваивается узлу с помощью соответствующей записи RR типа CNAME. Запись RR типа CNAME содержит в разделе "владелец" псевдоним владельца, а в разделе RDATA - соответствующее каноническое имя. Если в узле присутствует RR типа CNAME, то не должно быть никаких других данных для этого узла.

Когда сервер имен не может найти желаемый RR в наборе, связанном с доменным именем, сервер проверяет, нет ли в наборе RR записей типа CNAME соответствующего класса. Если есть, сервер имен включает запись RR CNAME в ответ и возобновляет запрос

к доменному имени, указанному в поле RDATA записи CNAME. Запросы, соответствующие типу CNAME, не возобновляются.

Если в RR типа CNAME вместо канонического имени указан псевдоним, сервер DNS должен проследить всю цепочку таких RR, пока не будет найдено каноническое имя, либо не будет выявлена ошибка.

5.2.3. Зоны домена

5.2.3.1. Разбиение домена на зоны

Внутри класса домена могут быть сделаны "разрезы" между двумя любыми смежными узлами. После того, как это сделано, каждая отдельная группа не разделенных разрезами узлов образует одну зону. Считается, что зона авторитетна для всех узлов в связанной области. Для разных классов "разрезы" могут быть произведены в разных местах.

Таким образом каждая зона имеет как минимум один узел и, следовательно, как минимум одно доменное имя, для которого данная зона авторитетна. В древовидной структуре домена каждая зона имеет один верхний узел, расположенный наиболее близко к корню дерева по сравнению с другими узлами зоны. Имя этого верхнего узла часто используется для обозначения зоны.

Данные, описывающие зону, могут быть разделены на четыре части:

- Авторитетные данные для всех узлов данной зоны
- Данные, определяющие верхний узел зоны
- Данные, описывающие делегированные подзоны
- Данные, позволяющие получить доступ к доменным именам подзон (так называемые "клеевые" записи).

Все эти данные представлены в виде записей RR. Таким образом зона может быть полностью описана набором RR. Зона может быть перенесена с одного сервера имен на другой путем передачи записей RR либо путем передачи целого контрольного файла.

Записи RR, описывающие верхний узел зоны, разделяются на: записи NR RR, представляющие список всех серверов имен данной зоны, и одну запись SOA RR, описывающую параметры управления зоной.

Записи RR, описывающие "разрезы" на "дне" зоны (определяющие подзоны), обозначаются NS RR и указывают на сервер имен подзоны.

Данные NS RR не являются частью авторитетных данных зоны, так как определяют "разрезы" между узлами, а не узлы (так как не содержат адресной информации). Каждая запись NS RR должна быть идентична верхней записи RR соответствующей подзоны. В отдельной зоне записи NS RR должны располагаться рядом с верхним узлом зоны (они входят в число авторитетных) или на "разрезах" вокруг "дна" зоны (они не входят в число авторитетных), но никогда не должны располагаться посредине зоны.

Так как записи NS RR содержат только имена серверов подзон, но не содержат адресов серверов подзон, зона содержит "клеевые" записи RR (не являющиеся авторитетными), которые являются адресными RR для данных серверов. "Клеевые" RR используются в ссылочном ответе.

5.2.3.2. Распространение изменений зоны

Для распространения изменений зоны может использоваться любая процедура передачи контрольного файла, но предпочтительным методом является использование сообщений протокола DNS.

В основной модели автоматической пересылки и обновления зоны один из серверов назначается первичным для данной зоны Изменения производятся на первичном сервере

(обычно путем исправления контрольного файла зоны). После исправления администратор дает первичному серверу загрузить новую зону. Вторичные серверы зоны периодически проверяют наличие изменений и получают новые копии зоны в случае, если сделаны изменения.

Для проверки наличия изменений вторичные серверы проверяют поле SERIAL записи SOA данной зоны. Если в зоне производятся какие-либо изменения, значение поля SERIAL всегда увеличивается. Механизм изменения может быть либо простым увеличением значения, либо может использовать текущую дату.

Параметры периодической переклички вторичных серверов устанавливаются в SOA RR данной зоны. Этими параметрами являются: REFRESH, RETRY и EXPIRE. Когда вторичный сервер загружает новую зону, он ждет REFRESH секунд перед новой проверкой поля SERIAL. Если не обнаружено изменений (значение поля SERIAL - прежнее), вторичный сервер опять ждет REFRESH секунд. В случае, если проверка не может быть произведена, сервер ждет RETRY секунд до повторной проверки. Если вторичному серверу не удалось провести проверку в течение EXPIRE секунд, он должен считать копию зоны устаревшей и уничтожить ее.

Это значит, что в случаях ошибки чтения зоны или в случае просрочки обновления зоны, сервер имен должен выдавать ответы на запросы таким образом, как если бы он вообще не владел этой зоной. Во время пересылки новых данных о зоне сервер имен должен выдавать старые данные, пока пересылка не будет полностью закончена. Если вторичный сервер обнаружил, что на первичном сервере зона изменена, сервер должен запросить командой AXFR передачу зоны. Первое и последнее сообщения, передаваемые в ответ на команду, должны содержать данные для верхнего авторитетного узла в зоне. Промежуточные сообщения должны содержать остальные RR зоны, как авторитетные, так и неавторитетные. Команда AXFR должна использовать протокол TCP. Данный механизм копирования зон может применяться вторичными серверами не только к первичным серверам, но и к другим вторичным серверам.

5.2.4. Домен IN-ADDR.ARPA

Internet использует специальный субдомен для поддержки трансляции адресов узлов. Назначение этого субдомена - предоставлять надежный метод трансляции адресов узлов в имена узлов, а также поддерживать запросы по установлению местонахождения всех шлюзов в отдельной сети в Internet.

Обе указанные функции аналогичны функциям, выполняемым инверсными запросами. Отличие заключается в том, что эта часть доменного имени структурирована согласно адресу, то есть гарантируется, что соответствующие данные могут быть найдены без выполнения поиска в доменном пространстве.

Субдомен начинается с метки IN-ADDR.ARPA и имеет подструктуру, соответствующую структуре адресации Internet. Доменные имена могут иметь до 4-х меток в дополнение к IN-ADDR.ARPA. Каждая метка представляет собой один октет адреса IP в формате десятичного целого от 0 до 255 (с опущенными лидирующими нулями). Адрес узла соответствует имени, в котором присутствуют все 4 метки. Например, узел с адресом 10.2.0.52 будет иметь имя 52.0.2.10.IN-ADDR.ARPA.

Имена с количеством меток менее 4-х соответствуют зоне, выделенной для отдельной сети.

Сетевые номера соответствуют некоторым неконечным узлам, располагающимся на различной глубине домена IN-ADDR.ARPA. Сетевые узлы используются для хранения указателей на первичные узловые имена шлюзов, присоединенных к данной сети. Так как шлюз находится более чем в одной сети, имеются два или более сетевых узлов, указывающих на него. Шлюзы также имеют указатели уровня узла на их полные адреса.

Шлюзовые указатели на сетевые узлы и обычные узловые указатели на полные сетевые адреса используют PTR RR для обратной ссылки на первичные доменные имена соответствующих узлов.

Пример:

Домен IN-ADDR.ARPA содержит информацию:
о шлюзе MILNET-GW.ISI.EDU между сетями 10 и 26 с адресами 10.2.0.22 и 26.0.0.103
о шлюзе GW.LCS.MIT.EDU между сетями 10 и 18 с адресами 10.0.0.77 и 18.10.0.4
об узле A.ISI.EDU
об узле MULTICS.MIT.EDU

База данных домена будет содержать:

```
10.IN-ADDR.ARPA.              PTR MILNET-GW.ISI.EDU.
10.IN-ADDR.ARPA.              PTR GW.LCS.MIT.EDU.
18.IN-ADDR.ARPA.              PTR GW.LCS.MIT.EDU.
26.IN-ADDR.ARPA.              PTR MILNET-GW.ISI.EDU.
22.0.2.10.IN-ADDR.ARPA.       PTR MILNET-GW.ISI.EDU.
103.0.0.26.IN-ADDR.ARPA.      PTR MILNET-GW.ISI.EDU.
77.0.0.10.IN-ADDR.ARPA.       PTR GW.LCS.MIT.EDU.
4.0.10.18.IN-ADDR.ARPA.       PTR GW.LCS.MIT.EDU.
103.0.3.26.IN-ADDR.ARPA.      PTR A.ISI.EDU.
6.0.0.10.IN-ADDR.ARPA.        PTR MULTICS.MIT.EDU.
```

Программа, желающая найти шлюз в сеть 10, выдаст запрос со значениями:
QTYPE=PTR, QCLASS=IN, QNAME=10.IN-ADDR.ARPA.
и получит ответ с записями RR:

```
10.IN-ADDR.ARPA.              PTR MILNET-GW.ISI.EDU.
10.IN-ADDR.ARPA.              PTR GW.LCS.MIT.EDU.
```

Разрешающая система, желающая найти имя узла с адресом 10.0.0.6 выдаст запрос со значениями: QTYPE=PTR, QCLASS=IN, QNAME=6.0.0.10.IN-ADDR.ARPA
и получит в ответ:

```
6.0.0.10.IN-ADDR.ARPA.        PTR MULTICS.MIT.EDU.
```

Замечание: если узел имеет два имени в различных доменах, только одно из этих имен может быть каноническим.

5.3. Записи ресурсов

Имя домена идентифицирует узел доменного дерева. В каждом узле имеется набор связанной информации о ресурсе. Этот набор может быть пустым. Набор информации о ресурсе, связанный с отдельным доменным именем содержится в одной или нескольких записях ресурса (RR). Порядок RR в наборе, относящемуся к одному доменному имени, является несущественным.

5.3.1. Общая структура записи ресурса RR

В отдельную запись ресурса RR входят разделы, приведенные в табл. 2

Таблица 2

Разделы отдельной записи ресурса RR

Имя элемента		Описание
Владелец	owner	доменное имя, в котором находится RR
Тип	type	16-битное значение, определяющее тип ресурса: A - адрес узла CNAME - каноническое имя HINFO - процессор или ОС, используемые узлом MX - почтовый шлюз домена NS - авторитетный сервер имен домена PTR - указатель на другую часть пространства доменных имен SOA - начало авторитетной зоны
Класс	class	16-битное значение, определяющее семейство или отдельный протокол: IN - система Internet CH - система Chaos
Время жизни	TTL	
Данные ресурса	RDATA	Данные, описывающие ресурс и зависящие от типа и класса записи: A - для IN 32-х битный адрес IP; для CH - имя домена и последующий 16-битный адрес Chaos CNAME - имя домена MX - 16-битное значение приоритета, за которым следует имя узла, работающего как почтовый шлюз для домена - владельца NS - имя узла PTR - имя домена SOA - несколько полей

Поле TTL интерпретируется как предел времени хранения RR в кэше. TTL не применяется для авторитетных данных внутри зоны. Для авторитетных данных внутри зоны применяется специальная временная организация. TTL назначается администратором для зоны, из которой поступают данные. TTL=0 запрещает кэширование данных. Так, записи SOA всегда имеют TTL=0. Если ожидается изменение RR, перед проведением изменения ее TTL может быть уменьшено для сокращения периода несоответствия во время замены. После проведения замены TTL увеличивается обратно до прежнего значения.

Данные в разделе RDATA хранятся как комбинация бинарных строк и имен домена. Имена домена часто используются как указатели на другие данные системы имен.

5.3.2. Формат представления RR в сообщении DNS

5.3.2.1. Формат RR приведен на рис. 3.

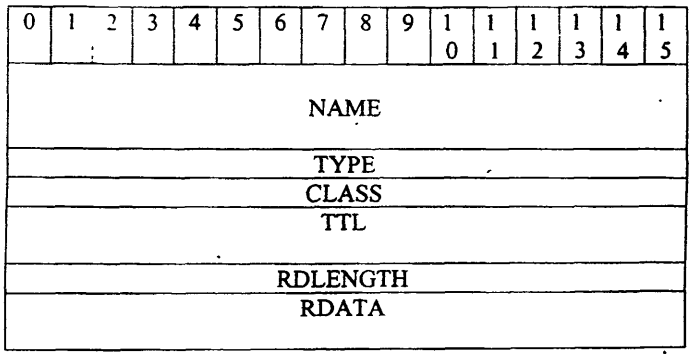


Рисунок. 3. Формат представления RR

Описание значений полей формата RR приведено в табл. 3.

Таблица 3

Описание значений полей формата RR

Поле	Длина, октет	Значение
NAME	-	Доменное имя владельца в печатной форме согласно п. 6.2.1.2.
TYPE	2	Тип RR
CLASS	2	Класс RR
TTL	4	32-битовое целое со знаком - время жизни
RDLENGTH	2	16-битовое целое без знака - длина поля RDATA в октетах
RDATA	-	Описание ресурса в соответствии со значениями TYPE и CLASS

5.3.2.2. Значения поля TYPE должны соответствовать табл. 4.

Таблица 4

Значения поля TYPE

Тип	Значение	Описание
Значения, разрешенные в запросе и ответе		
A	1	адрес узла
NS	2	авторитетный сервер имен
CNAME	5	каноническое имя для псевдонима
SOA	6	отметка начала авторитетной зоны
WKS	11	описание сервиса
PTR	12	указатель доменного имени
HINFO	13	информация об узле
MINFO	14	информация о почтовом ящике или списке рассылки
MX	15	почтовый шлюз
TXT	16	текстовая строка
Значения, разрешенные только в запросе (значения поля QTYPE)		
AXRF	252	запрос пересылки целой зоны
*	255	запрос всех записей

5.3.2.3. Значения поля CLASS

Значения поля CLASS приведены в табл. 5.

Таблица 5

Значение поля CLASS

Тип	Значение	Описание
Значения, разрешенные в запросе и ответе		
IN	1	Internet
CH	3	CHAOS
HS	4	Hesiod
Значения, разрешенные только в запросе (значения поля QCLASS)		
*	255	любой класс

5.3.2.4. Формат и значения поля RDATA

Все символы должны быть закодированы в ASCII

domain-name ::= <имя домена во внутренней форме представления>
Xbit ::= <бит>
0bit ::= <бит, установленный в 0>
unsigned_int32 ::= 32(Xbit) ; 32-разрядное целое без знака

Поле RDATA может иметь следующие типы форматов: CNAME-RDATA, HINFO-RDATA, MX-RDATA, NS-RDATA, PTR-RDATA, SOA-RDATA, TXT-RDATA, A-RDATA, WKS-RDATA.

5.3.2.4 1. **CNAME-RDATA ::= domain-name**

; domain-name содержит каноническое имя владельца. Имя владельца является псевдонимом.

5.3.2.4.2. **HINFO-RDAT ::= CPU OS**

CPU ::= character-string ; указывает на тип процессора
OS ::= character-string ; указывает операционную систему

; значения CPU и OS должны соответствовать RFC-1700[9]

5.3.2.1.3. **MX-RDATA ::= preference exchange**

preference ::= <16-битное целое> ; приоритет данной RR
; по отношению к другим RR того же владельца
exchange ::= domain-name ; узел почтового шлюза для данного
; *владельца*

; запись MX влечет образование дополнительной секции типа A
; для узла, указанного как шлюз

5.3.2.1.4. **NS-RDATA ::= NSDNAME**

NSDNAME ::= domain-name ; авторитетный узел для данного
; класса или домена

; NS RR объявляет, что узел с указанным именем имеет
; зону в указанном классе , начинающуюся от имени владельца

; Запись NS влечет образование обычной дополнительной
; секции для размещения записи типа A и,
; в случае использования в ссылке, специальный поиск зоны,
; в которой они будут располагаться в качестве "клеевых"
; данных

5.3.2.1.5. **PTR-RDATA ::= PTRDNAME**

PTRDNAME ::= domain-name ; указывает на некоторую позицию
; в пространстве доменных имен

; данные RR используются в особых доменах для указания на
; некоторые другие позиции в доменном пространстве
; (например, данные RR используются в домене IN-ADDR.ARPA

5.3.2.1.6. SOA-RDATA ::= MNAME RNAME SERIAL REFRESH
RETRY EXPIRE MINIMUM

MNAME ::= domain-name ; сервер имен, являющийся первичным
; для данной зоны
RNAME ::= domain-name ; почтовый ящик лица, ответственного
; за данную зону

SERIAL ::= unsigned_int32 .номер версии первичной копии зоны
REFRESH ::= unsigned_int32 .временной интервал перед тем,
; как данные о зоне должны быть
; обновлены
RETRY ::= unsigned_int32 : временной интервал перед
; повтором неудачного запроса

EXPIRE ::= unsigned_int32 : временной интервал от момента
; последнего обновления копии зоны,
; в течение которого эта копия
; считается авторитетной
MINIMUM ::= unsigned_int32 ; минимальное значение TTL,
; для экспортируемых RR данной зоны

Единицы всех временных значений - секунды.
Значение MINIMUM является нижней границей для значений TTL
всех записей в данной зоне

5.3.2.1.7. **TXT-RDATA ::= 1*(character-string)**

5.3.2.1.8. **A-RDATA ::= ADDRESS**

ADDRESS ::= <32-битный адрес IP>

; Узлы, имеющие несколько адресов IP, имеют несколько
; записей RR A
; в контрольном файле A-RDATA хранится как 4 десятичных
; числа, разделенных точками без пробелов.

5.3.2.1.9. **WKS-RDATA ::= ADDRESS PROTOCOL BIT-MAP**

ADDRESS ::= <32-разрядный адрес IP>
PROTOCOL ::= 8(Xbit) ; номер протокола IP
BIT-MAP ::= *(8(Xbit)) ; битовая маска

; запись WKS предназначена для описания хорошо известных
; сервисов, поддерживаемых отдельным протоколом на отдельных
; адресах IP
; Битовая маска указывает порт протокола. Первый бит
; соответствует 0-му порту, второй - 1-му и т.д.
; Значения номеров протоколов и портов должны
; соответствовать RFC 1700 [9]

5.3.3. Формат RR в контрольных файлах

Большинство RR занимают единственную строку, хотя возможны строки продолжения с использованием скобок.
Для улучшения читаемости могут быть включены пустые строки.
Начало строки указывает владельца. Если начало строки пустое, тогда владелец предполагается таким же, как и в предыдущей RR.
Далее идут TTL, класс и тип .

Более подробно формат RR в контрольном файле описан в п. 6.4.4.

5.3.4. Шаблоны

Имя владельца в записи RR может начинаться с символа "*". Такие RR называются шаблонами. Наиболее часто шаблоны используются для создания зон, которые в свою очередь, используются для перенаправления почты из Internet в некоторую другую почтовую систему. Любое имя, соответствующее шаблону, будет принадлежать такой зоне и обладать определенными свойствами согласно данным, указанным в RR с шаблоном, если только не существует RR, точно соответствующий имени.

Шаблоны не применяются, когда:

- запрос принадлежит другой зоне,
- известно, что существует запрашиваемое имя либо имя между запрашиваемым именем и шаблоном.

Например, если есть RR - шаблон с именем владельца "*.X" и в данной зоне также содержатся RR, прикрепленные к B.X, шаблоны будут применяться к запрашиваемому имени Z.X, но не к запрашиваемому имени B.X, A:B.X или X.

Символ "*" в запрашиваемом имени не имеет специального значения, но может использоваться для тестирования шаблонов в авторитетной зоне. Запрос с "*" является единственным способом получить ответ, содержащий RR - шаблоны. Результат такого запроса не должен кэшироваться.

