Network Working Group
Request for Comments: 2060
Obsoletes: 1730
Category: Standards Track

M. Crispin
University of Washington
December 1996

Протокол IMAP v.4, rev. 1

INTERNET MESSAGE ACCESS PROTOCOL - VERSION 4rev1

Статус документа

Этот документ содержит предлагаемый стандарт протокола (standards track protocol) для сообщества Internet и является запросом к обсуждению в целях развития стандарта. Информацию о текущем состоянии стандартизации протокола можно найти в Internet Official Protocol Standards (STD 1). Документ может распространяться свободно.

Аннотация

Протокол IMAP4rev1 (Internet Message Access Protocol, Version 4rev1) обеспечивает клиентам возможность доступа и манипуляций с почтовыми сообщениями на сервере. IMAP4rev1 разрешает манипуляции с удаленными папками сообщений, называемыми почтовыми ящиками (mailboxes), как с локальными почтовыми ящиками. IMAP4rev1 также позволяет offline-клиентам синхронизировать почтовые ящики с сервером (см. [IMAP-DISC]).

IMAP4rev1 включает операции создания, удаления и переименования почтовых ящиков, проверки наличия новых сообщений, удаления сообщений навсегда, установки и сброса флагов, разбора [RFC-822] и [MIME-IMB]; поиска и селективной выборки атрибутов сообщений, текста или их фрагментов. Доступ к сообщениям в IMAP4rev1 обеспечивается по номерам сообщений, которые просто присваиваются по порядку или задаются с обеспечением уникальности каждого номера.

Протокол IMAP4rev1 поддерживает один сервер. Механизм настройки конфигурации для поддержки множества серверов IMAP4rev1 обсуждается в работе [ACAP].

IMAP4rev1 не поддерживает функций передачи почты, для решения таких задач служат транспортные почтовые протоколы типа [SMTP].

При разработке IMAP4rev1 учитывалось требование совместимости с [IMAP2] и неопубликованным протоколом IMAP2bis. По мере разработки IMAP4rev1 от некоторых возможностей ранних версий протоколов пришлось отказаться. Устаревшие команды, отклики и форматы данных, с которыми реализация IMAP4rev1 может встретиться при работе с ранними версиями, описаны в [IMAP-OBSOLETE]. Другие вопросы совместимости с IMAP2bis (наиболее распространенный вариант ранних протоколов) обсуждаются в работе [IMAP-COMPAT]. Полное рассмотрение вопросов совместимости с редкими (и практически вымершими) вариантами [IMAP2] приводится в [IMAP-HISTORICAL] - этот документ представляет в основном исторический интерес.

Оглавление

1. Как работать с документом

1.1. Структура документа

Этот документ представляет протокол с точки зрения разработчика клиента или сервера IMAP4rev1. Кроме обзора главе 2, документ не содержит каких-либо материалов, помогающих понять работу протокола. Главы 3 - 5 содержат общий контекст и определения, используемые IMAP4rev1.

В главах 6, 7 и 9 описаны команды, отклики и синтаксис, соответственно. Соотношения между ними таковы, что почти невозможно разобраться с протоколом, рассматривая команды, отклики и синтаксис по отдельности. В частности, не следует пытаться вывести синтаксис команд, прочтя лишь их описание - обратитесь к главе, описывающей синтаксис.

1.2. Используемые соглашения

В примерах обозначения C: и S: показывают строки, передаваемые клиентом и сервером, соответственно. Термины, используемые для обозначения уровня требований, растолкованы ниже.

1. MUST - **необходимо**

   Это слово, а также термины **требуется** (**обязательно**) и **нужно** (SHALL) используется для требований, которые являются абсолютно необходимыми в данной спецификации.

2. MUST NOT - **недопустимо**

   Эта фраза или слова SHALL NOT (**не позволяется**) означают абсолютный запрет в рамках спецификации.

3. SHOULD - **следует**

   Это слово, а также глагол **рекомендуется** (RECOMMENDED) используется для обозначения требований, от выполнения которых можно отказаться при наличии разумных причин. Однако при таком отказе следует помнить о возможных проблемах в результате отказа и принимать взвешенное решение.

4. SHOULD NOT - **не следует**

   Эта фраза и глагол **не рекомендуется** (NOT RECOMMENDED) используются применительно к особенностям или функциям, которые допустимы и могут быть полезными, но могут вызывать проблемы. При реализации таких опций следует учитывать возможность возникновения проблем и принимать взвешенное решение.

5. MAY - **возможно**

   Это слово, а также прилагательное **необязательный** (**дополнительно**) обозначают элементы, реализация которых является необязательной. Одни разработчики могут включать такие опции в свою продукцию для расширения возможностей, а другие - опускать в целях упрощения. Реализация, не включающая ту или иную опцию, **должна** быть готова к работе с реализациями, которые используют эту опцию (возможно совместная работа будет обеспечиваться за счет некоторого ущерба функциональности). Включающие опцию реализации должны быть готовы (естественно, без использования такой опции) к взаимодействию с реализациями, которые такую опцию не поддерживают.