Пример использования шаблонов:

Пусть существует большая компания с большой сетью не-TCP/IP. Эта компания хочет создать почтовый шлюз. Если компания названа X.COM, и шлюз TCP/IP назван A.X.COM, то в зону COM могут быть введены следующие записи RR.

```
X.COM       MX    10    A.X.COM

*.X.COM     MX    10    A.X.COM

A.X.COM     A     1.2.3.4
A.X.COM     MX    10    A.X.COM

*.A.X.COM   MX    10    A.X.COM
```

Данные записи будут заставлять сервер на любой запрос MX для любого доменного имени, заканчивающегося X.COM возвращать запись MX RR, указывающую на A.X.COM. Последний шаблон необходим, так как действие первого шаблона перекрывается 4-й строкой.

5.4. Взаимодействие по протоколу DNS

Взаимодействие по протоколу DNS осуществляется путем обмена сообщениями DNS и контрольными файлами DNS.

5.4.1. Формат сообщений DNS

При взаимодействии по протоколу DNS путем обмена сообщениями DNS одна взаимодействующая сторона DNS, которая инициирует соединение, называется клиентом, а

...тая - сервером. Взаимодействие по протоколу DNS заключается в том, что клиент ...сылает серверу сообщения-запросы, а сервер выдает сообщения-ответы на запросы ...ента. Сообщения DNS имеют единый формат.

...рмат сообщения DNS должен соответствовать рис. 4

Заголовок	Header	
Запрос	Question	Вид запроса серверу имен
Ответ	Answer	Записи RR, содержащие ответ
Авторитетный	Authority	Записи RR, указывающие на авторитетный сервер
Дополнительно	Additional	Записи RR, содержащие дополнительную информацию

Рисунок. 4. Формат сообщения DNS

Секция заголовка является обязательной.

Формат заголовка показан на рис. 5.

1.	2.	3.	4.	5.	6.	7.	8.	9.	10.	11.	12.	13.	14.	15.	16.
ID															
QR	Opcode				AA	TC	RD	RA	Z			RCODE			
QDCOUNT															
ANCOUNT															
NSCOUNT															
ARCOUNT															

Рисунок. 5. Формат заголовка DNS

...писание полей заголовка DNS приведено в табл. 6.

Таблица. 6

Описание полей заголовка DNS

Обозначение	Длина. бит	Описание
ID	16	Идентификатор, генерируемый источником запроса. Указывается в соответствующем ответе.
QR	1	0 – запрос 1 – ответ
OPCODE	4	Тип запроса: 0 - стандартный (QUERY) 1 - инверсный (IQUERY) от 3 до 15 – зарезервировано
AA	1	Авторитетный ответ. Показывает, что ответ выдан авторитетным сервером для домена, к которому относится указанное в запросе имя.
TC	1	Показывает, что сообщение было усечено в связи с превышением длины допустимого сообщения для канала передачи.
RD	1	Желательна рекурсия. Если установлен в запросе, это показывает серверу имен, что данный запрос нужно обрабатывать рекурсивно. Копируется в ответ.
RA	1	Рекурсия доступна. Устанавливается в ответе, если сервер поддерживает рекурсию.
Z	3	Зарезервировано
RCODE	4	Код ответа *0 - нет ошибки* 1 - ошибка формата запроса 2 - внутренняя ошибка сервера 3 - ошибка имени (только для авторитетного сервера) 4 - данный вид запроса не реализован 5 - Отказ выполнения операции от 6 до 15 – зарезервировано
QDCOUNT	16	целое без знака, указывающее количество позиций в секции "запроса"
ANCOUNT	16	целое без знака, указывающее количество RR в секции "ответа"
NSCOUNT	16	целое без знака, указывающее количество RR в секции "авторитетный"
ARCOUNT	16	целое без знака, указывающее количество RR в секции "дополнительно"

Формат секции запроса показан на рис. 6.

QNAME
QTYPE
QCLASS

Рисунок. 6. Формат секции запроса

Описание формата секции запроса приведено в табл. 7

Таблица 7

Формат секции запроса

Обозначение	Длина, октет	Описание
QNAME	-	Имя домена во внутренней форме согласно 6.2.1.1.
QTYPE	2	код типа запроса
QCLASS	2	код класса запроса

5.4.2. Типы запросов и соответствующие им ответы

Ответ сервера имен либо отвечает на запрос, либо ссылается на другое множество серверов имен, либо сигнализирует состояние ошибки.

Как правило, клиенты не генерируют запросы к серверу имен непосредственно, а используют разрешающую систему, которая в свою очередь, посылает один или несколько запросов серверу имен, а затем обрабатывает ошибки и результаты запросов.

Тип запроса определяется значением 4-битного поля заголовка opcode. Opcode может иметь значения, трактуемые как стандартный запрос, инверсный запрос или запрос статуса.

Четыре секции запроса, следующие за заголовком, описаны в табл. 8.

Таблица 8

Секции запрса, следующие за заголовком

Question	Вопрос	Содержит имя запроса и параметры запроса
Answer	Ответ	Содержит записи RR, прямо отвечающие на запрос
Authority	Авторитетный	Содержит записи RR, которые описывают другие авторитетные серверы. Может (необязательно) содержать SOA RR для авторитетных данных в секции Ответ.
Additional	Дополнительно	Содержит записи RR, которые могут быть полезны при использовании записей RR, содержащихся в других секциях.

Содержание секций зависит от кода поля opcode.

5.4.2.1. Стандартные запросы

Стандартный запрос представляет собой целевое доменное имя QNAME, тип запроса (QTYPE) и класс запроса (QCLASS). Выполнение его предполагает возвращение соответствующих записей RR. Длина полей QTYPE и QCLASS составляет 16 бит. Поле QTYPE может содержать значения, приведенные на рис. 7.

<any type>	<любой тип>	подходит только данный тип записи
AXFR		специальная передача зоны QTYPE
*		подходят все типы записей RR

Рисунок 7. Значения поля QTYPE

Поле QCLASS может содержать значения, приведенные на рис. 8.

<any class>	<любой класс>	подходит только данный класс записи
*		подходят все классы записей RR

Рисунок 8. Значения поля QTYPE

Кроме информации, жестко удовлетворяющей условию запроса, сервер может возвращать в ответе некоторую дополнительную информацию.

Примечание: ответ на запрос с полем типа QTYPE, содержащим "*", никогда не будет авторитетным, так как отдельный сервер не имеет информации, является ли ответ авторитетным для всех классов.

5.4.2.2. Инверсные запросы

Сервер имен может поддерживать инверсные запросы. Если сервер имен не *поддерживает инверсный запрос, на подобный запрос он должен выдать ответ "Не реализовано" (Not-implemented).*

Инверсный запрос должен содержать единственную RR в секции ответа. Поля owner и TTL являются незначащими. Выдаваемый ответ должен содержать запросы в секции запроса, определяющие все имена, информацию о которых данный сервер имен имеет. Так как ни один сервер имен не обладает полной информацией обо всем домене, никогда нельзя сказать, является ли полным ответ на инверсный запрос.

В ответ на инверсный запрос сервер имен должен выдать:

- 0, одно или несколько доменных имен в секции QNAMES для указанного ресурса;
- код ошибки, указывающий, что сервер имен не поддерживает инверсные запросы для ресурса указанного типа.

Когда на инверсный запрос возвращается ответ, содержащий один или несколько имен, значения имени владельца и TTL в секции ответа, определяемые инверсный запрос, должны быть равны значениям из записи RR, соответствующей первому имени QNAME в списке.

В ответе на инверсный запрос может не содержаться истинный TTL, и могут не указываться случаи, когда идентифицированная RR является одной из множества. Поэтому записи RR, получаемые из ответов на инверсные запросы, не должны кэшироваться.

5.4.3. Сжатие сообщений

С целью уменьшения размера сообщений, доменная система может использовать метод сжатия, позволяющий избавиться от повторения доменных имен в сообщении. В данном методе целое доменное имя, представленное во внутреннем формате, заменяется указателем на предыдущее появление данного имени.

Формат указателя должен соответствовать рис. 9.

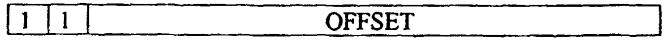


Рисунок. 9. Формат указателя

Первые два бита позволяют различать октеты указателя и октет длины метки. Поле OFFSET указывает на смещение в байтах от начала сообщения (то есть первого октета поля ID заголовка сообщения).

Метод сжатия позволяет доменному имени в сообщении быть представленным в одной из трех форм:

- доменное имя во внутреннем формате - последовательность меток с нулевым октетом в конце
- указатель
- последовательность меток с указателем в конце.

Реализация метода сжатия сообщений является не обязательной.

5.4.4. Контрольные файлы

Контрольные файлы - это текстовые файлы, содержащие записи RR в текстовой форме.

5.4.4.1. Формат контрольного файла

Контрольный файл состоит из набора записей.
Возможен перенос записи на следующую строку с использованием скобок.
Комментарии начинаются с символа ";" (точка с запятой).
Символы пробелов и табуляции выполняют функции разделителей между элементами позиции.

В контрольном файле могут быть записи следующего формата:

```
entry ::= ( <SP> [ comment ] ) /
        ( $ORIGIN <SP>  domain-name [ <SP> comment] ) /
        ( $INCLUDE <SP> file-name [ <SP> domain-name ]
                                [comment] ) /
        ( domain-name rr [ <SP> comment]) /
        ( <SP> rr [ comment] )

<SP> ::= (<SP> (" " / <TAB>)) / (" " / <TAB>)
```

Пустые строки с комментариями и без комментариями допустимы в любом месте контрольного файла.

domain-name - имя домена в символьной форме.

Имена доменов, заканчивающиеся символом "." (точка), называются абсолютными и считаются полными именами.

Доменные имена, не заканчивающиеся символом "." (точка), называются относительными.

Полное доменное имя получается соединением относительного имени и текущего основания относительного имени. Употребление относительного имени в случае неустановленного основания является ошибкой.

Позиция $ORIGIN переустанавливает текущее основание относительного доменного имени.

Позиция $INCLUDE вставляет указанный файл в текущий контрольный файл. Позиции $ORIGIN вставляемого файла не оказывают влияния на текущее основание относительного имени в родительском контрольном файле.

Элемент rr обозначает представление записи RR.

rr ::= ([TTL] [<SP> class] <SP> type <SP> RDATA) /
([class] [<SP> TTL] <SP> type <SP> RDATA)

Если поля TTL или class пропущены, то берутся предыдущие точно объявленные значения.

5.4.4.2. Использование специальных символов в контрольном файле

Строка символов (character-string) может быть представлена непрерывным набором символов без пробелов внутри или произвольным набором символов с пробелами, заключенным в двойные кавычки. Во втором случае, если в исходной строке встречается символ кавычки, перед ним должен быть вставлен символ \.

Возможны следующие специальные символы:

@ используется для указания на текущее основание относительного имени.

\X где X - любой символ, кроме цифры от 0 до 9, используется для квотирования символа X, так что специальный символ X используется в качестве печатного символа, а не специального.

\DDD где каждая D - это цифра, используется для обозначения октета, содержащего значение, соответствующее десятичному числу DDD

( ) (символы круглые скобки) используются для группирования данных, занимающих больше одной линии (то есть внутри которых содержатся символы <CRLF>)

(символ точка с запятой) используется для обозначения начала комментария.

5.4.4.3. Использование контрольного файла для определения зон

При составлении контрольного файла, определяющего зону должны быть учтены следующие моменты:

- Все RR файла должны иметь один и тот же класс
- Должна присутствовать только одна запись SOA RR
- Если есть поддомены, и требуется "клеевая" информация, то она должна присутствовать.

Пример контрольного файла, содержащего одну зону:

```
@  IN SOA    VENERA     Action\.domains (
                        20    ; SERIAL
                        7200  ; REFRESH
                        600   ; RETRY
                        3600000; EXPIRE
                        60)   ; MINIMUM

     NS     A.ISI.EDU.
```

```
    NS    VENERA
    NS    VAXA
    MX    10    VENERA
    MX    20    VAXA

A   A     26.3.0.103

VENERA A      10.1.0 52
    A     128.9.0.32

VAXA  A     10.2.0.27
    A     128.9.0.33
```

5.5. Разрешающая система

5.5.1. Функции разрешающей системы

Разрешающая система используется для организации рекурсивных запросов и кэширования запросов и ответов. В случае, если в состав сервера имен входит разрешающая система, он называется рекурсивным. Если в состав сервера имен входит разрешающая система, разрешающая система выполняет функции интерфейса с клиентом.

Клиентский интерфейс разрешающей системы выполняет три основные функции:

- трансляция имени в адрес узла
- трансляция адреса в имя узла
- функция общего поиска

В качестве ответа клиенту разрешающая система может выдать:

- одну или несколько RR
- ошибку имени NE
- ошибку "данные не найдены"

В дополнение к своим собственным ресурсам, разрешающая система может также иметь разделяемый доступ к доменам, обслуживаемым локальным сервером имен. Это дает разрешающей системе преимущество более быстрого доступа, но необходимо следить, чтобы кэшированная информация не подменяла собой истинные данные зоны. В данном разделе под термином "локальная информация" понимается информация, содержащаяся в кэше и разделяемых зонах. Авторитетным данным всегда отдается предпочтение перед кэшированными данными в случае, если присутствуют и те и другие.

5.5.2. Алгоритм работы разрешающей системы.

5.5.2.1. Шаг 1. Проверить, есть ли ответ на запрос в локальной информации. Если есть, выдать ответ клиенту. Поиск происходит в кэшированных данных. Если данные найдены, они предполагаются подходящими для нормального использования. Разрешающие системы могут форсированное игнорирование кэшированных данных по запросу клиента.

5.5.2.2. Шаг 2. Найти наилучшие серверы для продолжения запроса. Поиск серверов имен и занесение их в SLIST. Сначала в локально доступных NS RR серверов имен, ищется SNAME, затем имя родительского домена SNAME, затем прародительского и т.д. до корня. Если поиск заканчивается неудачей, разрешающая система инициализирует SLIST из SBELT.

5.5.2.3. Шаг 3 Отправить запросы и ждать, пока не придет ответ хотя бы на один из них.

5.5.2.4. Шаг 4 Анализировать ответ:

если ответ удовлетворяет запросу или содержит ошибку имени, данные кэшируются и возвращаются клиенту

если ответ содержит лучшую ссылку на другие серверы, кэшировать ссылочную информацию и перейти к шагу 2

если ответ показывает CNAME и не удовлетворяет запросу, кэшировать CNAME, заменить SNAME на каноническое имя из CNAME RR и перейти к шагу 1

если ответ показывает ошибку сервера или другую ненормальную информацию, удалить имя сервера из SLIST и вернуться к шагу 3.

5.5.3. Кэширование отрицательных ответов.

Режим работы DNS с кэшированием отрицательных ответов с установленным TTL является необязательным. Данная возможность является особенно важной в системах, реализующих сокращенные имена (naming shorthands) и использующих списки поиска, так как может случится, что для популярного сокращения в конце списка поиска потребуется суффикс, и таким образом будет генерироваться множество ошибок имен там, где это используется.

5.5.4. Работа разрешающей системы с псевдонимами

При попытке разрешить отдельный запрос разрешающая система может обнаружить, что имя, указанное в запросе является псевдонимом. Возможно, что наличие псевдонима должно быть указано клиенту. Поэтому, хотя в большинстве случаев разрешающая система при нахождении псевдонима просто запускает новый запрос с полученным каноническим именем, при выполнении функции общего поиска разрешающая система не должна прослеживать псевдоним, если оно удовлетворяет условиям запроса.

5.5.5. Рекурсивный режим работы сервера

Рекурсивный режим работы сервера имен является необязательным. Рекурсивный режим работы сервера возможен тогда, когда в сервере реализована разрешающая система. Рекурсивный режим должен использоваться только в случаях, когда клиент и сервер имен согласны его использовать. Согласование рекурсивного режима происходит следующим образом:

- во всех ответах сервера устанавливается в "1" бит RA - рекурсия доступна.
- запрос клиента содержит бит RD - рекурсия желательна.

Если установлены оба бита RA и RD, то при необходимости используется режим рекурсии.

Возможны следующие варианты ответов:

1. Рекурсия разрешена

- ответ на запрос, возможно предшествуемый одной или двумя CNAME RR
- ошибка имени, указываемая, что имя не существует. Может включать CNAME RR, указывающую, что запрос содержал псевдоним, для которого не существует канонического имени.
- индикация временной ошибки

2. Рекурсия запрещена

- ошибка имени, указывающая, что имя не существует
- индикация временной ошибки
- некоторая комбинация:

RR, отвечающих на запрос вместе с информацией, являются ли эти RR кэшированными

Ссылка на сервер имен

- дополнительные RR, являющиеся по мнению сервера полезными запросчику

ПРИЛОЖЕНИЕ 5

ТЕХНИЧЕСКИЕ ТРЕБОВАНИЯ К ТЕХНИЧЕСКИМ СРЕДСТВАМ СЛУЖБЫ ДОСТУПА К ИНФОРМАЦИОННЫМ РЕСУРСАМ ПО ПРОТОКОЛУ HTTP

1. ОБЛАСТЬ ПРИМЕНЕНИЯ

Настоящее приложение описывает технические требования к ТС службы доступа к информационным ресурсам а по протоколу HTTP в соответствии с RFC 2068 [14].
В приложении приведены операции взаимодействия клиента с различными типами ресурсов сети передачи данных, доступ к которым осуществляется посредством сервера HTTP. Определяются операции получения клиентом содержимого ресурса, согласования представления ресурса, передачи информации клиента на адрес ресурса, идентификации клиента.

Не все функции, содержащиеся в данном приложении, обязательны для ТС службы доступа к информационным ресурсам по протоколу HTTP, но если они выполняются , то их реализация должна соответствовать настоящему приложению.

2. ФУНКЦИОНАЛЬНЫЕ ТРЕБОВАНИЯ К СЕРВЕРУ HTTP

2.1. Соединения

2.1.1. Протокол нижнего уровня

Сообщения HTTP должны передаваться с использованием виртуального соединения TCP.

2.1.2. Установление соединения и закрытие соединения

При обмене сообщениями HTTP соединение должно инициироваться стороной клиента. Если сервер поддерживает стойкие соединения (см. п. 5.7.), то перед закрытием соединения он должен в сообщение ответа включить поле Connection с меткой close. Если сервер поддерживает стойкие соединения, то при получении запроса с полем Connection со значением close, он должен ответить на данный запрос и закрыть соединение.

2.1.3. Если сервер поддерживает стойкие соединения, он должен обеспечивать функцию перекачки сообщений клиента в соответствии с п. 5.7.1.1.2.

3. ПЕРЕЧЕНЬ И СТРУКТУРА СООБЩЕНИЙ ПРОТОКОЛА HTTP

3.1. Требования к элементам сообщений

3.1.1. В сообщении HTTP должен использоваться формат адреса URI в соответствии с п. 5.2.2.

3.1.2. В сообщении HTTP должен использоваться один из форматов представления даты, приведенных в п 5.2.3.1

3.1.3. Промежутки времени в полях сообщения HTTP должны указываться в секундах в виде десятичного числа.

3.2. Типы информации

3.2.1. Процедура согласования содержимого

3.2.1.1. Если сервер поддерживает управляемую сервером процедуру согласования содержимого, он должен использовать поле Vary для указания пределов варьирования представления ресурса. При осуществлении выбора представления сервер должен использовать информацию из полей Accept, Accept-Charset, Accept-Encoding, Accept-Language, Accept-Ranges в соответствии с п. 5.12.1., 5.12.2., 5.12.3., 5.12.4., 5.12.5 .

3.2.1.2. Если сервер поддерживает управляемую клиентом процедуру согласования содержимого, он должен включить перечень возможных представлений ресурса в поле Alternates.

3.2.2. Если сервер отправляет ответ с типом информации, отличным от application/octet-stream, он должен указать данный тип в поле Content-Type. Обозначение типа должно соответствовать RFC 2048 [15].

3.2.3. Если к телу сообщения применяется преобразование, тип преобразования должен быть указан в поле Transfer-Encoding в соответствии с п. 5.12.40.

3.3. Запросы

3.3.1. Структура запросов протокола HTTP должна соответствовать п. 5.3.

3.3.2. В сервере должна быть реализована обработка запросов методами GET, HEAD.

3.3.2.1. Если в сервере реализована обработка условных запросов методами GET, HEAD, то для задания условия должны использоваться поля заголовка запроса:

If-Modified-Since;
If-Match;
If-None-Match;
If-Range;
If-Unmodified-Since.

При этом содержание полей должно обрабатываться согласно п. 5.12.24, 5.12.25, 5.12.26, 5.12.27, 5.12.28.

3.3.2.2. Если в сервере реализована обработка запросов диапазонов методом GET, то для задания диапазона должно использоваться поле Range. Содержимое поля Range должно интерпретироваться в соответствии с п. 5.12.36.

3.3.3. Для предоставления клиенту возможности запрашивать информацию о параметрах соединения с запрашиваемым URI должен использоваться метод OPTIONS (раздел 5.8.2).

3.3.4. Для предоставления клиенту возможности передачи на ресурс с указанным URI дополнительной информации в виде сущности должен использоваться метод POST (п. 5.8.5).

3.3.5. Для предоставления клиенту возможности сохранения на сервере сущности под указанным URI должен использоваться метод PUT (п. 5.8.6). Если в результате запроса клиента сервер создал новый ресурс, он должен выдать ответ 201. В случае невозможности сохранить сущность под указанным URI сервер должен выдать ответ 301.

3.3.6. Для предоставления клиенту возможности удаления ресурса с указанным URI должен использоваться метод DELETE (п. 5.8.7).

3.3.7. Метод TRACE используется для организации удаленного шлейфа (п. 5.8.8). Конечному получателю запроса следует направить полученное сообщение обратно клиенту в качестве тела сущности. При этом клиент должен выдать сообщение 200. Конечный получатель - это либо первоначальный сервер, либо первый прокси или шлюз для получения значения 0 в ответ. Запрос TRACE не должен включать сущность.

3.4. Ответы

3.4.1. Структура ответов сервера должна соответствовать п. 5.5.

3.4.2. Сервер должен поддерживать ответы с кодами статуса, приведенные в табл. 1.

Таблица 1

Ответы с кодами статуса.

Код	Описание
100	Продолжить
101	Коммутируемые протоколы
200	OK
201	Создан
202	Принят
203	Ненадежная информация
204	Нет содержимого
205	Переустановить содержимое
206	Частичное содержимое
300	Много выборов
301	Перемещен на постоянный срок
302	Временно перемещен
303	См. другие
304	Не изменен
305	Используй прокси
400	Плохой запрос
401	Неавторизован
402	Требуется оплата
403	Запрещено
404	Не найдено
405	Метод не разрешен
406	Неприменим
407	Требуется идентификация на прокси
408	Тайм-аут запроса
409	Конфликт
410	Ушел
411	Требуется длина
412	Ошибка предварительной обработки
413	Сущность запроса слишком велика
414	URI запроса слишком велико
415	Тип информации не поддерживается
500	Внутренняя ошибка сервера
501	Не реализовано
502	Плохой шлюз
503	Служба недоступна
504	Тайм-аут шлюза
505	Версия HTTP не поддерживается

3.4.3. При генерации ответа сервер должен в поле Etag устанавливать значение, являющееся уникальным среди всех сущностей данного ресурса.

3.4.4. В ответе сервера обязательно должно присутствовать поле Age, значение которого должно вычисляться согласно п. 5.12.6.

3.4.5. В ответе сервера обязательно должно присутствовать поле Date, содержащее дату и время генерации сообщения сервером.

3.5. Взаимодействие клиента и сервера

3.5.1. Сервер не должен выдавать ответ 100 клиенту с версией HTTP ниже 1.1.

4. ТРЕБОВАНИЯ К РЕАЛИЗАЦИИ СЕРВЕРА HTTP

4.1. Требования к функциям кэша

4.1.1. Если в сервере реализованы функции кэша, они должны соответствовать п. 5.11.

4.1.2. Ответы на запрос с методом OPTIONS не должны кэшироваться.

4.1.3. Если кэш выдает устаревший ответ, в него должно быть включено предупреждение 10 (см. п. 5.12.45, табл. 3).

4.1.4. Если кэш не может провести проверку актуальности ответа, в данный ответ должно быть включено предупреждение 11 (см. п. 5.12.45, табл. 3).

4.1.5. Если кэш применяет преобразование содержимого тела сообщения, изменяющее его кодировку или тип информации, он должен включить в сообщение предупреждение 14 (см. п. 5.12.45, табл. 3).

4.1.6. Кэш должен обрабатывать ответы как устаревшие, если они имеют значение поля Expires более позднее, чем текущая дата или, если формат поля Expires не соответствует формату даты по п. 3.1.2. настоящих Технических требований.

4.2. Требования к функциям сервера прокси

4.2.1. Если в сервере реализованы функции сервера прокси, они должны соответствовать п. 5.11.

4.2.2. Сервер прокси не должен устанавливать стойкие соединения с клиентами версии HTTP/1.0. и ниже.

4.2.3. Если через прокси проходит ответ на запрос с методом OPTIONS, прокси должен исправить содержимое полей ответа, а также добавить или удалить поля ответа в соответствии с поддерживаемыми им возможностями.

4.2.4. Если сервер прокси не выполняет функции межсетевого экрана, он должен добавлять в поле Via проходящих через него сообщений свой идентификатор в соответствии с п. 5.12.44. Если сервер прокси выполняет выполняет функции межсетевого экрана, то он должен иметь сертификат Гостехкомиссии России.

4.2.5. Если в прокси реализована процедура идентификации клиента, она должна выполняться с использованием полей заголовка Proxy-Authenticate и Proxy-Authorization в соответствии с п. 5.12.33 и 5.12.34.

4.2.6. Прокси должен добавлять в сообщение запроса поле Host, если данное поле отсутствует в сообщении.

4.2.7. Прокси, получивший сообщение с методом TRACE и полем Max-Forwards=0, не должен направлять данное сообщение, а должен выдать ответ 200 с данным сообщением, включенным в сущность ответа.

4.3. Идентификация доступа

4.3.1. Для начала процедуры идентификации должен использоваться ответ 401 с полем WWW-Authenticate. В процедуре идентификации должно использоваться содержимое поля запроса Authorization. Если сервер не принимает сообщение клиента с полем Authorization, он снова должен выдать ответ 401.

4.3.2. Если сервер поддерживает первичную схему идентификации доступа, ее реализация должна соответствовать п. 5.9.1.

4.4. Согласование протокола

Если сервер поддерживает процедуру согласования протокола, данная процедура должна быть реализована в соответствии с п. 5.12.41.

5. ОПИСАНИЕ ПРОТОКОЛА HTTP

5.1. Базовые определения

Спецификация синтаксиса запросов и ответов HTTP приводится в расширенной форме Наура-Бекуса соответствующей RFC-822 [2].

Базовые определения:

```
OCTET      ::= <8 бит любых данных>
CHAR       ::= <любой символ US-ASCII (октеты от 0 до 127)>
UPALPHA    ::= <любая прописная буква US-ASCII "A".."Z">
LOALPHA    ::= <любая строчная буква US-ASCII "a".."z">
ALPHA      ::= UPALPHA | LOALPHA
DIGIT      ::= <любая цифра US-ASCII "0".."9">
CTL        ::= <любой управляющий символ US-ASCII
               (октеты от 0 до 31) и DEL (127)>
CR         ::= <US-ASCII CR, возврат каретки (13)>
LF         ::= <US-ASCII LF, пропуск линии (10)>
SP         ::= <US-ASCII SP, пробел (32)>
```

```
HT          ::= <US-ASCII HT, горизонтальная табуляция (9)>
<">         ::= <US-ASCII двойные кавычки (34)>
CRLF        ::= CR LF
```

Заголовок HTTP может быть разбит на несколько строк. 2-я и далее строка заголовка должны начинаться с пробела или табуляции.

```
LWS         ::= [CRLF] 1*( SP | HT )
```

Слова *TEXT могут содержать символы набора, отличного от ISO 8859-1[16] только в случае, если они закодированы в соответствии с RFC 2047[17].

```
TEXT        ::= <любой OCTET, кроме октетов CTL, но включая LWS>


HEX         ::= "A" | "B" | "C" | "D" | "E" | "F"
              | "a" | "b" | "c" | "d" | "e" | "f" | DIGIT

token       ::= 1*<any CHAR except CTLs or tspecials>

tspecials   ::= "(" | ")" | "<" | ">" | "@"
              | "," | ";" | ":" | "\" | <">
              | "/" | "[" | "]" | "?" | "="
              | "{" | "}" | SP | HT
```

Если в значении поля заголовка содержатся специальные символы, эти специальные символы должны содержаться в строке в кавычках.

```
comment     ::= "(" *( ctext | comment ) ")"
ctext       ::= <любой TEXT кроме "(" и ")">
```

Строка текста, заключенная в двойные кавычки ("строка в кавычках"), обрабатывается как единое слово.

```
        quoted-string  ::= ( <"> *(qdtext) <"> )
qdtext      ::= <any TEXT except <">>
```

Конструкция квотирования "\<символ>" может использоваться только внутри "строки в кавычках" и комментария.

```
quoted-pair   ::=  "\" CHAR
```

5.2. Элементы протокола

5.2.1. Версия HTTP

Версия содержится в поле HTTP-Version в первой строке сообщения.

```
HTTP-Version = "HTTP" "/" 1*DIGIT "." 1*DIGIT
```

Номер DIGIT не должен начинаться с нулей

Прокси и шлюзы никогда не должны посылать сообщения с номером версии большим, чем номер их собственной версии. Если прокси (шлюз) получает ответ с номером версии большим, чем номер версии этого прокси (шлюза), он должен либо изменить сообщение так, чтобы номер версии был понижен, либо ответить сообщением об ошибке, либо работать как тоннель. Если версия запроса, направленного к прокси, меньше версии прокси, то он может перенаправить запрос, повысив версию, но в ответе на данный запрос он должен выдать, используя ту же версию, что и первичный запрос.

5.2.2. Универсальный идентификатор ресурса (URI)

5.2.2.1. Синтаксис

URI может быть представлен в абсолютной форме или относительной к некоторой известной базовой URI форме. Абсолютный URI всегда начинается с имени схемы, за которым следует символ двоеточие.

```
URI          ::= ( absoluteURI | relativeURI ) [ "#" fragment ]

absoluteURI   ::= scheme ":" *( uchar | reserved )

relativeURI   ::= net_path | abs_path | rel_path

net_path      ::= "//" net_loc [ abs_path ]
abs_path      ::= "/" rel_path
rel_path      ::= [ path ] [ ";" params ] [ "?" query ]

path          ::= fsegment *( "/" segment )
fsegment      ::= 1*pchar
segment       ::= *pchar

params        ::= param *( ";" param )
param         ::= *( pchar | "/" )

scheme        ::= 1*( ALPHA | DIGIT | "+" | "-" | "." )
net_loc       ::= *( pchar | ";" | "?" )

query         ::= *( uchar | reserved )
fragment      ::= *( uchar | reserved )

pchar         ::= uchar | ":" | "@" | "&" | "=" | "+"
uchar         ::= unreserved | escape
unreserved    ::= ALPHA | DIGIT | safe | extra | national

escape        ::= "%" HEX HEX
reserved      ::= ";" | "/" | "?" | ":" | "@" | "&" | "=" | "+"
extra         ::= "!" | "*" | "'" | "(" | ")" | ","
safe          ::= "$" | "-" | "_" | "."
unsafe        ::= CTL | SP | <"> | "#" | "%" | "<" | ">"
```

national ::= <любой OCTET кроме ALPHA, DIGIT, reserved, extra, safe, и unsafe>

Если длина URI превышает максимальную длину, обрабатываемую сервером, сервер должен выдать ответ статуса 414.

5.2.2.2. HTTP URL

URL - универсальный указатель расположения ресурса. URL - тип URI, со схемой http.

http_URL ::= "http:" "//" host [":" port] [abs_path]

host ::= <Разрешенное имя домена Internet или адрес IP в форме десятичных чисел, разделенных точками, в соответствии с п. 2.1. RFC 1123 [18])

port ::= *DIGIT

Если port не указан, предполагается порт по умолчанию 80. Если в запросе не указан abs_path, вместо него должен присутствовать "/".

5.2.2.3. При сравнении двух URI с целью выяснения идентичности, клиент должен проводить пооктетное сравнение с учетом регистра всех символов URI. При этом:
- если порт не указан, он считается эквивалентным порту по умолчанию;
- при сравнении имен узлов регистр символов не должен учитываться;
- при сравнении схем регистр не должен учитываться;
- пустой абсолютный путь abs_path эквивалентен значению "/" abs_path.

Символы, не входящие в группы "reserved" и "unsafe" эквивалентны их коду, представленному как ""%" HEX HEX".

5.2.3. Формат даты и времени

5.2.3.1. Полная дата

Существует три допустимых формы представления даты/времени:

Sun, 06 Nov 1994 08:49:37 GMT ; RFC 822 [2], с учетом изменений,
; внесенных RFC 1123 [18]
Sunday, 06-Nov-94 08:49:37 GMT ; RFC 1036 [19]
Sun Nov 6 08:49:37 1994 ; формат ANSI C asctime()

Время и дата, указанные в метках даты/времени, должны соответствовать Среднему Гринвичскому времени

HTTP-date ::= rfc1123-date | rfc850-date | asctime-date
; rfc850 - rfc850[20]

```
rfc1123-date ::= wkday "," SP date1 SP time SP "GMT"
rfc850-date  ::= weekday "," SP date2 SP time SP "GMT"
asctime-date ::= wkday SP date3 SP time SP 4DIGIT

date1        ::= 2DIGIT SP month SP 4DIGIT
                 ; день месяц год (Например, 02 Jun 1982)
date2        ::= 2DIGIT "-" month "-" 2DIGIT
                 ; день-месяц-год (Например, 02-Jun-82)
date3        ::= month SP ( 2DIGIT | ( SP 1DIGIT ))
                 ; месяц день (Например, Jun  2)

time         ::= 2DIGIT ":" 2DIGIT ":" 2DIGIT
                 ; 00:00:00 - 23:59:59

wkday        ::= "Mon" | "Tue" | "Wed"
               | "Thu" | "Fri" | "Sat" | "Sun"

weekday      ::= "Monday" | "Tuesday" | "Wednesday"
               | "Thursday" | "Friday" | "Saturday" | "Sunday"

month        ::= "Jan" | "Feb" | "Mar" | "Apr"
               | "May" | "Jun" | "Jul" | "Aug"
               | "Sep" | "Oct" | "Nov" | "Dec"
```

5.2.3.2. Период в секундах

Некоторые поля заголовка HTTP позволяют указывать значения времени в виде целого числа секунд в десятичной форме, прошедших с момента получения сообщения.

```
delta-seconds ::= 1*DIGIT
```

5.2.4. Наборы символов

Термин "набор символов" в протоколе HTTP аналогичен такому же термину, определенному в протоколе MIME.

Набор символов указывает на метод преобразования последовательности октетов в последовательность символов.

```
charset ::= token
```

где значение token должно соответствовать RFC 1700 [9].

5.2.5. Кодирование содержимого

Значения способа кодирования содержимого указывают на преобразование кодировки (например, сжатие), которое применялось, либо может быть применено к сущности.

```
content-coding ::= token
```

Определены следующие значения token: gzip (GNU zip согласно RFC 1952 [21]) и compress.

5.2.6. Кодирование при передаче

Значения кодирования при передаче указывают на преобразование кодировки, которое применялось, либо может быть применено к телу сущности с целью обеспечения "безопасной транспортировки" по сети. Кодирование при передаче является свойством сообщения, а не сущности.

```
transfer-coding ::= "chunked" | transfer-extension
transfer-extension ::= token
```

Порционное кодирование (chunked encoding) изменяет тело сообщения таким образом, что оно передается как последовательность порций. Каждая порция имеет свой индикатор размера, за которым следует необязательный раздел, содержащий поля заголовка сущности. Таким образом кодирование при передаче позволяет передавать сообщения с динамически генерируемым содержимым.

```
Chunked-Body   ::= *chunk
                   "0" CRLF
                   footer
                   CRLF

chunk          ::= chunk-size [ chunk-ext ] CRLF
                   chunk-data CRLF

hex-no-zero    ::= <HEX excluding "0">

chunk-size     ::= hex-no-zero *HEX
chunk-ext      ::= *( ";" chunk-ext-name [ "=" chunk-ext-value ] )
chunk-ext-name ::= token
chunk-ext-val  ::= token | quoted-string
chunk-data     ::= chunk-size(OCTET)

footer         ::= *entity-header
```

5.2.7. Типы информации

Для обеспечения открытого определения и согласования типа информации используются поля заголовка Content-Type и Accept.

```
media-type     ::= type "/" subtype *( ";" parameter )
type           ::= token
subtype        ::= token
parameter      ::= attribute "=" value
attribute      ::= token
value          ::= token | quoted-string
```

Между элементами type и subtype, а также между parameter и value не должны использоваться символы LWS в качестве разделителя.
Значения media-type должны соответствовать RFC 2048 [15]

5.2.7.1. Канонические представления информации

Для каждого типа информации зарегистрирована каноническая форма представления. При пересылке в теле сообщения HTTP-информация должна быть представлена в канонической форме.

5.2.7.1.1. Тип text.

В качестве разделителя линии в теле сообщения при типе media-type text могут использоваться отдельные символы CR, LF и пара символов CRLF.
Если не указан набор символов, по умолчанию должен использоваться ISO-8859-1 [16].

5.2.7.2. Если сообщение содержит несколько тел с различными типами информации, все части сообщения должны иметь формат MIME. В отличие от MIME эпилог (epilogue) не должен передаваться.