Термин «пользователь» (User) используется для обозначения человека, а слово «клиент» обозначает пользовательские программы.

Термин «соединение» (Connection) описывает всю последовательность операций взаимодействия между клиентом и сервером от момента организации связи и до ее завершения. Термины «сеанс» или «сессия» (Session) обозначают последовательность операций обмена между клиентом и сервером с момента выбора почтового ящика (команда SELECT или EXAMINE) и до отмены этого выбора (команда SELECT или EXAMINE для другого почтового ящика, команда CLOSE или разрыв соединения).

Символом будем называть 7-битовый символ из набора US-ASCII, если явно не указано иное. Другие символьные наборы обозначаются с использованием CHARSET, как описано в [MIME-IMT] и определено в [CHARSET]. Опция CHARSET имеет дополнительную семантику, описанную в вышеупомянутых работах.

2. Обзор

2.1. Канальный уровень

Протокол IMAP4rev1 предполагает наличие гарантированного потока данных (например, от TCP). При использовании транспортного протокола TCP сервер IMAP4rev1 прослушивает порт 143.

2.2. Команды и отклики

Соединение IMAP4rev1 включает организацию связи между клиентом и сервером, стартовое приветствие сервера и операции взаимодействия между клиентом и сервером (команды клиента, данные от сервера, отклики сервера).

Все взаимодействие между клиентом и сервером происходит в форме обмена текстовыми строками, завершающимися последовательностью CRLF. Получатель IMAP4rev1 (клиент или сервер) читает строку или последовательность из заданного числа октетов, следующих за ней.

2.2.1. Передатчик клиента и приемник сервера

Взаимодействие начинается с команды клиента. Каждая команда начинается с префикса-идентификатора (обычно короткая последовательность букв и цифр типа A0001, A0002 и т. п.), называемого тегом (tag). Теги генерируются клиентом для каждой команды.

Существует два случая, когда строка от клиента не содержит полной команды - в одном случае сопровождаются счетчиком октетов (см. описание String в главе 4. Форматы данных), в другом - аргументы команды требуют отклика от сервера (см. описание команды AUTHENTICATE). В обоих случаях сервер передает в ответ на команду

соответствующий отклик с запросом продолжения команды (если сервер готов к приему имеющихся у клиента данных и оставшейся части команды)[1]. Эти отклики начинаются с префикса "+".

Приемник IMAP4rev1 на сервере читает строку команды от клиента, разбирает ее, отделяя команду от аргументов, и передает серверу, а тот генерирует отклик о завершении команды.

2.2.2. Приемник клиента и передатчик сервера

Данные, передаваемые сервером клиенту и отклики о состоянии, не показывающие завершение команды, передаются с префиксом "*" и называются откликами без пометок (untagged).

Данные от сервера **могут** передаваться в результате клиентской команды, а **могут** генерироваться по инициативе сервера. Между этими данными нет никакой синтаксической разницы.

Отклик от сервера о завершении команды показывает результат операции. Он помечается тегом клиентской команды, послужившей началом операции. Таким образом, при одновременной обработке множества команд тег в отклике сервера позволяет идентифицировать команду, с которой связан отклик. Существуют три варианта откликов о завершении команды - OK (успешное завершение), NO (неудача) и BAD (ошибка протокола - например, нераспознанная команда или некорректный синтаксис).

Приемник IMAP4rev1 у клиента читает строку отклика от сервера и предпринимает в ответ на нее действия, в зависимости от префикса в отклике (тег, "*" или "+"). Клиент **должен** быть постоянно готов к восприятию любых откликов от сервера. Принятые от сервера данные **следует** записывать, чтобы впоследствии клиент мог обращаться к сохраненным данным, не запрашивая их у сервера повторно. В некоторых случаях данные **должны** записываться. Этот вопрос более подробно рассматривается ниже (см. 7. Отклики сервера).

2.3. Атрибуты сообщения

Кроме текста каждое сообщение содержит набор связанных с ним атрибутов. Эти атрибуты можно отыскивать вместе с текстом сообщения или независимо от него.

2.3.1. Номера сообщений

Доступ к сообщениям IMAP4rev1 обеспечивается по одному из двух номеров - идентификатору сообщения или уникальному идентификатору.

2.3.1.1. Уникальный идентификатор сообщения (UID)

32-битовый идентификатор, присваиваемый каждому сообщению и совместно с уникальным идентификатором корректности (см. ниже) образующий 64-битовое значение, для которого гарантируется уникальность в масштабе почтового ящика. Уникальные номера для каждого почтового ящика выделяются строго по нарастанию - каждое новое сообщение имеет значение UID, превышающее идентификаторы добавленных ранее сообщений.