5.2.8. Метки продуктов

Метки продуктов позволяют взаимодействующим по сети приложениям определить название и версию программного обеспечения, используемую на другой стороне.

```
product         ::= token ["/" product-version]
product-version ::= token
```

Например:

```
User-Agent: CERN-LineMode/2.15 libwww/2.17b3
Server: Apache/0.8.4
```

5.2.9. Параметр качества содержимого

Процедура согласования содержимого использует короткие числа с плавающей запятой для указания относительной важности (веса) различных параметров согласования. Веса приведены к нормализованной форме в диапазоне от 0 до 1, где 0 - минимальный вес, а 1 - максимальный. Вес не должен содержать более 3-х цифр после десятичной точки.

```
qvalue         ::= ( "0" [ "." 0*3DIGIT ] )
                 | ( "1" [ "." 0*3("0") ] )
```

5.2.10. Теги языка

Теги языка указывают на естественный язык, используемый для передачи информации между людьми. Синтаксис и допустимое содержание тега языка должны соответствовать RFC 1766 [22].

```
language-tag  ::= primary-tag *( "-" subtag )

primary-tag   ::= 1*8ALPHA
```

subtag ::= 1*8ALPHA

Первый двухсимвольный тег primary-tag - это аббревиатура языка в соответствии с ISO 639 [23], а первый двухбуквенный subtag - это код страны по ISO 3166 [24].

5.2.11. Теги сущности

Теги сущности используются для сравнения двух или более сущностей, поступивших от одного запрашиваемого ресурса.

entity-tag ::= [weak] opaque-tag

weak ::= "W/"
opaque-tag ::= quoted-string

Теги сущности должны быть уникальны для всех версий всех сущностей, относящихся к отдельному ресурсу.

5.2.12. Блоки диапазонов

HTTP 1.1 позволяет клиенту затребовать, чтобы только часть (диапазон) сущности ответа была включена в ответ. Сущность может быть разбита на поддиапазоны в соответствии с различными структурными блоками.

range-unit ::= bytes-unit | other-range-unit

bytes-unit ::= "bytes"
other-range-unit ::= token

Единственным обязательным для реализации блоком является bytes-unit.

5.3. Сообщение HTTP

5.3.1. Типы сообщений

Сообщения разделяются на запросы клиента к серверу и ответы сервера клиенту.

HTTP-message ::= Request | Response

Запросы и ответы используют общий формат сообщений RFC 822 [2] для передачи сущностей (полезной нагрузки сообщения). Оба типа сообщения состоят из начальной строки, одной или более строк заголовка, пустой строки, указывающей на конец заголовка, и необязательного тела сообщения.

generic-message ::= start-line
*message-header
CRLF
[message-body]

start-line ::= Request-Line | Status-Line

Сервер должен игнорировать пустые строки, поступающие перед Request-Line.

5.3.2. Заголовки сообщений

Поля заголовка HTTP, включая поля общего заголовка, заголовка запроса, заголовка ответа и заголовка сущности, должны соответствовать формату п. 3.1. RFC 822 [2]. Поля заголовка могут занимать несколько строк, но строки, следующие за первой, должны начинаться с 1*(SP | HT)

message-header ..= field-name ":" [field-value] CRLF

field-name ::= token
field-value ::= *(field-content | LWS)

field-content ::= <октеты, составляющие значение поля и состоящие либо из *TEXT, либо из комбинаций token, tspecials и quoted-strings>

Порядок полей в заголовке не имеет значения. Но прокси не должен изменять порядок полей в перенаправляемом сообщении.

5.3.3. Тело сообщения

Тело сообщения используется для переноса тела сущности и отличается от тела сущности только когда применено кодирование при передаче.

message-body ::= entity-body | <entity-body закодированное в соответствии с Transfer-Encoding>

Ответы 1xx (информационные), 204 (нет содержимого) и 304 (не изменен) не должны содержать тело сообщения. Все остальные ответы должны содержать тело, хотя возможно и нулевой длины.

На наличие тела сообщения указывает присутствие полей Content-Length и Transfer-Encoding.

5.3.4. Длина сообщения

При включении тела сообщения в сообщение, длина (конец) этого тела сообщения определяется следующим образом:

1. Любой ответ, который не должен содержать тело сообщения, всегда заканчивается первой пустой строкой после заголовка, независимо от того, присутствуют ли поля заголовка сущности.

2. Если присутствует поле заголовка Transfer-Encoding и указывает на применение порционного кодирования, то длина определяется порционным кодированием.

3. Если присутствует поле Content-Length, его значение определяет длину тела сообщения в байтах.

4. Если сообщение использует тип информации "multipart/byteranges", который является самоограничивающимся, то этот тип информации определяет длину тела сообщения. Этот тип информации должен использоваться сервером только тогда, когда в запросе присутствует заголовок Range с несколькими определителями byte-range, который показывает, что клиент может обработать ответ "multipart/byteranges".

5. Закрытием соединения (только для тела ответа).

Сообщения не должны содержать одновременно поле Content-Length и поля порционного кодирования. В случае получения подобного сообщения значение Content-Length должно игнорироваться.

5.3.5. Общие поля заголовка

Данные поля используются и в запросах, и в ответах и применяются к сообщению в целом, а не к передаваемым сущностям.

```
general-header ::= Cache-Control
                 | Connection
                 | Date
                 | Pragma
                 | Transfer-Encoding
                 | Upgrade
                 | Via
```

Нераспознанные поля заголовка должны обрабатываться как поля заголовка сущности.

5.4. Запрос

Сообщение запроса содержит в первой строке сообщения метод, применяемый к ресурсу, идентификатор ресурса и версию используемого протокола.

```
Request      ::= Request-Line
               *( general-header
                | request-header
                | entity-header )
               CRLF
               [ message-body ]
```

5.4.1. Request-Line (Строка запроса)

Request-Line начинается с метки метода, за которым следует Request-URI и версия протокола, заканчивающаяся CRLF. Элементы разделены символами SP. Появление символов CR или LF, кроме заключительной последовательности CRLF, запрещено.

```
Request-Line   ::= Method SP Request-URI SP HTTP-Version CRLF
```

5.4.1.1. Method (Метод)

Регистр символа слова, обозначающего метод, является существенным.

```
Method       ::= "OPTIONS"
               | "GET"
               | "HEAD"
               | "POST"
               | "PUT"
               | "DELETE"
               | "TRACE"
```

| extension-method

extension-method = token

Поддержка методов GET и HEAD является обязательной для реализации сервера.

5.4.1.2. Request-URI (URI запроса)

Request-URI определяет ресурс, к которому применяется запрос. Символ "*" показывает, что запрос должен применяться не к отдельному ресурсу, а к серверу вцелом.

Request-URI ::= "*" | absoluteURI | abs_path

5.4.2. Идентификация ресурса

Если сервер-источник различает ресурсы на основе запрашиваемого узла (например, поддерживает виртуальные имена узлов), он должен использовать следующие правила для определения запрашиваемого источника.

5.4.2.1. Если Request-URI является absoluteURI, тогда имя узла является частью Request-URI, и поле Host должно игнорироваться.

5.4.2.2. Если Request-URI не является absoluteURI, и запрос включает поле заголовка Host, узел определяется значением Host.

5.4.2.3. Если узел, определенный с помощью правил 5.4.2.1 и 5.4.2.2., не является действительным узлом на сервере, сервер должен выдать ответ 400.

5.4.3. Поля заголовка запроса.

Поля заголовка запроса позволяют клиенту передать дополнительную информацию о запросе и о самом клиенте.

request-header ::= Accept
| Accept-Charset
| Accept-Encoding
| Accept-Language
| Authorization
| From
| Host
| If-Modified-Since
| If-Match
| If-None-Match
| If-Range
| If-Unmodified-Since
| Max-Forwards
| Proxy-Authorization
| Range
| Referer
| User-Agent

5.5. Ответ

После получения и обработки сообщения запроса сервер отвечает сообщением ответа:

```
Response      ::= Status-Line
                *( general-header
                 | response-header
                 | entity-header )
                CRLF
                [ message-body ]
```

5.5.1. Status-Line (Строка статуса)

Строка статуса содержит версию протокола и дополнительную текстовую фразу. Строка статуса не может разрываться символами CR или LF.

Status-Line ::= HTTP-Version SP Status-Code SP Reason-Phrase CRLF

5.5.1.1. Status-Code (код статуса)

```
Status-Code  ::= "100"  ; Продолжить
               | "101"  : Коммутируемые протоколы
               | "200"  ; OK
               | "201"  : Создан
               | "202"  : Принят
               | "203"  ; Ненадежная информация
               | "204"  ; Нет содержимого
               | "205"  ; Переустановить содержимое
               | "206"  ; Частичное содержимое
               | "300"  ; Много выборов
               | "301"  : Перемещен на постоянный срок
               | "302"  ; Временно перемещен
               | "303"  ; См. другие
               | "304"  ; Не изменен
               | "305"  ; Используй прокси
               | "400"  ; Плохой запрос
               | "401"  : Неавторизован
               | "402"  ; Требуется оплата
               | "403"  ; Запрещено
               | "404"  ; Не найдено
               | "405"  ; Метод не разрешен
               | "406"  ; Неприменим
               | "407"  ; Требуется идентификация на прокси
               | "408"  ; Тайм-аут запроса
               | "409"  ; Конфликт
               | "410"  ; Ушел
               | "411"  ; Требуется длина
               | "412"  ; Ошибка предварительной обработки
               | "413"  ; Сущность запроса слишком велика
               | "414"  ; URI запроса слишком велико
               | "415"  ; Тип информации не поддерживается
```

| "500" . Внутренняя ошибка сервера
| "501" : Не реализовано
| "502" . Плохой шлюз
| "503" . Служба недоступна
| "504" . Тайм-аут шлюза
| "505" : Версия HTTP не поддерживается
| extension-code

extension-code ::= 3DIGIT
Reason-Phrase ::= *<TEXT, включая CR, LF>

Первая цифра поля Status-Code определяет класс ответа. Возможные значения приведены в табл. 2.

Таблица 2

Первая цифра поля Status-Code.

1xx	Информационный	Запрос получен, продолжаю процесс
2xx	Успех	Команда получена, понята и принята
3xx	Перенаправление	Должны быть предприняты дальнейшие действия для завершения запроса
4xx	Ошибка клиента	Запрос содержит синтаксическую ошибку или не может быть выполнен
5xx	Ошибка сервера	Сервер не может выполнить очевидно правильный запрос.

От реализаций HTTP не требуется различать значения всех указанных кодов статуса. Но реализации должны различать класс статуса (первую цифру кода). Нераспознанный ответ любого класса должен обрабатываться как код x00 данного класса, но не должен кэшироваться.

5.5.2. Поля заголовка ответа

response-header ::= Age
| Location
| Proxy-Authenticate
| Public
| Retry-After
| Server
| Vary
| Warning
| WWW-Authenticate

Поля заголовка ответа позволяют серверу передать дополнительную информацию, которая не может быть передана в строке статуса Status-Line.

5.6. Сущность

Сущность состоит из полей заголовка сущности и необязательного тела сущности.

5.6.1. Поля заголовка сущности

```
entity-header   = Allow
                | Content-Base
                | Content-Encoding
                | Content-Language
                | Content-Length
                | Content-Location
                | Content-MD5
                | Content-Range
                | Content-Type
                | ETag
                | Expires
                | Last-Modified
                | extension-header

extension-header ::= message-header
```

5.6.2. Тело сущности

Формат и кодирование тела сущности определяется значением полей заголовка сущности.

```
entity-body   ::= *OCTET
```

5.6.2.1. Тип

Тип данных тела сущности определяется полями заголовка Content-Type и Content-Encoding.

```
entity-body ::= Content-Encoding( Content-Type( data ) )
```

Поле Content-Type определяет тип информации.
Поле Content-Encoding должно присутствовать обязательно.
Если поле Content-Type отсутствует, по умолчанию принимается тип информации "application/octet-stream".

5.6.2.2. Длина тела сущности - это длина тела сообщения после удаления транспортного кодирования.

5.7. Соединения

5.7.1. Стойкие соединения

Принцип стойких соединений TCP, в отличие от использования отдельных соединений TCP на каждый отдельный цикл запрос клиента - ответ сервера, предполагает использование одного соединения TCP для нескольких обращений к серверу (даже к нескольким URL, если они расположены на одном сервере).

Использование стойких соединений является необязательным.

5.7.1.1. Общее функционирование

В версии 1.1 протокола HTTP стойкие соединения используются по умолчанию. Стойкие соединения предоставляют механизм, по которому клиент и сервер могут выдавать сигнал о закрытии соединения TCP. Для этого используется поле заголовка Connection. После того, как было просигнализировано закрытие, клиент не должен передавать запросы по этому соединению.

5.7.1.1.1. Согласование

Если сервер хочет закрыть соединение, он должен сигнализировать об этом в поле Connection. Если клиент хочет закрыть соединение, он должен сигнализировать об этом в поле Connection.

5.7.1.1.2. Перекачка

Клиент, поддерживающий стойкие соединения, может "перекачивать" свои запросы, то есть высылать несколько запросов, не дожидаясь ответов на каждый запрос. Сервер должен высылать ответы в порядке получения запросов.

5.7.1.2. Прокси

Сервер прокси должен обслуживать сигнализацию стойких соединений раздельно для клиентов и серверов-источников (или других прокси), с которыми у него есть соединения. Прокси не должен устанавливать стойкое соединение с клиентом версии 1.0.

5.7.1.3. Требования к реализации

Прокси, серверы и клиенты должны иметь возможность восстановления информации в случае несинхронизированного закрытия соединения.

Сервер HTTP версии 1.1 при обработке запроса клиента HTTP версии 1.0 (или более раннего) не должен передавать ответ 100. Он должен либо дождаться окончания обработки запроса, либо преждевременно закрыть соединение.

При получении метода в соответствии с разделом 6.8 от клиента HTTP версии 1.1 сервер HTTP версии 1.1 должен либо выдать ответ 100 и продолжить чтение входного потока, либо выдать сообщение об ошибке. При выдаче сообщения об ошибке сервер не должен выполнять запрошенный метод.

5.8. Определения методов

5.8.1. Безопасные и идемпотентные методы

5.8.1.1. Безопасные методы

Безопасными называют методы GET (п. 5.8.3) и HEAD (п. 5.8.4). Они всегда означают действия получения, в которых клиент не отвечает за непредвиденный побочный эффект от их выполнения, в отличие от методов POST (п. 5.8.5), PUT (п. 5.8.6) и DELETE (п. 5.8.7).

5.8.1.2. Идемпотентные методы

Идемпотентными называют методы GET (п. 5.8.3), HEAD (п. 5.8.4), PUT (п 5.8.6) и DELETE (п. 5.8.7), для которых побочный эффект от выполнения нескольких идентичных запросов будет равен побочному эффекту от выполнения одного запроса.

5.8.2. Метод OPTIONS

Метод OPTIONS представляет собой запрос информации о факультативных возможностях соединения, доступных в цепочке запрос/ответ для данного запрашиваемого URI. Данный метод позволяет клиенту определить возможности и (или) требования, связанные с данным ресурсом, а также возможности сервера без выполнения действий над ресурсом.

Если в поле Request-URI установлен символ "*", запрос относится к серверу в целом.

Если запрос OPTIONS проходит через прокси, прокси должен исправить запрос, исключив те опции, которые относятся к возможностям прокси, и которые не могут быть получены через этот прокси.

На запрос с методом OPTIONS сервер должен выдать ответ 200.

5.8.3. Метод GET

Метод GET позволяет получить любую информацию (в форме сущности), указанную полем Request-URI.

Если сообщение запроса содержит поля заголовка If-Modified-Since, If-Match, If-None-Match или If-Range, метод GET называют условным ("conditional GET").

Если сообщение запроса содержит поле заголовка Range, метод GET называют частичным ("partial GET").

5.8.4. Метод HEAD

Метод HEAD идентичен методу GET за исключением того, что сервер не должен возвращать тело сообщения в ответе.

5.8.5. Метод POST

Метод POST используется для выполнения запроса, в результате которого сервер назначения принимает содержащуюся в запросе сущность, как новую придаточную часть ресурса, определяемого Request-URI в строке запроса Request-Line.

5.8.6. Метод PUT

Метод PUT используется для выполнения запроса, в результате которого сущность, содержащаяся в запросе, сохраняется под указанным Request-URL.

Если создан новый ресурс, сервер должен выдать ответ 201.

Получатель запроса не должен игнорировать поля заголовка Content-* (Например, Content-Range), которые он не понимает или не поддерживает, а должен выдать ответ 501.

В случае невозможности сохранения сущности под указанным URI, сервер не должен сохранять сущность под другим URI, а должен выдать ответ 301.

5.8.7. Метод DELETE

Метод DELETE используется для выполнения запроса, в результате которого сервер удаляет ресурс, идентифицируемый Request-URI.
Сервер может не поддерживать данный метод вообще, либо не поддерживать его для отдельных ресурсов и типов ресурсов.

5.8.8. Метод TRACE

Метод TRACE используется для организации удаленного шлейфа для сообщения запроса. Окончательный получатель запроса должен вернуть данный запрос в теле сущности ответа 200.

Запрос TRACE не должен включать сущность.

Ответы на данный запрос не должны кэшироваться.

5.9. Идентификация доступа

Механизм идентификации клиента не обязателен для реализации.
Для указания схемы идентификации используется нечувствительный к регистру символов маркер.

Сообщение ответа 401 используется сервером-источником для начала процедуры идентификации клиента. Данный ответ должен содержать поле заголовка WWW-Authenticate, содержащее как минимум один вызов, применимый к запрошенному ресурсу.

```
auth-scheme    ::= token

auth-param     ::= token "=" quoted-string

challenge      ::= auth-scheme 1*SP realm *( "," auth-param )

realm          ::= "realm" "=" realm-value

realm-value    ::= quoted-string
```

Атрибут realm (область), являющийся нечувствительным к регистру символов, должен присутствовать во всех вызовах процедуры идентификации независимо от схемы. Значение realm-value совместно с каноническим URL корня определяет защищаемую область.

Клиент, желающий выполнить процедуру идентификации, должен после получения ответа 401 или 411 (п. 3.4.2, табл.1) выдать запрос с полем заголовка Authorization. Значение поля Authorization должно состоять из мандатов (credentials), содержащих идентифицирующую информацию о клиенте для указанной в идентификационном вызове защищаемой области.

```
credentials    ::= basic-credentials
                 | auth-scheme #auth-param
```

Если первичный запрос был идентифицирован, те же самые мандаты могут использоваться для всех других запросов, относящихся к данной защищаемой области в течение периода времени, определенного схемой идентификации.

Если сервер не принимает запрос с мандатами, он может выдать ответ 401 (п. 3.4.2, табл.1). При этом ответ 401 должен содержать поле заголовка WWW-Authenticate.

Приложения могут использовать механизм идентификации, отличный от описанного.

Прокси должен быть абсолютно прозрачен для процедуры идентификации клиента.

5.9.1. Первичная схема идентификации

Первичная схема основана на идентификации клиента по идентификатору и паролю. Пример вызова, выдаваемого сервером для начала процедуры идентификации:

WWW-Authenticate: Basic realm="WallyWorld"

Ответный запрос клиента содержит мандат, состоящий из идентификатора и пароля пользователя, разделенных двоеточием и закодированным в кодировке base64.

basic-credentials ::= "Basic" SP basic-cookie

basic-cookie ::= <закодированный в base64 user-pass,
ограниченного до 76 символов на строка>

user-pass ::= userid ":" password

userid ::= *<TEXT кроме ":">

password ::= *TEXT

Пример ответного запроса клиента:

Authorization: Basic QWxhZGRpbjpvcGVuIHNlc2FtZQ==

5.10. Согласование содержимого

Согласование содержимого представляет собой механизм для выбора клиентом наиболее подходящего ему представления ресурса. Любой ответ, содержащий тело сущности, может быть предметом согласования содержимого.
Согласование содержимого может быть двух типов: управляемое сервером и управляемое клиентом.

5.10.1. Управляемое сервером согласование содержимого

В случае управляемого сервером согласования данных сервер на основе данных запроса делает предположение о наилучшем для клиента типе информации для данного ресурса. Для согласования содержимого сервером используются следующие поля заголовка запроса пользователя:

Accept, Accept-Charset, Accept-Encoding, Accept-Language, User-Agent.

В случае управляемого сервером согласования данных сервер-источник должен включать в каждый кэшируемый ответ поле Vary.

5.10.2 Управляемое клиентом согласование содержимого

В случае управляемого клиентом согласования данных выбор наилучшего представления ресурса в ответе выполняется клиентом после получения начального ответа от сервера-источника. Выбор делается на основе списка доступных представлений, включенного в заголовок ответа в поле Alternatives, либо в тело ответа.

5.10.3. Прозрачное согласование

Прозрачное согласование является комбинацией типов согласования управляемого клиентом и управляемого сервером. Прозрачное согласование может выполняться кэшем. Кэш, выполняющий прозрачное согласование, должен включать в заголовок ответа поле Vary.

5.11. Функционирование кэша и прокси

5.11.1. Ответы с кодом статуса 206 не должны кэшироваться.

5.11.2. Кэш и прокси не должны изменять или добавлять ни в запросе ни в ответе поля Content-Location, ETag, Expires, Last-Modified.

5.11.3. В ответе, содержащем директиву Cache-Control со значением no-transform, и в любом запросе кэш и прокси не должны изменять или добавлять следующие поля: Content-Encoding, Content-Length, Content-Range, Content-Type.
Если кэш или прокси изменяют или добавляют эти поля, они должны при этом добавить предупреждение 14 (см. п. 5.12.45, табл. 3), если только в данном сообщении уже нет предупреждения 14.

5.11.4. Выполнение проверки актуальности

Если кэш выполняет запрос проверки актуальности к серверу, и сервер выдает ответ 304, кэш должен составить и отправить ответ клиенту, а также заменить поля заголовков хранящихся в кэше позиций (включая поля, указанные в п. 5.11.2. и 5.11.3.) на значения, полученные в ответе 304.

5.11.5. Кэш, который получает неполный ответ (например, с меньшим количеством байт данных, чем указано в поле Content-Length заголовка), в случае, если он сохраняет данный ответ, должен его обрабатывать как частичный. Частичные ответы могут комбинироваться в кэше. Кэш должен выдавать частичный ответ только используя код статуса 206.

5.11.6. Кэш не должен обрабатывать ответы, в которых в части rel_path URL содержатся символы "?", если только в них не указано точное время истечения.

5.12. Определения полей заголовка

5.12.1. Accept

Поле заголовка запроса Accept определяет типы информации, которые являются приемлемыми для ответа.

```
Accept          ::= "Accept" ":"
                    #( media-range [ accept-params ] )

media-range     ::= ( "*/*"
                    | ( type "/" "*" )
                    | ( type "/" subtype )
                    ) *( ";" parameter )

accept-params   ::= ";" "q" "=" qvalue *( accept-extension )
                    ; параметры указывают относительный фактор качества
                    ; (от 0 до 1)  По умолчанию - 1.

accept-extension ::= ";" token [ "=" ( token | quoted-string ) ]
```

Символ "*" используется для группирования типов информации в диапазоны. *При этом "*/*" означает все типы данных, а "тип/*" - все подтипы данного типа.*

Пример:

```
Accept: text/plain; q=0.5, text/html,
    text/x-dvi; q=0.8, text/x-c
```

Будет означать: предпочтительными типами информации являются text/html и text/x-c, но если таких нет, тогда нужно выслать сущность с типом text/x-dvi. А если и такого нет, тогда выслать text/plain.

5.12.2. Accept-Charset

Поле заголовка запроса Accept-Charset используется для указания наборов символов, приемлемых в ответе:

```
Accept-Charset ::= "Accept-Charset" ":"
        1#( charset [ ";" "q" "=" qvalue ] )
```

Например:

```
Accept-Charset: iso-8859-5, unicode-1-1;q=0.8
```

Если данное поле отсутствует, считается, что для клиента приемлем любой набор символов.

5.12.3. Accept-Encoding

Поле заголовка запроса аналогично полю Accept, но определяет приемлемые методы кодирования содержимого.

```
Accept-Encoding  ::= "Accept-Encoding" ":"
                     #( content-coding )
```

Например:

```
Accept-Encoding: compress, gzip
```

5 12.4 Accept-Language

Поле заголовка запроса аналогично полю Accept, но определяет приемлемые естественные языки, используемые в ответе:

```
Accept-Language .:= "Accept-Language" ":"
                    1#( language-range [ ";" "q" "=" qvalue ] )

language-range  ::= ( ( 1*8ALPHA *( "-" 1*8ALPHA ) ) | "*" )
```

По умолчанию определитель качества q=1.

Пример:

```
Accept-Language: da, en-gb;q=0.8, en;q=0.7
```

означает: я предпочитаю датский, но также приемлемыми являются британский английский и другие типы английского.

5.12.5. Accept-Ranges

Поле ответа Accept-Ranges позволяет серверу указать приемлемые диапазоны в запросах к ресурсу.

```
Accept-Ranges     ::= "Accept-Ranges" ":" acceptable-ranges

acceptable-ranges ::= 1#range-unit | "none"
```

Клиент может генерировать запросы с диапазоном байтов, даже если он не получил ответа с таким полем в заголовке.

5.12.6. Age

Поле ответа Age содержит приблизительный период времени, прошедший с момента отправки ответа сервером-источником.

```
Age ::= "Age" ":" age-value

age-value ::= delta-seconds
```

Значения Age должны рассчитываться кэшем следующим образом:

```
apparent_age = max(0, response_time - date_value);
corrected_received_age = max(apparent_age, age_value);
response_delay = response_time - request_time;
corrected_initial_age = corrected_received_age + response_delay;
resident_time = now - response_time;
current_age   = corrected_initial_age + resident_time;
```

где

age_value - это значение Age из заголовка данного ответа,

полученного кэшем
date_value - значение Data заголовка сообщения от сервера-источника
request_time - местное время, когда кэш сделал запрос, в результате которого появилось данное сообщение
response_time - местное время, когда кэш получил ответ
now - текущее местное время

Если кэш получает значение большее, чем значение наибольшего положительного целого, с которым он может работать, либо при переполнении во время вычисления Age, он должен передать поле Age со значением 2147483648 (2 ^ 31). Поле Age должно присутствовать в каждом ответе.

5.12.7. Allow

Поле заголовка Allow содержит список методов, поддерживаемых для ресурса, указанного в Request-URI. Поле заголовка Allow должно присутствовать в ответе 501.

```
Allow       ::= "Allow" ":" 1#method
```

Например:

Allow: GET, HEAD, PUT

Прокси не должен изменять поле Allow, даже если он не поддерживает все указанные методы.

5.12.8. Authorization

Клиент, которому необходимо идентифицироваться, - обычно после ответа 401 (п.4.4.2, табл. 1), - может выполнить процедуру идентификации, включив заголовок ответа поля Authorization. Содержимое поля Authorization состоит из информации, идентифицирующей права клиентского агента на область запрашиваемых им ресурсов.

```
Authorization  = "Authorization" ":" credentials
```

5.12.9. Cache-Control

Общее поле Cache-Control используется для определения директив, которым должны подчиняться все кэши в цепочке запрос/ответ.
Директивы должны передаваться через прокси или шлюз вне зависимости от того, имеют ли они для данного прокси или шлюза какое-либо значение.

```
Cache-Control   ::= "Cache-Control" ":" 1#cache-directive

cache-directive ::= cache-request-directive
                  | cache-response-directive

cache-request-directive ::=
                  "no-cache" [ "=" <"> 1#field-name <"> ]
                | "no-store"
```

```
                | "max-age" "=" delta-seconds
                | "max-stale" [ "=" delta-seconds ]
                | "min-fresh" "=" delta-seconds
                | "only-if-cached"
                | cache-extension

cache-response-directive ::=
                  "public"
                | "private" [ "=" <"> 1#field-name <"> ]
                | "no-cache" [ "=" <"> 1#field-name <"> ]
                | "no-store"
                | "no-transform"
                | "must-revalidate"
                | "proxy-revalidate"
                | "max-age" "=" delta-seconds
                | cache-extension

cache-extension ::= token [ "=" ( token | quoted-string ) ]
```

5.12.10. Connection

Общее поле заголовка Connection позволяет отправителю определить факультативные возможности для отдельного соединения транспортного уровня, которые не должны передаваться серверами прокси по дальнейшим соединениям транспортного уровня.

```
Connection-header ::= "Connection" ":" 1#(connection-token)
connection-token  ::= token
```

Перед тем, как направить сообщение с полем Connection, для каждого маркера connection-token в данном поле сервер должен удалить поле (поля) заголовка с соответствующим именем.

Специальный маркер "close" является сигналом закрытия соединения после выполнения данного запроса.

Приложения HTTP версии 1.1, которые не поддерживают стойкие соединения, должны включать маркер "close" в каждое сообщение.

5.12.11. Content-Base

Поле заголовка сущности Content-Base может использоваться для идентификации базового URI, используемого совместно с относительными URL внутри сущности.

```
Content-Base     ::= "Content-Base" ":" absoluteURI
```

Если поле Content-Base отсутствует, базовый URI сущности определяется либо ее полем Content-Location (если Content-Location URI - это абсолютный URI), либо URI, использованным для инициирования запроса.

5 12.12. Content-Encoding

Поле заголовка сущности Content-Encoding используется как модификатор к типу информации. Присутствие данного поля показывает, что к телу сущности было применено дополнительное кодирование содержимого.

Content-Encoding ::= "Content-Encoding" ":" 1#content-coding

Если было применено несколько кодирований, они должны быть перечислены в порядке, в котором они применялись.

5.12.13. Content-Language

Поле заголовка сущности Content-Language описывает естественный язык информации в теле сущности.

Content-Language ::= "Content-Language" ":" 1#language-tag

5.12.14. Content-Length

Поле заголовка сущности Content-Length указывает на размер тела сообщения, в октетах. В случае метода HEAD поле заголовка содержит размер тела сообщения, которое могло бы быть послано клиенту в случае использования метода GET.

Content-Length ::= "Content-Length" ":" 1*DIGIT

5.12.15. Content-Location

Поле заголовка сущности Content-Location может быть использовано для указания местоположения ресурса для сущности.

Content-Location ::= "Content-Location" ":"
(absoluteURI | relativeURI)

5.12.16. Content-MD5

Поле заголовка сущности Content-MD5, в соответствии с RFC 1864 [25], содержит контрольный код MD5 для тела сущности, обеспечивая проверку целостности сообщения "из конца в конец".

Content-MD5 ::= "Content-MD5" ":" md5-digest

md5-digest ::= < 128 bit MD5 в соответствии с RFC 1864[25],
закодированный в 64Base>

5.12.17. Content-Range

Поле заголовка сущности Content-Range посылается с частичным телом сущности, чтобы указать, где частичное тело должно быть вставлено в полное тело сущности. Так же оно указывает общий размер полного тела сущности.

Content-Range = "Content-Range" ":" content-range-spec

content-range-spec ::= byte-content-range-spec

byte-content-range-spec ::= bytes-unit SP first-byte-pos "-"
last-byte-pos "/" entity-length

entity-length ::= 1*DIGIT

5.12.18. Content-Type

Поле заголовка сущности Content-Type указывает на тип информации тела сущности.

Content-Type ::= "Content-Type" ":" media-type

5.12.19. Date

Поле заголовка сущности Date представляет дату и время генерации сообщения.

Date ::= "Date" ":" HTTP-date

Данное поле должно быть включено во все ответы.

5.12.20. ETag

Поле заголовка сущности ETag определяет тег сущности для размещенной сущности.

ETag ::= "ETag" ":" entity-tag

Например:

ETag: "xyzzy"
ETag: W/"xyzzy"
ETag: ""

5.12.21. Expires

Поле заголовка сущности Expires указывает дату и время, после который данный ответ будет считаться устаревшим. Формат этого поля должен соответствовать RFC 1123 [18].

Expires ::= "Expires" ":" HTTP-date

Если формат поля не соответствует данным требованиям, клиенты и кэши должны обрабатывать такой ответ как устаревший.

5.12.22. From

Поле заголовка запроса From должно содержать адрес электронной почты Internet пользователя, контролирующего программу-клиент

From ::= "From" " " mailbox

5.12.23. Host

Поле заголовка запроса Host указывает узел Internet и номер порта ресурса, который запрашивается.

Host ::= "Host" ":" host [":" port]

Клиент HTTP версии 1.1. должен включать поле Host во все запросы. Если поле Host отсутствует, прокси должен добавить это поле.

Сервер HTTP версии 1.1. должен отвечать сообщением 400 при получении запроса с отсутствующим полем Host.

5.12.24. If-Modified-Since

Поле заголовка запроса If-Modified-Since используется с методом GET, чтобы сделать его условным: если запрошенный вариант не был изменен с указанного времени, сущность не будет возвращена. Вместо этого будет возвращен ответ 304.

If-Modified-Since ::= "If-Modified-Since" ":" HTTP-date

Если формат даты неправильный, сервер должен выдать ответ на обычный (не условный) GET.

5.12.25. If-Match

Поле заголовка запроса If-Match используется вместе с методом, позволяющем сделать его условным. Клиент, у которого есть одна или более ранее полученных сущностей, может проверить, является ли одна из них текущей версией, включив соответствующий список тегов сущностей в данное поле. Значение "*" обозначает любую текущую сущность.

If-Match ::= "If-Match" ":" ("*" | 1#entity-tag)

Если хотя бы один тег сущностей совпадает с тегом сущности, которая могла бы быть возвращена на подобный безусловный запрос GET, либо если стоит символ "*" и существует текущая версия данного ресурса, тогда сервер может выполнить запрошенный метод.

Сервер должен использовать сильную функцию сравнения.

При отсутствии совпадения тегов, либо при отсутствии текущей версии сущности, сервер не должен выполнять указанный метод, а должен возвратить ответ 412.

5.12.26. If-None-Match

Поле заголовка запроса If- None-Match используется с методом, чтобы сделать его условным. Использование данного поля подобно полю If-Match.

Если ни один из тегов сущностей не совпадает с тегом сущности, которая могла бы быть возвращена на подобный безусловный запрос GET, либо если стоит символ "*" и не существует текущая версия данного ресурса, тогда сервер может выполнить запрошенный метод.

If-None-Match = "If-None-Match" ":" ("*" | 1#entity-tag)

5.12.27. If-Range

Если у клиента есть в кэше частичная копия сущности, и он хочет получить актуальную целую сущность в свой кэш, он может использовать условный GET с полем If-Range.

If-Range ::= "If-Range" ":" (entity-tag | HTTP-date)

Данный запрос будет означать: если сущность не изменилась, отправьте недостающую часть. В противном случае, отправьте целую сущность. Таким образом, сервер должен выдать ответ 206 либо 200.

5.12.28. If-Unmodified-Since

Поле заголовка запроса If- None-Match используется с целью сделать его условным. Если запрошенный ресурс не был изменен с указанного времени, сервер должен выполнить запрошенную операцию так, как если бы поле If-Unmodified-Since отсутствовало.

If-Unmodified-Since ::= "If-Unmodified-Since" ":" HTTP-date

Если запрошенный вариант был изменен с указанного времени, сервер должен выдать ответ 412.

5.12.29. Last-Modified

Поле сущности Last-Modified указывает дату и время, в которое, по сведениям сервера-источника, данный вариант был изменен последний раз.

Last-Modified ::= "Last-Modified" ":" HTTP-date

5.12.30. Location

Поле ответа Location используется для перенаправления получателя на иной адрес, чем Request-URI для выполнения запроса или идентификации нового ресурса.
Для ответа 201 в данном поле указывается новый созданный ресурс. Для ответов 3xx Location указывает на URL для автоматического перенаправления.

Location ::= "Location" ":" absoluteURI

5.12.31. Max-Forwards

Данное поле запроса может использоваться с методом TRACE для ограничения числа прокси или шлюзов, которые могут направлять запрос.

Max-Forwards = "Max-Forwards" " " 1*DIGIT

Значение поля представляет собой десятичное число, показывающее остающееся число разрешенных пересылок сообщения.

Прокси, получивший запрос с полем Max-Forwards=0, не должен направлять данное сообщение, а должен выдать ответ 200.

5.12.32. Pragma

Поле общего заголовка Pragma используется для включения директив, зависящих от реализации, которые могут использоваться любым получателем в цепочке запрос/ответ. Все директивы Pragma определяют особые факультативные функции протокола.

```
Pragma           ::= "Pragma" ":" 1#pragma-directive
pragma-directive ::= "no-cache" | extension-pragma
extension-pragma ::= token [ "=" ( token | quoted-string ) ]
```

Директивы Pragma должны передаваться без изменения через прокси или шлюз, вне зависимости от их значения для данного прокси или шлюза.

5.12.33. Proxy-Authenticate

Поле заголовка ответа Proxy-Authenticate должно быть включено в ответ 407. Значение данного поля состоит из вызова, указывающего схему идентификации и параметры, применимые к прокси, имеющему данный Request-URI.

Proxy-Authenticate ::= "Proxy-Authenticate" ":" challenge

5.12.34. Proxy-Authorization

Поле заголовка запроса Proxy-Authorization позволяет клиенту идентифицировать самого себя и пользователей. Значение поля состоит из мандатов, содержащих идентифицирующую информацию о клиенте для прокси и (или) пространство запрашиваемого ресурса.

Proxy-Authorization ::= "Proxy-Authorization" ":" credentials

В отличие от поля Authorization поле Proxy-Authorization используется только ближайшим внешним прокси, который затребовал идентификацию используя поле Proxy-Authenticate. В прокси может быть реализован механизм групповой идентификации клиента совместно с другими серверами прокси.

5.12.35. Public

Поле заголовка Public ответа содержит перечень методов, поддерживаемых сервером. Данное поле может использоваться совместно с полем Allow, которое в свою очередь перечисляет методы, применимые для конкретного Request-URI.

Public .:= "Public" ":" 1#method

Данное поле применимо только к серверу, имеющему непосредственное соединение с клиентом.

Если ответ, содержащий данное поле, проходит через сервер прокси, сервер прокси должен изменить содержимое данного поля таким образом, чтобы оно соответствовало поддерживаемым этим прокси методам.

5.12.36. Range

5.12.36.1. Диапазоны байтов

Так как все сущности HTTP представлены в сообщениях HTTP в виде последовательности байтов, понятие диапазона байтов применимо для любой сущности. Но поддержка операции выделения диапазонов байтов не является обязательной.

Операция выделения диапазона байтов может определять один или несколько диапазонов байтов внутри одной сущности.

ranges-specifier ::= byte-ranges-specifier

byte-ranges-specifier ::= bytes-unit "=" byte-range-set

byte-range-set ::= 1#(byte-range-spec | suffix-byte-range-spec)

byte-range-spec ::= first-byte-pos "-" [last-byte-pos]

first-byte-pos ::= 1*DIGIT

last-byte-pos ::= 1*DIGIT

suffix-byte-range-spec ::= "-" suffix-length

suffix-length ::= 1*DIGIT

где
first-byte-pos – смещение первого байта диапазона,
last-byte-pos – смещение последнего байта в диапазоне. Диапазон определяется от поля first-byte-pos до last-byte-pos включительно. Смещение отсчитывается от нулевого байта сущности.