В отличие от порядковых номеров уникальные идентификаторы не обязаны следовать непрерывно и сохраняются только в течение одного сеанса. Это позволяет клиенту ресинхронизировать свое состояние по отношению к предыдущему сеансу (например, для отключенных или сеансовых клиентов). Вопросы синхронизации обсуждаются в работе [IMAP-DISC].

С каждым почтовым ящиком связывается уникальный идентификатор корректности, который передается в коде UIDVALIDITY неотмеченных откликов OK при выборе почтового ящика. Если уникальные идентификаторы из предыдущего сеанса не могут использоваться в данной сессии, новое значение уникального идентификатора корректности **должно** быть больше, чем использованное в предыдущем сеансе значение.

> Примечание: Значения уникальных идентификаторов для почтового ящика **должны** строго возрастать. Если физически сообщения сохраняются агентом, отличным от IMAP, в ином порядке или неупорядочены, это требует регенерации уникальных идентификаторов для почтового ящика, поскольку прежние уникальные идентификаторы не будут идти строго по возрастанию в результате изменения порядка. Другим случаем регенерации уникальных идентификаторов являются системы, в которых при хранении сообщений не поддерживается механизм уникальной идентификации. Признавая, что это может быть неприемлемо для некоторых серверных сред, настоящая спецификация **настоятельно рекомендует** использовать системы хранения сообщений, позволяющие избежать этой проблемы.

Другой причиной невозможности использования прежних идентификаторов является случай, когда почтовый ящик удаляется и вместо него позднее создается новый ящик с таким же именем. Поскольку имена ящиков совпадают, клиент может не знать о том, что это новый ящик, если уникальные идентификаторы корректности не отличаются. Эффективным выходом будет использование в качестве уникальных идентификаторов корректности 32-битовое представление даты и времени создания почтовых ящиков. Можно использовать и константы (например, 1), если обеспечивается невозможность их повторного использования даже в случаях удалением или переименования почтовых ящиков с последующим созданием ящика с прежним именем.

Уникальный идентификатор сообщения **недопустимо** изменять в течение сеанса и **не следует** менять между сессиями. Однако, если для следующего сеанса сохранение уникального идентификатора невозможно, новое значение **должно** быть больше предыдущего.

---

[1]Если сервер обнаруживает в команде ошибку, он передает соответствующий отклик (BAD completion response) с тегом, соответствующим команде (как описано ниже) для отказа от выполнения команды и предотвращения передачи клиентом продолжения.
Сервер может также передавать отклики о завершении некоторых других команд (если обрабатывается множество команд одновременно) или неотмеченные (untagged) данные. В любом случае запрос н продолжение команды сохраняется и клиент в ответ на этот запрос выполняет соответствующие действия. Во всех случаях клиент **должен** передать всю команду (включая все отклики на запросы продолжения команды и сами команды продолжения) до инициализации новой команды.

2.3.1.2. Порядковый номер сообщения

Относительные номера сообщений (от 1 до текущего числа писем в ящике). Номера **должны** упорядочиваться в соответствии с увеличением уникальных идентификаторов. При добавлении каждого нового сообщения ему присваивается следующий свободный номер (на 1 больше последнего из использованных номеров).

Порядковые номера сообщений могут изменяться в течение сеанса. Например, при полном удалении письма из ящика порядковые номера всех последующих сообщений уменьшаются.

Кроме идентификации сообщений в смысле доступа к ним, порядковые номера могут использоваться в расчетах. Например, при получении неотмеченного отклика EXISTS 11 после того, как был получен неотмеченный отклик 8 EXISTS, позволяет вычислить, что были получены три новых сообщения - 9, 10, 11. Другой пример: если сообщение с номером 287 в ящике, содержащем 523 письма, имеет UID 12345, значит 286 сообщений имеют меньшие значения UID, а 236 - большие UID.

2.3.2. Флаги сообщения

С сообщением связывается список (возможно пустой) именованных маркеров (флагов). Установка флагов обеспечивается их включением в список, сброс - удалением из списка. Протокол IMAP4rev1 поддерживает два типа флагов - для сессии и постоянные.

Имена системных флагов определяются настоящей спецификацией (см. таблицу). Все системные флаги начинаются с символа "\". Некоторые из системных флагов (\Deleted и \Seen) используют специальную семантику.