Получатель неправильно определенного диапазона должен его игнорировать.

С помощью поля suffix-byte-range-spec можно указать N последних байтов сообщения.

При превышении значением suffix-byte-range-spec длины сущности, диапазон определяет всю сущность.

5.12.36 2. Запросы на получение диапазона

Запрос на получение диапазона должен использовать условный или безусловный метод GET. Для определения диапазона используется поле заголовка запроса Range.

Range ::= "Range" ":" ranges-specifier

При выдаче успешного ответа на запрос с определенным диапазоном сервер должен использовать ответ 206

5.12.37. Referer

Поле заголовка запроса Referer позволяет клиенту указать адрес (URI) ресурса, от которого был получен Request-URI.

Данное поле не должно включаться в заголовок, если источник, от которого был получен Request-URI, не имеет собственного URI (например, если Request-URI был введен с консоли пользователя).

Referer ::= "Referer" ":" (absoluteURI | relativeURI)

Например:

Referer: gost.e-ecolog.ru

URI, указанное в поле Referer, не должно быть URI фрагмента. Для относительного URI в качестве базы должен использоваться Request-URI.

Рекомендуется, чтобы клиент имел возможность включать/отключать отправку поля Referer в запросах.

5.12.38. Retry-After

Поле заголовка ответа Retry-After может использоваться с ответом 503 для указания того, насколько долго сервис будет недоступен данному клиенту.

Retry-After ::= "Retry-After" ":" (HTTP-date | delta-seconds)

5.12.39. Server

Поле заголовка ответа Server содержит информацию о программном обеспечении, используемом на сервере-источнике для обработки запроса.

Server ::= "Server" ":" 1*(product | comment)

Прокси не должен изменять заголовок Server при направлении ответа.

Прокси может включить поле Via для идентификации самого себя в качестве прокси, направившего ответ.

5.12.40. Transfer-Encoding

Поле общего заголовка Transfer-Encoding указывает, что к телу сообщения был применен определенный тип преобразования для обеспечения безопасной пересылки между отправителем и получателем (оно определяет кодирование при пересылке). В

отличие от Content-Encoding значение поля применимо ко всему телу сообщения, а не к отдельной сущности

```
Transfer-Encoding      ::= "Transfer-Encoding" ":"
                           1=transfer-coding
```

5.12.41. Upgrade

Поле общего заголовка Upgrade позволяет клиенту указать поддерживаемый им дополнительный протокол связи, на который он предлагает серверу переключиться. Сервер должен использовать данное поле в ответе 101 для указания протокола, на который произошло переключение.

```
Upgrade       ::= "Upgrade" ":" 1#product
```

Ответ 101 с полем Upgrade должен быть первым ответом сервера после переключения на альтернативный протокол по запросу клиента с полем Upgrade.

Согласование протокола с использованием поля Upgrade применимо только для текущего соединения транспортного уровня.

Поле заголовка Upgrade должно использоваться внутри поля заголовка Connection.

5.12.42. User-Agent

Поле заголовка запроса User-Agent содержит информацию о программном обеспечении клиента, выдавшего запрос. Клиенты не обязательно должны включать данное поле в запрос.

```
User-Agent     ::= "User-Agent" ":" 1*( product | comment )
```

5.12.43. Vary

Поле заголовка ответа Vary используется сервером, чтобы сигнализировать, что сущность, содержащаяся в ответе, была выбрана из ряда доступных представлений с использованием механизма управляемого сервером согласования содержимого. Перечень полей, содержащихся в данном поле, указывает на размерность варьирования представления ресурса. Символ "*" указывает на то, что размерность варьирования не определена.

```
Vary  ::= "Vary" ":" ( "*" | 1#field-name )
```

Поле Vary должно включаться в каждый кэшируемый ответ, являющийся предметом управляемого сервером согласования содержимого.

Поле Vary может включаться в некэшируемый ответ, являющийся предметом управляемого сервером согласования содержимого.

Если кэш получает запрос, Request-URI которого определяет одну или более содержащих поле Vary позиций кэша, он не должен использовать эти позиции кэша, если только в запросе не присутствуют все поля, перечисленные в поле Vary, и содержимое этих полей в запросе не совпадает с содержимым аналогичных полей в позиции кэша.

Сравнение полей заголовка производится без учета конкретного вида разделителей LWS.

5.12.44. Via

Поле общего заголовка Via должно использоваться шлюзами и серверами прокси для указания промежуточных получателей и протоколов, с помощью которых было передано сообщение между клиентом и запрашиваемым сервером или между сервером-источником и клиентом.

```
Via ::=  "Via" ":" 1#( received-protocol received-by [ comment ] )

received-protocol  := [ protocol-name "/" ] protocol-version
protocol-name      ::= token
protocol-version   ::= token
received-by        ::= ( host [ ":" port ] ) | pseudonym
pseudonym          ::= token
```

Если протоколом является HTTP, тогда (и только тогда) received-protocol может не указываться. Вместо настоящего имени узла может указываться псевдоним.

Каждый получатель сообщения должен добавить свою информацию таким образом, чтобы в конечном счете список соответствовал порядку пересылавших сообщение приложений.

По умолчанию port=80.

Добавление информации в поле Via является необязательным.

Прокси, если он выполняет функции межсетевого экрана, может стереть информацию, добавленную в поле Via, либо заменять смежные цепочки идентификаторов получателей на единое имя.

Прокси не должен заменять смежные цепочки идентификаторов получателей на единое имя, если в данной цепочке присутствуют идентификаторы различных протоколов получателей.

5.12.45. Warning

Поле заголовка ответа Warning используется для переноса дополнительной информации о статусе ответа, которая не была отражена в коде статуса ответа.

```
Warning    ::= "Warning" ":" 1#warning-value

warning-value ::= warn-code SP warn-agent SP warn-text
warn-code  ::= 2DIGIT
warn-agent ::= ( host [ ":" port ] ) | pseudonym
                ; имя или псевдоним сервера, добавившего
                ; поле заголовка Warning
warn-text  ::= quoted-string
```

Если в тесте warn-text используется набор символов, отличный от ISO-8859-1 [16], текст должен быть закодирован в соответствии с RFC 2047 [17].

Кэш не должен удалять поля Warning. Но при проверке актуальности позиции кэш должен удалить все поля Warning, ранее присоединенные к данному сообщению. Коды warn-code приведкны в табл. 3.

Таблица 3

Коды предупреждений warn-code

Код	Описание
10	Ответ устарел. Должен быть включен в устаревший ответ. Кэш не должен удалять данный код, пока не будет положительного ответа на проверку актуальности
11	Ошибка проверки актуальности. Должна быть включена в ответ, выдаваемый кэшем, если ответ является устаревшим, и возникла ошибка при попытке проверить актуальность ответа (возможно, из-за недостижимости сервера).
12	Работа в разъединенном режиме. Может использоваться, если кэш намеренно отключен от остальной части сети на период времени.
13	Эвристическое истечение срока Должно использоваться, если кэш эвристически выбрал срок актуальности более 24 часов, и возраст ответа превышает 24 часа.
14	Применено преобразование Должно использоваться промежуточным кэшем или сервером прокси, если они применяют какое-либо преобразование, изменяющее кодировку содержимого (указанную в поле Content-Encoding) или тип информации (указанный в поле Content-Type) ответа. Не должно удаляться из ответа даже после проверки актуальности.
99	Различные предупреждения Текст предупреждения может включать произвольную информацию, предназначенную для чтения человеком-пользователем или для записи в журнальный файл. Система, получившая данное предупреждение, не должна автоматически предпринимать никаких действий.

5.12.46. WWW-Authenticate

Поле заголовка ответа WWW-Authenticate должно быть включено в сообщение ответа 401 (п. 4.4.2, табл.1). Значение поля состоит как минимум из одного вызова, указывающего на схему идентификации и параметров, применимых для данного Request-URI.

```
WWW-Authenticate  ::= "WWW-Authenticate" ":" 1#challenge
```

ПРИЛОЖЕНИЕ 6

ТЕХНИЧЕСКИЕ ТРЕБОВАНИЯ К ТЕХНИЧЕСКИМ СРЕДСТВАМ СЛУЖБЫ ДОСТУПА К ИНФОРМАЦИОННЫМ РЕСУРСАМ ПО ПРОТОКОЛУ NNTP

1. ОБЛАСТЬ ПРИМЕНЕНИЯ

Настоящее приложение описывает технические требования к ТС службы доступа к информационным ресурсам а по протоколу NNTP в соответствии с RFC 977 [26].

В приложении приведены операции взаимодействия клиента с различными типами ресурсов сети передачи данных, доступ к которым осуществляется посредством сервера NNTP. NNTP обеспечивает распространение сообщений электронных новостей по сети передачи данных общего пользования с использованием протокола NNTP и предназначен для подключения к ВСС России. Протокол NNTP используется для передачи статей электронных новостей между серверами NNTP и от сервера клиенту.

Не все функции, содержащиеся в данном приложении, обязательны для ТС службы доступа к информационным ресурсам по протоколу NNTP, но если они выполняются , то их реализация должна соответствовать настоящему приложению.

2. ФУНКЦИОНАЛЬНЫЕ ТРЕБОВАНИЯ К ВЗАИМОДЕЙСТВИЮ КЛИЕНТА NNTP И СЕРВЕРА NNTP

2.1. Соединения

2.1.1. Протокол нижнего уровня

При использовании протоколом NNTP в качестве протокола нижнего уровня TCP, должен использоваться порт 119.

При использовании TCP каждый семибитный символ, передаваемый по соединению TCP должен передаваться в отдельном октете. При этом символ должен быть выровнен вправо, а старший бит октета установлен в 0.

2.1.2. Установление соединения и закрытие соединения

Клиент должен инициировать установление соединения протокола нижнего уровня. После установления соединения сервер должен выдать ответ 200 либо 201.

Закрытие соединения осуществляется в следующей последовательности: клиент выдает команду QUIT, сервер выдает ответ 205, сервер разрывает соединение протокола нижнего уровня.

3. ПЕРЕЧЕНЬ И СТРУКТУРА СООБЩЕНИЙ ПРОТОКОЛА NNTP

Данные между клиентом и сервером передаются в форме сообщений. Необходимо различать сообщения, передаваемые между клиентом и сервером протокола NNTP и сообщения электронных новостей, к которым клиенту предоставляется доступ по протоколу NNTP.

Сообщения NNTP подразделяются на команды и ответы .

3.1. Команды

3.1.1. Структура команд

Команда должна состоять из кода команды и аргументов. Для ряда команд аргумент не требуется. Слово команды и последующий аргумент, а также аргументы между собой должны быть отделены одним или более символами <SP>.

Различие между строчными и прописными символами в кодах команд и аргументах команд является несущественным.

Каждая команда должна заканчиваться символами <CRLF>.

Длина команды должна быть менее или равна 512 символов, включая все символы <SP>, символы разделителей и <CRLF>. Все команды должны состоять из одной строки.

3.1.2. Список команд

Список команд приведен в табл. 1.

Таблица 1

Список команд

Команда	ARTICLE [< message-id > / <nnn>]
Аргументы	Message-id - идентификатор статьи Nnn – числовой идентификатор статьи
Описание	Сервер выдает ответ статуса с указателем текущей статьи и идентификатором статьи, а также текстовый ответ с заголовком и текстом указанной статьи. Message-id соответствует идентификатору, указанному в заголовке статьи. Nnn соответствует числовому идентификатору статьи в данной группе новостей. Если в качестве аргумента указан message-id, указатель текущей строки не должен меняться при выполнении данной команды. Если аргумент отсутствует, должны выдаваться данные текущей статьи. Если присутствует аргумент nnn, указатель текущей строки после выполнения команды должен быть установлен в nnn.
Ответы статуса	220, 221, 222, 223, 412, 420, 423, 430

Команда	BODY [<message-id > / <nnn>]
Аргументы	Message-id - идентификатор статьи nnn – числовой идентификатор статьи
Описание	Сервер выдает ответ статуса с указателем текущей статьи и идентификатором статьи, а также текстовый ответ с текстом указанной статьи. Аргументы обрабатываются аналогично команде ARTICLE
Ответы статуса	220, 221, 222, 223, 412, 420, 423, 430

Команда	HEAD [< message-id > / <nnn>]
Аргументы	message-id - идентификатор статьи nnn - числовой идентификатор статьи
Описание	Сервер выдает ответ статуса с указателем текущей статьи и идентификатором статьи, а также текстовый ответ с заголовком указанной статьи. Аргументы обрабатываются аналогично команде ARTICLE
Ответы статуса	220, 221, 222, 223, 412, 420, 423, 430

Команда	STAT [<message-id >/ <nnn>]
Аргументы	message-id - идентификатор статьи nnn - числовой идентификатор статьи
Описание	Сервер выдает ответ статуса с указателем текущей статьи и идентификатором статьи. Текстовый ответ не возвращается. Данная команда используется для установки указателя текущей статьи. Аргументы обрабатываются аналогично команде ARTICLE
Ответы статуса	220, 221, 222, 223, 412, 420, 423, 430

Команда	GROUP <ggg>
Аргументы	ggg - имя группы новостей
Описание	Выбор группы новостей. Сервер выдает ответ статуса с номерами первой и последней статьи в группе, а также с примерным числом статей в группе. Указатель текущей статьи должен быть установлен на первую статью в группе. В случае указания неправильного (несуществующего) имени группы новостей остается выбранной предыдущая группа новостей, и указатель текущей статьи не меняется. Разница между прописными и строчными буквами в имени группы новостей не должна быть существенна.
Ответы статуса	211, 411

Команда	HELP
Аргументы	-
Описание	Помощь. Сервер должен выдавать ответ статуса и текстовый ответ, содержащий перечень команд, реализованных на данном сервере с кратким указанием их назначения.
Ответы статуса	100

Команда	IHAVE <message-id>
Аргументы	Message-id – идентификатор статьи
Описание	Команда информирует сервер, что у клиента есть статья с идентификатором message-id. Если на сервере есть копия данной статьи, он должен выдать ответ 335. Если на сервере нет копии данной статьи, он должен выдать ответ 435. Если сервер выдал ответ 335, клиент должен выслать статью в формате, аналогичном формату текстового ответа сервера.
Ответы статуса	235, 335, 435, 436, 437

Команда	LAST
Аргументы	-
Описание	Команда устанавливает указатель текущей статьи на предшествующую статью в текущей группе новостей.
Ответы статуса	223, 412, 420, 422

Команда	LIST
Аргументы	-
Описание	Выдается список доступных групп новостей, относящаяся к ним информация в ответе статуса 215 со следующим за ним текстовом ответом. При отсутствии доступных групп новостей должен выдаваться пустой текстовый ответ.
Ответы статуса	215

Команда	NEWGROUPS <date> <time> [GMT] [< distributions >]
Аргументы	date - дата создания группы новостей в формате YY MM DD. YY - год, MM - месяц, DD - день. . Для значений YY от 00 до 30 первые две цифры года равны 20 . Для значений YY от 70 до 99 берется 20 столетие - первые две цифры года равны 19. time - время создания группы новостей в формате HHMMSS. HH – часы, MM - минуты, SS - секунды. GMT - указывает на то, что время задано относительно 0-го меридиана distributions - список групп распространения distributions ::= 1#distribution distribution ::= <группа распространения новостей>
Описание	Выдается список групп новостей, созданных позднее времени, определяемого аргументами date и time, и группы распространения которых совпадают с указанными в аргументе distributions. Время, указываемое date и time, предполагается вычисленным для временной зоны сервера, если только не указан аргумент GMT.
Ответы статуса	231

Команда	NEWNEWS <newsgroups> <date> <time> [GMT] [< distribution>]
Аргументы	newsgroups - список шаблонов имени группы новостей. newsgroups ::= 1#(["!"] newsgroup) ; Символ "!" обозначает отрицание. newsgroup ::= <шаблон имени группы новостей. В шаблоне имени новостей может использоваться символ "*" (звездочка) в качестве обозначения произвольного количества пропущенных символов.> date - дата отправки или приема time - время отправки или приема Формат даты и времени аналогичен команде NEWGROUPS GMT - указывает на то, что время задано относительно 0-го меридиана distribution - список групп распространения. Используется аналогично команде NEWGROUPS

Описание	Сервер выдает список идентификаторов статей, которые были получены или отправлены в указанную группу новостей позднее указанной даты и времени. Список должен выдаваться в текстовом ответе. Под информацию об одной статье отводится одна строка. При отсутствии статей, удовлетворяющих условию, должен выдаваться пустой текстовый ответ.
Ответы статуса	230

Команда	NEXT
Аргументы	-
Описание	Команда устанавливает указатель текущей статьи на следующую статью в текущей группе новостей. .
Ответы статуса	223. 412, 420, 421

Команда	POST
Аргументы	-
Описание	Отправка статьи от клиента серверу. После команды POST сервер выдает ответ статуса 340, если он разрешает отправку статьи, и 440, если отправка статьи запрещена. При получении ответа сервера 340 клиент выдает статью в формате, аналогичном формату текстового ответа сервера.
Ответы статуса	240, 340, 440, 441

Команда	QUIT
Аргументы	-
Описание	На данную команду сервер должен выдать ответ 205 и закрыть соединение. Если соединение с клиентом разрывается по какой-либо причине, сервер не должен делать попытки восстановления соединения.
Ответы статуса	205

Команда	SLAVE
Аргументы	-
Описание	Данная команда показывает серверу, что клиентом является промежуточный сервер. Эта команда может быть использована сервером для установления повышенного приоритета данному соединению.
Ответы статуса	202

3.2. Ответы

Ответы делятся на текстовые ответы и ответы статуса.

3.2.1. Текстовый ответ должен состоять из серии текстовых строк. Каждая строка должна заканчиваться символами <CRLF>. Последовательность "<CRLF>.<CRLF>" (0x0D 0x0A 0x2E 0x0D 0x0A) указывает на конец текстового ответа.

При посылке сервером текстового ответа каждую последовательность "<CRLF>." (0x0D 0x0A 0x2E) сервер должен заменять на "<CRLF>.."(0x0D 0x0A 0x2E 0x2E). Клиент должен выполнять обратное преобразование. Указатель конца текстового ответа данному преобразованию не подвергается.

3.2.2. Ответ статуса должен состоять из 3-х значного кода ответа (передаваемого как три символа), за которым следует текст.

Значения номера ответа первой и второй цифры приведены в табл. 2 и табл. 3.

Таблица 2

Таблица 2. Содержание первой цифры ответа.

первая цифра	1	Информационное сообщение
	2	Положительный окончательный ответ (успешное выполнение команды)
	3	Положительный промежуточный ответ (часть операции успешно выполнена. Можно посылать следующую команду операции)
	4	Временный отрицательный окончательный ответ (команда реализована, но не может быть выполнена по какой-либо причине)
	5	Постоянный отрицательный окончательный ответ (команда не реализована, некорректно задана, произошла серьезная ошибка)

Таблица 3

Таблица 3. Содержание второй цифры ответа.

вторая цифра	0	Ошибки соединения, конфигурирования, другие ошибки
	1	Выбор группы новостей
	2	Выбор статьи новостей
	3	Функции распространения
	4	Отправка
	8	Нестандартные расширения
	9	Отладка

третья цифра позволяет сделать более точное разделение значений ответов по функциональным категориям, определенным второй цифрой.

Перед текстом ответа статуса могут следовать один или несколько параметров. Все числовые параметры представлены в виде десятичного числа. Параметры разделяются между собой, а также от последующего текста символами <SP>.

3.2.3. Список ответов

Должны быть реализованы ответы, перечисленные в табл. 4.

Таблица 4

Список ответов

Код	Текст	Последующий текстовый ответ с учетом требований п. 3.2.1.
100	Далее следует текст помощи	Текстовые строки помощи
199	Вывод отладки	-
200	Сервер готов Разрешена отправка статей на сервер.	-
201	Сервер готов. Запрещена отправка статей на сервер.	-
202	Учтен статус промежуточного сервера	-
205	Закрытие соединения	-
211	<n> <f> <l> <s> группа выбрана n ::= <приблизительное число статей в группе> f ::= <номер первой статьи в группе> l ::= <номер последней статьи в группе> s ::= <имя группы>	-
215	далее следует список групп новостей	Набор строк: <group> <last> <first> <p> group ::= <имя группы новостей> last ::= <номер последней статьи в группе> first ::= < номер первой статьи в группе> p ::= "y"/"n" ; указывает на то, разрешена ли отправка статей в данную группу
220	<n> <a> статья получена Далее следуют заголовок и тело статьи n ::= <номер статьи> a ::= <идентификатор статьи> Если идентификатор статьи отсутствует в заголовке статьи, вместо него должен быть поставлен "0".	Заголовок и тело статьи в виде текстовых строк
221	<n> <a> статья получена Далее следует заголовок статьи Значения n и a аналогично ответу 220.	Заголовок статьи в виде текстовых строк с учетом.
222	<n> <a> статья получена Далее следует тело статьи Значения n и a аналогично ответу 220.	Тело статьи в виде текстовых строк
223	<n> <a> статья получена Значения n и a аналогично ответу 220.	-
230	Далее следует перечень идентификаторов новых сообщений	набор строк message-id <CRLF> message-id ::= <идентификатор сообщения>
231	Далее следует список новых групп новостей	Список групп новостей в формате, аналогичном ответу 215

235	Статья передана без ошибок	-
240	Статья отправлена без ошибок	-
335	Жду передачи статьи.	-
340	Жду отправки статьи.	-
400	Служба не работает	-
411	Нет такой группы новостей	-
412	Не была выбрана группа новостей	-
420	Не был установлен указатель текущей статьи	-
421	Последняя статья в данной группе	-
422	Первая статья в данной группе	-
423	Нет статьи с таким номером в данной группе	-
430	Не найдена такая статья	-
435	Статья не нужна. Не отправлять.	-
436	Ошибка передачи. Попробуйте позже.	-
437	Статья отклонена. Больше ее не отправляйте.	-
440	Отправка не разрешена.	-
441	Ошибка отправки	-
500	Команда не распознана	-
501	Синтаксическая ошибка команды	-
502	Ограничение доступа. Нет разрешения.	-
503	Ошибка программы. Команда не выполнена.	-

4. ТРЕБОВАНИЯ К СТРУКТУРЕ СТАТЕЙ

4.1. Формат электронной статьи должен соответствовать формату, описанному RFC 822 [2].

4.2. Обязательно должны присутствовать поля заголовка:

from;
date;
newsgroups;
subject;
message-id;
path.

4.3. Кроме указанных в п. 4.2. могут присутствовать поля заголовка:

followup-to;
expires;
reply-to;
sender;
references;

control;
distribution;
keywords;
summary;
approved;
lines;
xref,
organization.

Обязательные поля заголовка статьи должны интерпретироваться следующим образом (определения элементов, используемых в выражениях описания, приведены в RFC 822 [2]):

1. f_from ::= (domain <SP> ["(" full_name ")]) /
(full_name <SP> "<" domain ">")
domain ::= <электронный адрес отправителя>
full_name ::= <Полное имя отправителя. Может состоять из любых печатных символов ASCII , кроме "(" , ")" , "<" , ">" , "," , ":" , "@" , "!" , "/" , "=" , ";">

2. Date ::= Day "," <SP> DD <SP> Mon <SP> YY <SP> HH ":" MM ":" SS <SP>
TIMEZONE

Элементы данного поля должны соответствовать RFC 822 [2]

3. Newsgroups ::= 1#Newsgroup
Newsgroup ::= <Имя существующей группы новостей. Не должно содержать слова "all" в качестве элемента иерархического имени>

4. Subject ::= ["Re:"] S
S ::= <Краткое описание темы сообщения.>

"Re:" должно быть указано в случае, если данная статья является ответом на другую статью. При этом требуется обязательное наличие поля References.

5. Message-ID ::= "<" addr-spec ">" ; идентификатор статьи
addr-spec ::= local-part "@" domain

domain - полное доменное имя узла, с которого сообщение поступило в сеть.
Выражение local-part должно быть уникальным для всех статей в данном домене.

6. Path ::= <путь, который статья прошла до достижения данной системы>

Когда система направляет сообщение, она должна добавить свое имя в список Path. Имя может быть отделено от других любым символом пунктуации, кроме точки.

7. References ::= * (<SP> Message-ID) ;

ПРИЛОЖЕНИЕ 7

ТЕХНИЧЕСКИЕ ТРЕБОВАНИЯ К ТЕХНИЧЕСКИМ СРЕДСТВАМ СЛУЖБЫ ДОСТУПА К ИНФОРМАЦИОННЫМ РЕСУРСАМ ПО ПРОТОКОЛУ FTP

1. ОБЛАСТЬ ПРИМЕНЕНИЯ

Настоящее приложение описывает технические требования к ТС службы доступа к информационным ресурсам а по протоколу NNTP в соответствии с RFC 959 [27].

В приложении приведены операции для удаленного управления файловой структурой сервера и пересылку файлов между узлами вычислительной сети общего пользования. Протокол FTP описывает операции удаления и переименования файлов удаленной файловой системы сервера с рабочего места клиента; пересылку файлов с сервера на сервер, от клиента серверу, от сервера клиенту; получение информации о файловой системе сервера.

Не все функции, содержащиеся в данном приложении, обязательны для ТС службы доступа к информационным ресурсам по протоколу FTP, но если они выполняются , то их реализация должна соответствовать настоящему приложению.

2. ФУНКЦИОНАЛЬНЫЕ ТРЕБОВАНИЯ К ВЗАИМОДЕЙСТВИЮ СЕРВЕРА FTP И КЛИЕНТА FTP

2.1. Модель FTP

Сервер FTP должен взаимодействовать с клиентом FTP в соответствии с основной (обязательной) и дополнительной (необязательной) моделью FTP.

2.1.1. Схема основной модели FTP

Схема основной модели FTP показана на рис. 1.

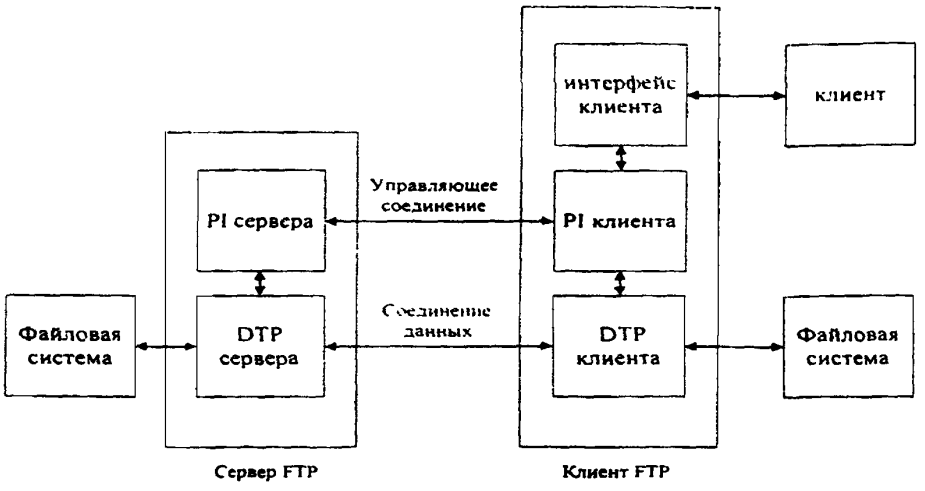


Рис. 1. Основная модель использования FTP

2.1.2. Схема дополнительной модели FTP

Схема дополнительной модели FTP показана на рис. 2.

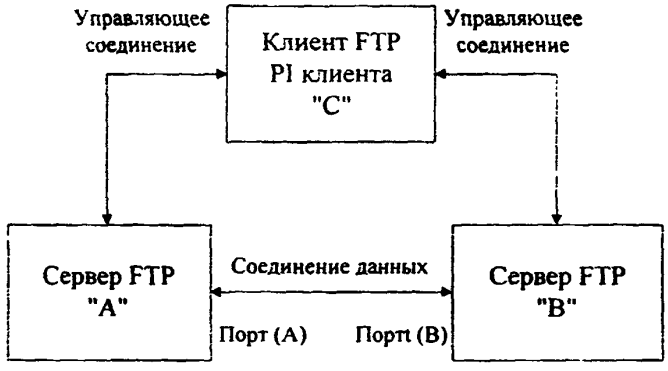


Рис. 2. Дополнительная модель использования FTP

2.2. Управляющее соединение

В ТС FTP должно быть реализовано управляющее соединение.

Управляющее соединение FTP устанавливается клиентом по протоколу TCP с использованием порта L на стороне сервера и порта U на стороне клиента. По умолчанию L=21.

2.2.1. Процедура установления соединения

Интерпретатор протокола сервера "слушает" порт TCP L. Клиент инициирует полнодуплексное управляющее соединение.

При передаче данных между серверами (п. 3.1.2.) PI клиента устанавливает управляющее соединение с обоими серверами.

На рис. 3 представлена процедура установления управляющего соединения дополнительной модели работы FTP.

PI клиента - сервер A		PI клиента - сервер B	
C->A	Connect	C->B	Connect
C->A	PASV		
A->C	227 Entering passive mode A1, A2, A3, A4, a1, a2		
		C->B	PORT A1, A2, A3, A4, a1, a2
		B->C	200 Okay
C->A	STOR	C->B	RETR
B->A Connect to host-A, host-B			

Рис. 3. Процедура установления управляющего соединения дополнительной модели FTP

2.2.2. Управление соединением

Установка, разрыв и управление соединением должно выполняться по протоколу Telnet.

Управляющее соединение должно быть закрыто сервером по запросу клиента.

2.3. Соединение данных

В ТС FTP должно быть реализовано соединение данных.

Соединение данных должно быть установлено на соответствующем порте перед передачей данных и выбраны соответствующие параметры передачи. DTP клиента и DTP сервера имеют порт по умолчанию - U-1 и L-1 соответственно. Длина передаваемых байтов - 8 бит.

2.3.1. Процедура установления соединения.

Установление соединения инициирует сервер.

Пассивный DTP (DTP клиента или DTP второго сервера) слушает порт данных перед тем, как послать команду запроса передачи. Команда запроса передачи FTP определяет направление передачи данных. Сервер, получив запрос, инициирует соединение данных на порту. После установления соединения начинается передача данных между DTP и PI сервера. Сервер посылает подтверждающий ответ процессу PI клиента.

Инициировать изменение портов, отличных от установленных по умолчанию, может только PI клиента. (команда PORT).

2.3.2. Процедура разрыва соединения.

Закрытие соединения, как правило, инициирует сервер.

Исключением является случай, когда DTP клиента посылает данные в режиме, определяющем закрытие соединения для индикации EOF. Сервер должен закрыть соединение при следующих условиях:

1. сервер выполнил передачу данных в режиме, требующем закрытия для индикации EOF.
2. сервер получил команду ABORT от клиента
3. спецификация порта изменена командой от клиента
4. управляющее соединение закрыто
5. произошла невосстановимая ошибка

В остальных случаях сервер инициирует закрытие соединения данных посылкой процессу клиента ответов 250 или 226.

2.3.3. Управление соединением данных

2.3.3.1.На стороне сервера номер порта соединения данных по умолчанию должен быть на 1 меньше номера порта управляющего соединения.

2.3.3.2. Процедура согласования портов данных, отличных от установленных по умолчанию (нестандартных)

PI клиента определяет нестандартный порт на стороне клиента командой PORT либо запрашивает о нестандартном порте на стороне сервера командой PASV. Эти команды могут использоваться вместе или по отдельности.

2.3.3.3. Переустановка соединения данных

При использовании режима передачи данных stream mode конец файла определяется закрытием соединения. Проблема передачи нескольких файлов может иметь два решения:

1. установка нестандартного порта.
2. использование другого режима передачи данных

2.3.4. Режимы передачи данных

Передающий узел должен преобразовывать знаки конца строки и конца записи, используемые для внутреннего хранения файла, в представление, соответствующее режиму передачи данных и файловой структуре. Принимающий узел должен выполнять обратную операцию.

Обязательно должны быть реализованы stream mode и block mode.

2.3.4.1. Stream mode (режим потока).

Данные передаются как поток байтов (в том числе байтов данных). Нет ограничений на используемый тип представления, позволяется использовать структуры записей.

Если файл имеет структуру записей (структурирован по записям), символы EOR и EOF индицируются двухбайтовым кодом. Причем первым байтом должен быть escape-символ, а второй байт должен быть 1 (единица в менее значащем бите) для EOR и 2

(единица во втором бите) для EOF. Последовательность из escape-символа и символа 3 (два младших бита равны 1) означает одновременное присутствие EOR и EOF.

Если файл неструктурирован, символ EOF индицируется передающим узлом путем закрытия соединения данных. Все передаваемые байты являются байтами данных.

Данный тип передачи устанавливается по умолчанию.

2.3.4.2. Block mode - поблочный режим

Файл передается как серия блоков данных с заголовками.
Длина заголовка - 3 байта - 16 младших бит занимает поле counter, 8 старших бит - descriptor.

Descriptor 8 бит	Count 16 бит

Заголовок содержит поля:

1. Поле счета (counter) - показывает общую длину блока в байтах (без учета заголовка), тем самым определяя начало следующего блока (блоки передаются один за другим без пауз и разрывов)

2. Поле описателя (дескриптора) (descriptor):

Код	Значение
128	EOF - последний блок файла
64	EOR - последний блок записи
16	restart marker - в блоке данных содержится символ рестарта
32	suspect data - передаваемые данные в блоке могут содержать ошибку

Допускается наличие нескольких дескрипторов в одном блоке (коды логически складываются - операция "И").

Данными маркера рестарта может быть набор печатных символов используемого алфавита (NVT-ASCII например), в который не должен входить пробел (Space).

2.3.4.3. Compressed mode - режим сжатия

Посылается три вида информации - регулярные данные (в виде строки байтов), сжатые данные (состоящие из репликаторов и заполнителей) и управляющая информация в виде двухбайтовой escape-последовательности. Если посылаются от 1 до 127 байт регулярных данных, этим данным должен предшествовать байт, в самом левом бите которого установлен 0, а в остальных содержится число байт регулярных данных.

2.3.4.3.1. Блок несжатых данных

1	7
0	n

8
d(1)

8
d(n)

n байт данных

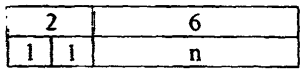
2.3 4.3.2. Блок репликатора

Для сжатия строки из n репликаций (копий) байта данных d посылаются два байта



2.3.4.3.3. Блок заполнителя

Строка из n байтов заполнителя может быть сжата в один байт. Байт заполнителя зависит от типа представления данных (для ASCII и EBCDIC - это <SP> (Space, 32 - ASCII, 64-EBCDIC), для типов Image и Local - 0).



Escape - последовательность - это сдвоенные байты, первый из которых - это escape-символ - 0, а второй содержит код дескриптора, определенный в Block Mode. Код дескриптора имеет то же значение, что и в Block Mode и применяется к строке байтов.

3. ТРЕБОВАНИЯ К ТИПАМ ДАННЫХ

При передаче информации по соединению данных могут использоваться следующие типы данных как: ASCII, EBCDIC, IMAGE и LOCAL, а также могут поддерживаться структуры данных: file, record, page. Обязательными для реализации являются типы данных ASCII и IMAGE, а также структуры данных file и record. Должен поддерживаться режим управления форматом Nonprint.

3.1. Типы данных

Тип данных определяет размер логических байтов и кодировку передаваемых байтов. Длина передаваемых байтов всегда составляет 8 бит. Размер логических байтов в типах ASCII, EBCDIC и IMAGE составляет 8 бит, а в типе LOCAL определяется параметром команды TYPE.

Могут поддерживаться следующие типы данных:
1. тип ASCII (NVT-ASCII). Этот тип должен использоваться по умолчанию.
2. тип EBCDIC.
3. тип IMAGE.
4. тип LOCAL.

3.2. Управление форматом

При использовании типов данных ASCII и EBCDIC для управления вертикальным форматированием при выдаче на печать (на экран) могут использоваться управляющие символы.

При использовании типов данных ASCII и EBCDIC могут быть реализованы следующие режимы форматирования:

3.2.1. Nonprint
3.2.2. Telnet CONTROLS
3.2.3. CARRIAGE CONTROLS ASA

По умолчанию устанавливается тип Nonprint.

3.3. Структуры данных.

Определены три типа структуры файлов.

3.3.1. Структура **file**. Файл считается непрерывной последовательностью байтов данных. Устанавливается по умолчанию.

3.3.2. Структура **record**. Файл состоит из последовательных записей. Данная структура применима для типов данных ASCII и EBCDIC.

3.3.3. Структура **page**. Файл состоит из независимых индексированных страниц. Каждая страница должна иметь заголовок, состоящий из следующих полей:

- длина заголовка (в логических байтах) минимум 4.
- индекс страницы (идентификатор страницы в файле)
- длина данных (количество логических байт данных)
- тип страницы

0 - последняя, при этом длина заголовка должна быть 4, длина данных - 0
1 - простая (длина заголовка должна быть 4)
2 - страница описателя (служит для передачи информации о файле в целом)
3 - страница контроля доступа (включает дополнительное поле заголовка для информации управления доступом. Длина заголовка - 5.

Все поля заголовка страницы имеют длину один логический байт.

4. ТРЕБОВАНИЯ К ВОССТАНОВЛЕНИЮ ОТ ОШИБОК И РЕСТАРТУ

В ТС FTP может быть реализована функция рестарта.

4.1. Процедура рестарта

Процедура рестарта предоставляется для защиты клиентов от грубых ошибок системы (включая ошибки узла, процесса FTP и сети передачи данных). Процедура определена только для режимов передачи Block mode и Compressed mode. Отправитель данных периодически вставляет в поток передаваемых данных маркер рестарта с данными маркера рестарта. Принимающий узел выделяет из потока данных маркеры рестарта и отправляет их клиенту. Для выдачи клиенту сообщения о рестарте на удаленном узле должен использоваться ответ 110. В случае системной ошибки клиент может начать заново передачу данных, идентифицируя контрольную точку с помощью процедуры рестарта. Для этого посылается команда рестарта с кодом маркера в качестве аргумента.

4.2. Формат маркера рестарта

Данные маркера рестарта кодируются печатными символами алфавита, установленного по умолчанию (ASCII или EBCDIC). Маркер должен содержать

информацию о счетчике битов, записей и любую другую информацию, в соответствии с контрольной точкой данных. Конкретное содержание маркера имеет значение только для передающего узла и поэтому не оговаривается.