\Seen	Сообщение было прочитано.
\ Answered	На сообщение был отправлен ответ.
\Flagged	Флаг важности сообщения.
\Deleted	Сообщение «удалено» для последующего уничтожения с помощью EXPUNGE.
\Draft	Сообщение не закончено (помечено как черновик - draft).
\Recent	Сообщение недавно доставлено в почтовый ящик. Этот сеанс является первым, в котором это сообщение появилось. Следующие сеансы не будут устанавливать флаг \Recent для этого сообщения. Это флаг не может быть изменен клиентом. Если невозможно определить, является ли сеанс для данного сообщения первым, сообщение **следует** рассматривать как новое (установить флаг). Если с одним почтовым ящиком одновременно организовано более одного соединения, невозможно определить для каких сеансов будет установлен флаг \Recent.

Имена флагов задаются разработчиками сервера и не должны начинаться с символа "\". Серверы **могут** позволять клиенту определять новые имена флагов в почтовом ящике (см. описание отклика PERMANENTFLAGS).

Флаги могут быть постоянными или сеансовыми (для каждого флага это определяется независимо). К постоянным относятся те флаги, клиент сохраняются после завершения сеанса, т. е. в следующем сеансе они будут иметь те же значения. Изменения сеансовых флагов сохраняются только в течение сеанса. Флаг \Recent представляет собой специальный случай сеансового системного флага (этот флаг не может использоваться в качестве аргумента команды STORE).

2.3.3. Внутренняя дата сообщения

Внутренние значения даты и времени сообщения на сервере. Это не дата и время заголовка [RFC-822], а дата и время приема сообщения. Для сообщений, доставленных по протоколу SMTP, в этот атрибут **следует** включать дату и время окончательной доставки сообщения, в соответствии с [SMTP]. Для сообщений, доставленных с помощью команды IMAP4rev1 COPY, в этот атрибут **следует** включать внутренние значения даты и времени отправителя. Для сообщений, поступивших в результате использования команды IMAP4rev1 APPEND, в этот атрибут **следует** включать дату и время, указанные в описании команды APPEND. Во всех других случаях значение атрибута определяется используемой реализацией.

2.3.4. Размер сообщения [RFC-822]

Число октетов в сообщении, выраженное в формате [RFC-822].

2.3.5. Структура конверта

Разобранное представление информации из конверта [RFC-822] (не следует путать с конвертом [SMTP] сообщения.

2.3.6. Структура тела сообщения (Body Structure)

Разобранное представление структуры информации [MIME-IMB] в сообщении.

2.4. Текст сообщения

Кроме извлечения полного текста сообщения [RFC-822] протокол IMAP4rev1 позволяет извлекать отдельные части текста (в честности, заголовок [RFC-822], тело сообщения [RFC-822], часть тела [MIME-IMB] заголовок [MIME-IMB]).

3. Состояния сервера и команды

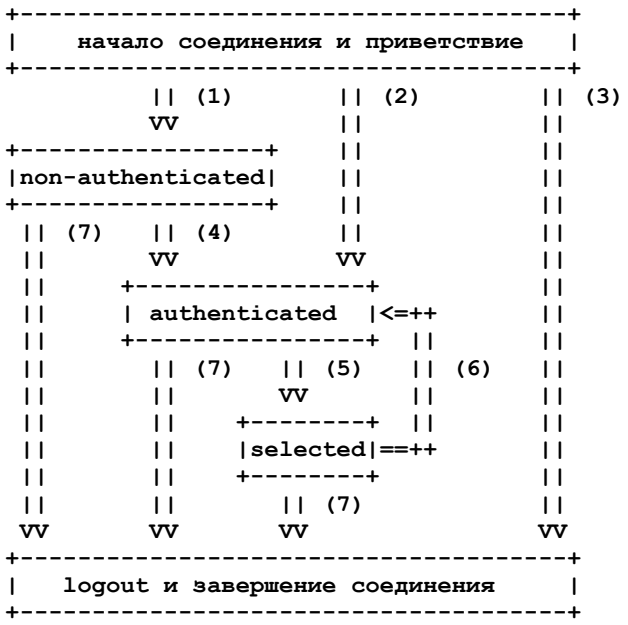
Сервер IMAP4rev1 может находиться в одном из 4 состояний. Большая часть команд применима в любом из состояний сервера. Если клиент пытается использовать недоступную в данном состоянии сервера команду, возникает протокольная ошибка и сервер возвращает отклик BAD или NO (в зависимости от реализации сервера).

3.1. Состояние Non-Authenticated

В этом состоянии клиент **должен** свои полномочия прежде, чем передавать какие-либо команды. Это состояние является первым после организации соединения, если соединение не аутентифицировано заранее.

3.2. Состояние Authenticated

В этом состоянии полномочия клиента уже проверены и клиент **должен** выбрать почтовый ящик для работы, прежде, чем подавать серверу какие-либо команды для работы с сообщениями. Это состояние возникает при организации сеанса клиентом с предварительной аутентификацией или после предъявления клиентом корректных полномочий, а также после ошибок при выборе почтового ящика.