5. ТРЕБОВАНИЯ К СТРУКТУРЕ И СОСТАВУ СООБЩЕНИЙ СЕРВЕРА FTP И КЛИЕНТА FTP

5.1. Команды FTP

От клиента FTP к серверу FTP по управляющему соединению информация передается в форме команды клиента FTP.

5.1.1. Формат команд FTP

Команды FTP являются строками символов алфавита NVT-ASCII. Возможно использование другого языка. Аргументы отделяются символом <SP>. Конец определяется символом <CLRF>. Не должно делаться различия между прописными и строчными буквами как в команде, так и в аргументе.

5.1.2. Перечень команд FTP приведен в табл. 1.

Таблица 1

Перечень команд FTP

№	Команда	Сокращение	Описание	Поле аргумента
Команды управления доступом				
1.	USER NAME	USER	Имя клиента	идентификатор клиента.
2.	PASSWORD	PASS	Пароль	пароль клиента
3.	ACCOUNT	ACCT	Полномочия	идентификатор клиентских полномочий
4.	Change working directory	CWD	Сменить рабочую директорию	новая рабочая директория
5.	Change to parent directory	CDUP	Вернуться в родительскую директорию	-
6.	Structure mount	SMNT	Смонтировать структуру	имя пути, определяющее директорию
7.	Reinitialize	REIN	Реинициализация. (закрытие соединения данных и сброс всех установок на по умолчанию)	
8.	Logout	QUIT	Выход	
Команды параметров передачи. Аргументы команд параметров передачи являются необязательными. Команда без параметров сбрасывает установки на по умолчанию.				

9.	Data port	PORT	Порт данных	h1, h2, h3, h4, p1, p2, где h1 .. h4 – адреса узла интернет, p1, p2 – адреса порта TCP.
10.	Passive	PASV	Пассивный режим	
11.	Representation type	TYPE	Тип представления данных	A – ASCII E – EBCDIC I – Image L <размер байта> - Local Byte Для A и E определен второй параметр: N – непечатный T – Telnet C – ASA **По умолч. - A N**
12.	File structure	STRU	Структура файла	F – неструктурирован R – структ. Записей P – структура страниц **По умолч. - F**
13.	Transfer mode	MODE	Режим передачи	S – Stream (поток) B – Block C – Compressed **По умолч.- S**
Команды услуг FTP				
14.	Retrieve	RETR	Пересылка от DTP сервера к DTP клиента (второго сервера)	имя файла
15.	Store	STOR	Прием процессом DTP сервера данных и сохранение в виде файла с замещением данных в случае совпадения имени файла.	имя файла
16.	Store Unique	STOU	Прием процессом DTP сервера данных и сохранение с генерацией уникального имени файла	имя файла
17.	Append	APPE	Прием процессом DTP сервера данных и добавление к существующему файлу	имя файла
18.	Allocate	ALLO	резервирование памяти	Число байт (логических) [R <максимальный размер записи или страницы>]

19.	Restart	REST	Рестарт. За этой командой должна немедленно следовать команда передачи файла.	
20.	Rename from	RNFR	Старое имя переименовываемого файла. За этой командой должна немедленно следовать команда RNTO.	старое имя файла
21.	Rename to	RNTO	Переименование файла. Этой команде должна предшествовать команда RNFR.	новое имя файла
22.	Abort	ABOR	Отмена предыдущей команды услуг FTP и соответствующего процесса передачи данных.	
23.	Delete	DELE	Удаление файла на сервере	имя файла
24.	Remove directory	RMD	удаление директории (субдиректории)	имя директории
25.	Make directory	MKD	Создание директории (субдиректории)	имя директории
26.	Print working directory	PWD	Вызов ответа с информацией о рабочей директории	
27.	List	LIST	Вызов ответа со списком файлов в произвольном формате типа ASCII (EBCDIC) по соединению данных. Только при типе представления ASCII и EBCDIC.	[путь файла (маска)]
28.	Name list	NLST	Вызов ответа со списком директорий в формате путей, разделенных <CRLF> или <NL> по соединению данных. Только при типе представления ASCII и EBCDIC.	[путь файла (маска)]
29.	Site parameters	SITE	Дополнительные услуги сервера	

30.	System	SYST	Вызов ответа с типом операционной системы сервера, обозначенным в соответствии с RFC 943 [28].	
31.	Status	STAT	Вызов ответа статуса по управляющему соединению.	[путь файла (маска)]
32.	Help	HELP	Вызов ответа с информацией помощи по управляющему соединению	[имя команды]
33.	Noop	NOOP	Вызов ответа сервера OK.	

5.1.3. Синтаксис команд приведен в п.7.

5.1.4. Диаграммы состояний, поясняющие выполнение команд определены в п.8.

5.1.5. Обязательно должны быть реализованы команды: USER, REIN, PORT, TYPE, STRU, MODE, RETR, STOR, NOOP.

5.2. Ответы FTP

От сервера клиенту по управляющему соединению информация передается в форме ответа сервера. Ответ сервера должен следовать после каждой команды клиента.

5.2.1. Формат ответов FTP

После одной команды может быть более одного ответа. Ответ сервера может состоять из одной или нескольких строк.

Однострочный ответ состоит из:

трехзначного номера ответа, передаваемого как три символа,
символа <SP>,
текста,
символа <CRLF> .

Многострочный ответ состоит из:

трехзначного номера ответа, передаваемого как три символа,
символа "-"
текста первой строки
символа <CRLF>
текста второй строки
.....
трехзначного номера ответа, передаваемого как три символа,
символа <SP>,
текста последней строки,
символа <CRLF> .

Значения номера ответа:

первая цифра

1	Положительный предварительный ответ
2	Положительный окончательный ответ
3	Положительный промежуточный ответ
4	Временный отрицательный окончательный ответ
5	Постоянный отрицательный окончательный ответ

вторая цифра

0	Синтаксис
1	Информация
2	Соединения
3	Аутентификация и полномочия
4	Пока не определено
5	Файловая система

третья цифра позволяет сделать более точное разделение значений ответов по функциональным категориям, определенным второй цифрой

5.2.2. Список кодов ответов приведен в табл. 2. Для всех ответов, кроме 110, текст ответа не обязательно должен соответствовать приведенному в табл. 2.

Таблица 2

Список кодов ответов

Код	Текст
110	Ответ на маркер рестарта. В данном ответе текст ответа должен точно соответствовать приведенному ниже определению: MARK <yyyy> = <mmmm> Где yyyy - маркер потока данных от процесса клиента, а mmmm - эквивалентный маркер сервера. Символы пробела между маркерами и символом "=" являются существенными.
120	Служба будет готова через nnn минут.
125	Соединение данных открыто. Начало передачи.
150	Статус файла - ОК. Открываю соединение данных.
200	Команда выполнена успешно.
202	Команда не реализована. Является излишней на данном узле.
211	Статус системы или ответ помощи
212	Статус директории
213	Статус файла
214	Сообщение помощи.
215	NAME <system> <type> Имя системы имен. NAME - официальное имя системы в соответствии со списком Assigned Numbers.
227	Вхождение в пассивный режим. (h1, h2, h3, h4, p1, p2).

230	Пользователь идентифицирован, продолжение
250	Операция с файлом выполнено успешно.
257	"PATHNAME" создано.
220	Служба готова для нового пользователя.
221	Служба закрывает управляющее соединение.
225	Соединение данных открыто. Передача не осуществляется.
226	Закрытие соединения данных. Операция с файлом выполнена успешно.
331	Имя пользователя принято, необходим пароль.
332	Need account for login.
350	Запрошенная операция над файлом ожидает дальнейшей информации.
421	Сервис недоступен. Закрываю управляющее соединение.
425	Не могу открыть соединение данных.
426	Соединение закрыто. Передача отменена.
450	Файл недоступен.
451	Запрошенная операция отменена. локальная ошибка.
452	Запрошенная операция не принята. Недостаточно памяти.
500	Синтаксическая ошибка. Команда нераспознана.
501	Синтаксическая ошибка в аргументах.
502	Команда не реализована.
503	Неправильная последовательность команд.
504	В команде не реализован данных параметр.
530	Нет подсоединения.
532	Необходимы права на сохранение файла.
550	Запрошенная операция отменена. Файл недоступен.
551	Запрошенная операция отменена. Неизвестный тип страницы.
552	Запрошенная операция отменена. Превышен предел памяти.
553	Запрошенная операция не принята. Неправильное имя файла.

5.3. Последовательность команд и ответов

Разрешенные типы ответов сервера на команды клиента определены в табл. 3. Первыми приведены предварительные ответы (с допустимыми успешными ответами под ними), затем положительные и отрицательные окончательные ответы и в конце - промежуточные ответы с остающимися командами.

Таблица 3

Последовательность команд и ответов

Операция	Команда	Коды разрешенных ответов
Установление соединения		120
		220
		220
		421
Login	USER	230
		530
		500, 501, 421
		331
	PASS	230
		202
		530
		500, 501, 503, 421
		332
	ACCT	230
		202
		530
		500, 501, 503, 421
	CWD	250
		500, 501, 502, 421, 530, 550
	CDUP	200
		500, 501, 502, 421, 530, 550
	SMNT	202, 250
		500, 501, 502, 421, 530, 550
Logout	REIN	120
		220
		220
		421
		500, 502
	QUIT	221
		500
Transfer parameters	PORT	200
		500, 501, 421, 530
	PASV	227
		500, 501, 502, 421, 530
	MODE	200
		500, 501, 504, 421, 530
	TYPE	200
		500, 501, 504, 421, 530
	STRU	200
		500, 501, 504, 421, 530

File action commands	ALLO	200
		202
		500, 501, 504, 421, 530
	REST	500, 501, 502, 421, 530
		350
	STOR	125, 150
		(110)
		226, 250
		425, 426, 451, 551, 552
		532, 450, 452, 553
		500, 501, 421, 530
	STOU	125, 150
		(110)
		226, 250
		425, 426, 451, 551, 552
		532, 450, 452, 553
		500, 501, 421, 530
	RETR	125, 150
		(110)
		226, 250
		425, 426, 451
		450, 550
		500, 501, 421, 530
	LIST	125, 150
		226, 250
		425, 426, 451
		450
		500, 501, 502, 421, 530
	NLST	125, 150
		226, 250
		425, 426, 451
		450
		500, 501, 502, 421, 530
	APPE	125, 150
		(110)
		226, 250
		425, 426, 451, 551, 552
		532, 450, 550, 452, 553
		500, 501, 502, 421, 530
	RNFR	450, 550
		500, 501, 502, 421, 530
		350

File action commands	RNTO	250
		532, 553
		500, 501, 502. 503. 421, 530
	DELE	250
		450, 550
		500, 501, 502, 421, 530
	RMD	250
		500, 501, 502, 421, 530, 550
	MKD	257
		500, 501, 502, 421, 530, 550
	PWD	257
		500, 501, 502, 421, 550
	ABOR	225, 226
		500, 501, 502, 421
Informational commands	SYST	215
		500, 501, 502, 421
	STAT	211, 212, 213
		450
		500, 501, 502, 421, 530
	HELP	211, 214
		500, 501, 502, 421
Miscellaneous commands	SITE	200
		202
		500, 501, 530
	NOOP	200
		500, 421

6. СИНТАКСИС КОМАНД FTP

USER <SP> <username> <CRLF>
PASS <SP> <password> <CRLF>
ACCT <SP> <account-information> <CRLF>
CWD <SP> <pathname> <CRLF>
CDUP <CRLF>
SMNT <SP> <pathname> <CRLF>
QUIT <CRLF>
REIN <CRLF>
PORT <SP> <host-port> <CRLF>
PASV <CRLF>
TYPE <SP> <type-code> <CRLF>
STRU <SP> <structure-code> <CRLF>
MODE <SP> <mode-code> <CRLF>

RETR <SP> <pathname> <CRLF>
STOR <SP> <pathname> <CRLF>
STOU <CRLF>
APPE <SP> <pathname> <CRLF>
ALLO <SP> <decimal-integer>
[<SP> R <SP> <decimal-integer>] <CRLF>
REST <SP> <marker> <CRLF>
RNFR <SP> <pathname> <CRLF>
RNTO <SP> <pathname> <CRLF>
ABOR <CRLF>
DELE <SP> <pathname> <CRLF>
RMD <SP> <pathname> <CRLF>
MKD <SP> <pathname> <CRLF>
PWD <CRLF>
LIST [<SP> <pathname>] <CRLF>
NLST [<SP> <pathname>] <CRLF>
SITE <SP> <string> <CRLF>
SYST <CRLF>
STAT [<SP> <pathname>] <CRLF>
HELP [<SP> <string>] <CRLF>
NOOP <CRLF>

Определения аргументов команд

<username> ::= <string>
<password> ::= <string>
<account-information> ::= <string>
<string> ::= <char> | <char><string>
<char> ::= любой из 128 символов ASCII кроме <CR> and <LF>
<marker> ::= <pr-string>
<pr-string> ::= <pr-char> | <pr-char><pr-string>
<pr-char> ::= печатные символы, любой код ASCII от 33 до 126
<byte-size> ::= <number>
<host-port> ::= <host-number>,<port-number>
<host-number> ::= <number>,<number>,<number>,<number>
<port-number> ::= <number>,<number>
<number> ::= любое десятичное целое от 1 до 255
<form-code> ::= N | T | C
<type-code> ::= A [<sp> <form-code>]
| E [<sp> <form-code>]
| I
| L <sp> <byte-size>
<structure-code> ::= F | R | P
<mode-code> ::= S | B | C
<pathname> ::= <string>
<decimal-integer> ::= любое десятичное целое

7. ДИАГРАММЫ СОСТОЯНИЙ

7.1 Команды ABOR, ALLO, DELE, CWD, CDUP, SMNT, HELP, NOOP, PASV, QUIT, SITE, PORT, SYST, STAT, RMD, MKD, PWD, STRU, TYPE.

Диаграмма состояний сервера FTP для команд ABOR, ALLO, DELE, CWD, CDUP, SMNT, HELP, NOOP, PASV, QUIT, SITE, PORT, SYST, STAT, RMD, MKD, PWD, STRU, TYPE приведены на рис.4 Используемые обозначения:

B - Начало
W - Ожидание
S - Успешное выполнение
F - Восстановимая ошибка
E - Невосстановимая ошибка

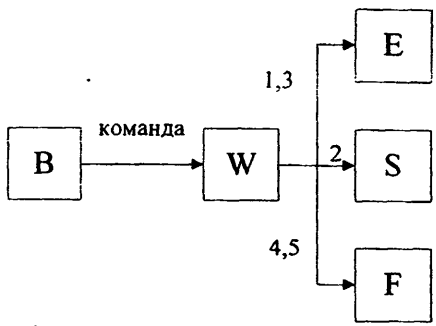


Рис. 4. Диаграмма состояний сервера FTP для команд ABOR, ALLO, DELE, CWD, CDUP, SMNT, HELP, NOOP, PASV, QUIT, SITE, PORT, SYST, STAT, RMD, MKD, PWD, STRU, TYPE

7.2. Команды APPE, LIST, NLST, REIN, RETR, STOR, STOU

Диаграмма состояний сервера FTP для команд APPE, LIST, NLST, REIN, RETR, STOR, STOU приведены на рис.5. Используемые обозначения:

B - Начало
W - Ожидание
S - Успешное выполнение
F - Восстановимая ошибка
E - Невосстановимая ошибка

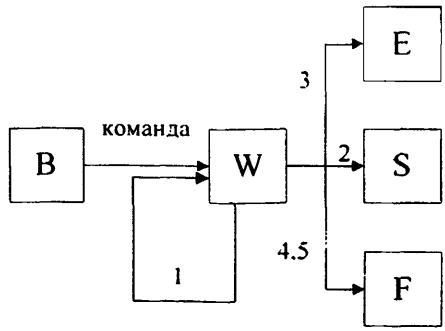


Рис. 5. Диаграмма состояний сервера FTP для команд APPE, LIST, NLST, REIN, RETR, STOR, STOU

7.3. Команды RNFR и RNTO

Диаграмма состояний сервера FTP для команд RNFR и RNTO приведены на рис.6.
Используемые обозначения:

B - Начало
W - Ожидание
S - Успешное выполнение
F - Восстановимая ошибка
E - Невосстановимая ошибка
M - Среднее состояние

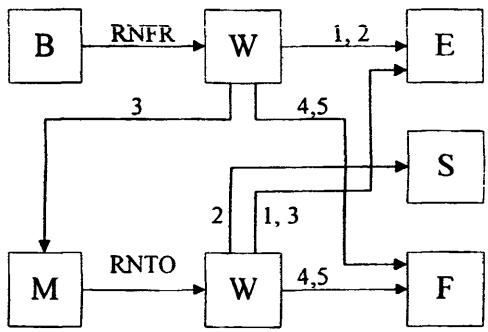


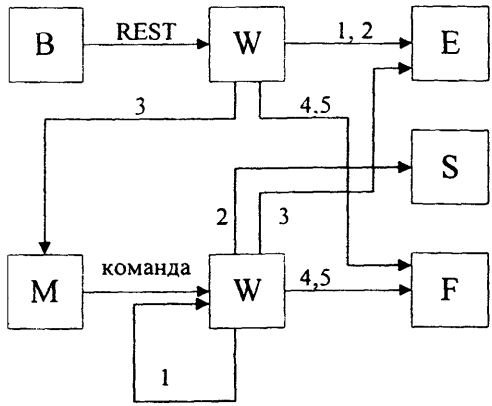
Рис. 6. Диаграмма состояний сервера FTP для команд RNFR и RNTO

7.4. Команда REST

Диаграмма состояний сервера FTP для команды REST приведена на рис.7.
Используемые обозначения:

B - Начало
W - Ожидание
S - Успешное выполнение

F - Восстановимая ошибка
F - Невосстановимая ошибка
M - Среднее состояние



Обозначения: команда - APPE, STOR или RETR

Рис. 7. Диаграмма состояний сервера FTP для команды REST

7.5. Команды USER, PASS, ACCT

Диаграмма состояний сервера FTP для команд USER, PASS, ACCT приведена на рис.8. Используемые обозначения:

B - Начало
W - Ожидание
S - Успешное выполнение
F - Восстановимая ошибка
E - Невосстановимая ошибка
M - Среднее состояние

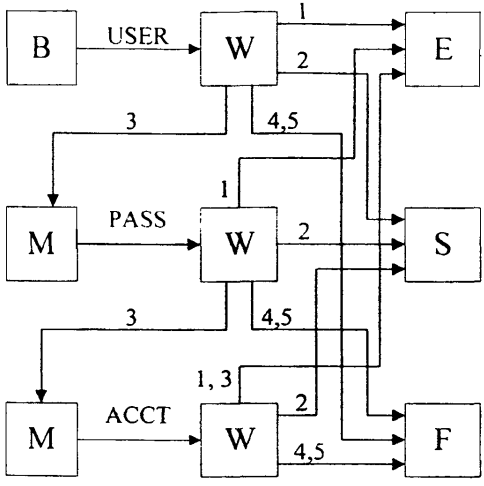


Рис. 8. Диаграмма состояний сервера FTP для команд USER, PASS, ACCT

gost.e-ecolog.ru