3.3. Состояние Selected

В этом состоянии уже выбран почтовый ящик для работы.

3.4. Состояние Logout

В этом состоянии соединение завершается и сервер разрывает связь. Данное состояние может вводиться по запросу клиента или по инициативе сервера.

(1) Соединение без предварительной аутентификации (приветствие OK)

(2) Соединение с предварительной аутентификацией (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 ошибка при выполнении команды SELECT или EXAMINE

(7) Команда LOGOUT, отключение (shutdown) сервера или разрыв соединения.

4. Форматы данных

Протокол IMAP4rev1 использует текстовые команды и отклики. Данные IMAP4rev1 могут передаваться в оной из описанных ниже форм: атом (atom), число, строка, список в скобках, NIL.

4.1. Атом (Atom)

Атом состоит из одного или нескольких символов общего назначения.

4.2. Число

Число содержит одну или более десятичных цифр, представляющих числовое значение.

4.3. Строка

Строки могут использовать одну из двух возможных форм - литеральную (буквальную - literal) или квотированную (строка в кавычках - quoted string). Литеральная форма представляет собой общую форму представления строк. Квотированные строки позволяют упростить обработку за счет сужения используемого в них набора символов.

Литеральная строка представляет собой последовательность (возможно, пустую) октетов (включая символы CR и LF), начинающуюся со счетчика октетов в форме {число октетов} за которым следуют символы CRLF. Для строк, передаваемых сервером клиенту, октеты данных следуют сразу же после CRLF. Для строк, передаваемых от клиента к серверу, клиент **должен** сначала получить от сервера подтверждение с запросом на продолжение команды (эти запросы описаны ниже) и только после этого передавать данные и оставшуюся часть команды.

Квотированная строка представляет собой последовательность (возможно, пустую) 7-битовых символов, включая CR и LF, заключенную в кавычки (").

Пустые строки представляются как "" (только кавычки) или {0}CRLF (литеральная строка с 0 октетов)[1].

4.3.1. 8-битовые и двоичные строки

8-битовые текстовые и бинарные данные поддерживаются с помощью транспортного кодирования содержимого [MIME-IMB]. Реализации IMAP4rev1 **могут** передавать 8-битовые и мультиоктетные символы в литералах, но делать это **следует** только в случаях задания [CHARSET] (набор символов).

Хотя бинарные данные определены спецификацией, их некодированная передача не допускается. Бинарными называют строки с символами NUL. Реализации **должны** преобразовывать бинарные данные в текстовый формат (например, BASE64) до их передачи. Строки с большим числом символов CTL, также **могут** считаться бинарными.

4.4. Список в скобках (Parenthesized List)

Структуры данных представляются в форме заключенных в скобки списков (parenthesized list) - последовательность элементов, разделенных пробелами, ограничивается с обеих сторон скобками. Списки могут содержать внутри себя другие списки в скобках.

Пустой список представляется в виде ().

4.5. Пустой формат (NIL)

Специальный атом NIL используется для представления несуществующих данных определенного типа, представляемых как строка или список в скобках, в отличие от пустой строки - "" или списка ().

5. Работа протокола

5.1. Именование почтовых ящиков

Интерпретация имен почтовых ящиков зависит от реализации. Однако почтовый ящик с независимым от регистра именем INBOX зарезервирован в качестве основного почтового ящика пользователя на данном сервере.

5.1.1.Иерархия имен

Желательно экспортировать иерархические имена почтовых ящиков - имена **должны** записываться со снижением уровня иерархии слева направо и односимвольным разделителем между иерархическими уровнями. Разделитель для всех переходов между уровнями должен быть один.

5.1.2. Соглашение о пространстве имен

По этому соглашению первый иерархический элемент любого имени почтового ящика должен начинаться с символа #, идентифицирующего пространство имен (namespace) остальной части имени. Это позволяет различать разные типы почтовых хранилищ (mailbox store), каждый из которых имеет свое пространство имен.

Например, реализация, предлагающая доступ к конференциям USENET **может** использовать пространство #news для отделения имен конференций USENET от других почтовых ящиков. В результате конференция comp.mail.misc будет иметь почтовый ящик #news.comp.mail.misc, а имя comp.mail.misc может указывать на иной объект (например, почтовый ящик пользователя).

5.1.3. Соглашение об использовании других языков в именах

В соответствии с этим соглашение имена почтовых ящиков могут использовать символы других языков с обновленной версией кодирования UTF-7, описанной в работе [UTF-7]. Обновление было предпринято с целью разрешения перечисленных ниже проблем UTF-7:

1. UTF-7 использует знак "+" для смещения (shifting), что не позволяет применять "+" в именах почтовых ящиков (в частности, с названиях конференций USENET.
2. UTF-7 использует кодирование BASE64, применение символа "/" в котором конфликтует с использованием "/" в качестве разделителя иерархических уровней.
3. UTF-7 запрещает некодированное представление символа "\", что не позволяет использовать его в качестве разделителя иерархических уровней.
4. UTF-7 запрещает некодированное представление символа "~", используемого в некоторых серверах в качестве индикатора домашнего каталога.
5. UTF-7 допускает множество альтернативных форм представления одной строки (в частности, печатные символы US-ASCII можно представлять в кодированной форме).

В обновленном варианте UTF-7 печатные символы US-ASCII (за исключением "&") представляют себя, т. е., символы с кодами 0x20 - 0x25 и 0x27 - 0x7e. Символ "&" (0x26) представляется 2-октетной последовательностью "&-". Все прочие символы (0x00 - 0x1f, 0x7f - 0xff, 16-битовые символы Unicode) представляются с использованием модифицированного кода BASE64 (в соответствии с [UTF-7] "," используется взамен "/"). Обновленное кодирование BASE64 **недопустимо** использовать для представления любых печатных символов US-ASCII. Символ "&" используется для перехода к модифицированному кодированию BASE64, а "-" для перехода к US-ASCII. Все имена начинаются с символа US-ASCII и **должны** заканчиваться символом US-ASCII (т. е., имя из символов Unicode **должно** завершаться знаком "- "). В качестве примера приведем имя, содержащее текст на английском, японском и китайском языках - ~peter/mail/&ZeVnLIqe-/&U,BTFw-

[1] Клиент **должен** получать запрос на продолжение даже при передаче литеральной строки нулевой длины.

5.2. Обновление размеров и статуса сообщений

В любой момент сервер может передать данные, которых клиент не запрашивал. Иногда такая передача может даже **требоваться**. Например, агенты (не сервер) **могут** добавлять сообщения в почтовый ящик (например, доставка новой почты), менять флаги сообщений в почтовых ящиках (например, одновременный доступ к почтовому ящику нескольких агентов) и даже удалять сообщения. Сервер **должен** передавать обновленные данные о размере почтового ящика, если таковые обнаружены в процессе обработки команды. Серверу **следует** автоматически передавать обновленную информацию о флагах без явного запроса на такое обновление со стороны клиента. Для передачи уведомлений об удалении почты существуют специальные правила, позволяющие избежать ошибок при синхронизации (см. 7.4.1. Отклик EXPUNGE).

Независимо от используемого в реализации клиента способа запоминания полученных от сервера данных, клиент **должен** записывать обновления размера почтового ящика. **Недопустимо** предполагать, что любая команда после начального выбора почтового ящика будет возвращать его размер.

5.3. Отклики при отсутствии обрабатываемых команд

Серверу разрешается передавать непомеченные отклики (за исключением EXPUNGE) при отсутствии обрабатываемых команд. Реализации серверов, передающие такие отклики, **должны** принимать во внимание вопросы управления потоком данных. В частности, сервер **должен** (1) убедиться, что размер передаваемых данных не превышает максимальный размер окна, доступный нижележащему транспортному уровню или (2) использовать запись без блокировки (non-blocking write).

5.4. Таймер автоматического отключения

Если сервер использует автоматическое отключение по таймеру в случае отсутствия активности, значение таймера автоматического отключения **должно** быть не менее 30 минут. Получив от клиента **любую** команду, серверу **следует** сбросить таймер.

5.5. Обработка множества команд

Клиент **может** передавать новые команды, не дожидаясь отклика о завершении обработки предыдущих команд, с учетом правил однозначности (см. ниже) и управления потоком данных на нижележащем уровне. Сервер **может** начать обработку новой команды до завершения обработки прежних, учитывая правила однозначности. Однако все отклики с запросами на продолжение команды и продолжения команд **должны** быть согласованы до того, как будет передана следующая команда.

Правило однозначности связано с возможной зависимостью результатов выполнения команды от результатов завершения предшествующих команд. Для клиентов **недопустимо** передавать команду без ожидания, если результаты могут быть неоднозначными. Если сервер видит возможность неоднозначности, он должен выполнять команды в заданном клиентом порядке.

Тривиальным случаем возникновения неоднозначности является ситуация, когда результаты выполнения одной команды напрямую воздействуют на результат другой (например, команды FETCH и STORE для флагов сообщения). Менее тривиальные ситуации, для которых допустимы непомеченные отклики EXPUNGE (все команды, за исключение FETCH, STORE, SEARCH), поскольку непомеченный отклик EXPUNGE может делать некорректными порядковые номера в следующей команде. Это не вызывает проблем для команд FETCH, STORE, SEARCH, поскольку для этих команд запрещена передача откликов EXPUNGE, пока не будет завершена обработка всех команд. Следовательно, если клиент передаст любую из команд, кроме FETCH, STORE, SEARCH, он **должен** дождаться отклика и только после этого может передавать команды с порядковыми номерами сообщений.

Ниже приведены несколько примеров некорректных последовательностей команд:

```
FETCH + NOOP + STORE
STORE + COPY + FETCH
COPY + COPY
CHECK + FETCH
```

Приведенные ниже последовательности команд можно передавать, не дожидаясь откликов:

```
FETCH + STORE + SEARCH + CHECK
STORE + COPY + EXPUNGE
```

6. Команды клиента

В этой главе описаны команды протокола IMAP4rev1. Все команды организованы по состояниям, для которых возможно использование этих команд (например, команды, корректные для состояний authenticated и selected, перечислены в числе команд для состояния authenticated).

Аргументы команд указываются ключевым словом "Аргументы:" в описании команды и рассмотрены здесь только с функциональной точки зрения, а синтаксис аргументов описан в главе «Формальный синтаксис».

Некоторые команды ведут к генерации сервером специфического отклика - эти отклики помечены ключевым словом "Отклик:" в описании команды. Дополнительная информация об откликах содержится в глава «Отклики» и «Формальный синтаксис». По любой команде клиента сервер может передавать в ответ данные.

Ключевое слово "Результат:" в описании команды указывает на возможные помеченные (tagged) отклики о состоянии и специфическую интерпретацию таких откликов.

6.1. Команды клиента для любого состояния

Команды CAPABILITY, NOOP и LOGOUT могут использоваться во всех состояниях.

6.1.1. Команда CAPABILITY

Аргументы: не используются

Отклик: **требуется** непомеченный отклик CAPABILITY

Результат: OK - успешное завершение

BAD - команда не поддерживается.

Команда CAPABILITY запрашивает у сервера список поддерживаемых возможностей. Сервер **должен** передавать один непомеченный отклик CAPABILITY, указывая в списке возможностей по крайней мере IMAP4rev1, и только поток возвращать помеченный отклик OK. Возвращаемый список возможностей не зависит от пользователя или состояния соединения, следовательно, не возникает необходимости использования команды CAPABILITY более одного раза в каждом соединении.

Имена возможностей, начинающихся с AUTH=, показывают, что сервер использует механизм аутентификации. Все такие имена, по определению, являются частью данной спецификации. Например, аутентификация с помощью экспериментального метода blurdybloop будет обозначаться возможностью AUTH=XBLURDYBLOOP, но не XAUTH=BLURDYBLOOP или XAUTH=XBLURDYBLOOP.

Имена других возможностей указывают на расширения, пересмотры или изменения данной спецификации. Дополнительная информация приведена ниже при описании откликов CAPABILITY. В данной спецификации не определяется никаких дополнительных возможностей сверх базовой возможности IMAP4rev1, поддерживаемой автоматически без каких-либо запросов со стороны клиента.

Информация о расширениях и экспериментальных возможностях приводится в параграфе Ошибка: источник перекрёстной ссылки не найден.

Пример

```
C: abcd CAPABILITY
S: * CAPABILITY IMAP4rev1 AUTH=KERBEROS_V4
S: abcd OK CAPABILITY completed
```

6.1.2. Команда NOOP

Аргументы: не используются

Отклик: для этой команды не существует специфических откликов (см. ниже)

Результат: OK - команда выполнена

BAD - команда не поддерживается или некорректны аргументы

Команда NOOP всегда завершается успешно, поскольку она не делает ничего. Поскольку любая команда может возвращать обновленное состояние как непомеченные данные, команду NOOP можно использовать для периодического получения обновлений во время бездействия клиента. Можно также использовать команду NOOP для сброса таймера бездействия на сервере.

Пример

```
C: a002 NOOP
S: a002 OK NOOP completed
. . .
C: a047 NOOP
S: * 22 EXPUNGE
S: * 23 EXISTS
S: * 3 RECENT
S: * 14 FETCH (FLAGS (\Seen \Deleted))
S: a047 OK NOOP completed
```

6.1.3. Команда LOGOUT

Аргументы: не используются

Отклик: Требуется непомеченный отклик BYE

Результат: OK - команда выполнена

BAD - команда не поддерживается или некорректны аргументы

Команда LOGOUT информирует сервер о намерении клиента закончить работу. Сервер **должен** передать непомеченный отклик BYE до передачи отмеченного отклика OK, а потом закрыть соединение.

Пример

```
C: A023 LOGOUT
S: * BYE IMAP4rev1 Server logging out
S: A023 OK LOGOUT completed
```

Сервер и клиент закрыли соединение.

6.2. Клиентские команды - состояние Non-Authenticated

В состоянии, когда аутентификация еще не произведена, команда AUTHENTICATE или LOGIN позволяет провести аутентификацию и перейти в состояние authenticated. Команда AUTHENTICATE обеспечивает общий механизм для различных методов аутентификации, а LOGIN использует традиционную пару «имя пользователя - пароль», передаваемую в виде текста.

Для некоторых почтовых ящиков сервер **может** позволять работу без аутентификации. По соглашения команда LOGIN может использоваться с именем пользователя anonymous (безымянный). Пароль является **обязательным**. Пароль и права доступа для анонимного пользователя зависят от реализации.

После аутентификации (включая анонимную) соединение не может быть переведено в состояние non-authenticated.

В дополнение к универсальным командам (CAPABILITY, NOOP, LOGOUT), данное состояние поддерживает команды AUTHENTICATE и LOGIN.

6.2.1. Команда AUTHENTICATE

Аргументы: имя механизма аутентификации

Отклик: могут быть запрошены дополнительные данные

Результат: OK - аутентификация завершена успешно, текущее состояние authenticated

NO - неудача при аутентификации - неподдерживаемый механизм, нет полномочий

BAD - команда не поддерживается или некорректны аргументы, аутентификации не произошло.

Команд AUTHENTICATE показывает серверу механизм аутентификации (типа описанного в работе [IMAP-AUTH]). Если сервер поддерживает этот механизм, выполняется обмен данными для проверки полномочий (аутентификации клиента). **Можно** также согласовать **дополнительный** механизм защиты для последующего обмена данными. Если запрошенный метод аутентификации не поддерживается, серверу **следует** отвергать команду AUTHENTICATE, передавая помеченные отклик NO.

Процесс аутентификации включает в себя обмен данными, состоящими из вопросов сервера и ответов клиента, - набор вопросов зависит от реализации. Вопрос сервера включает запрос продолжения команды, состоящий из маркера "+", за которым следует строка в формате BASE64. Отклик клиента представляет собой строку в коде BASE64. Если клиент хочет прервать обмен данными аутентификации, он просто передает строку, содержащую один символ "*". Получив такой ответ, сервер **должен** отвергнуть команду AUTHENTICATE и передать отмеченный отклик BAD.

Механизм защиты обеспечивает целостность и конфиденциальность соединения. Если такой механизм согласован, он применяется ко всем данным, передаваемым через соединение. Механизм защиты начинает работать сразу после получения последовательности CRLF,завершающей аутентификацию клиента и последовательности CRLF, завершающей отклик OK. После включения механизма защиты поток октетов команд и откликов обрабатывается в буферах шифрования. Содержимое каждого буфера передается через соединение как поток октетов, которому предшествует 4-октетное поле размера. Максимальный размер шифрованного буфера определяется используемым механизмом.

Механизм аутентификации является **необязательным**. Механизм защиты также **необязателен** и аутентификация **может** быть реализована без дополнительной защиты. Если команда AUTHENTICATE завершается отказом с откликом NO, клиент **может** попытаться использовать иной механизм аутентификации в следующей команде AUTHENTICATE или **может** попытаться войти в систему по команде LOGIN. Иными словами, клиент **может** запрашивать механизмы аутентификации в порядке снижения уровня предпочтений вплоть до команды LOGIN, как последней попытки.

Пример

```
S: * OK KerberosV4 IMAP4rev1 Server
C: A001 AUTHENTICATE KERBEROS_V4
S: + AmFYig==
C[1]: BAcAQU5EUkVXLkNNVS5FRFUAOCAsho84kLN3/IJmrMG+25a4DT+nZImJjnTNHJUtxAA+o0KPKf
    HEcAFs9a3CL5Oebe/ydHJUwYFdWwuQ1MWiy6IesKvjL5rL9WjXUb9MwT9bpObYLGOKi1Qh
S: + or//EoAADZI=
C: DiAF5A4gA+oOIALuBkAAmw==
S: A001 OK Kerberos V4 authentication successful
```

6.2.2. Команда LOGIN

Аргументы: имя пользователя, пароль

Отклик: для этой команды не существует специфических откликов

Результат: OK - вход в системы завершен, состояние authenticated

NO - неудача - имя пользователя или пароль отвергнуты

BAD - команда не поддерживается или некорректны аргументы

Команда LOGIN используется для представления клиента серверу и передачи пароля в виде открытого текста.

Пример

```
C: a001 LOGIN SMITH SESAME
S: a001 OK LOGIN completed
```

6.3. Команды клиента - состояние Authenticated

После успешного завершения аутентификации становятся доступными команды для работы с почтовыми ящиками как атомами. Команды SELECT и EXAMINE позволяют выбирать почтовые ящики и переводят соединение в состояние selected.

В дополнение к универсальным командам (CAPABILITY, NOOP, LOGOUT) после успешной аутентификации поддерживаются команды SELECT, EXAMINE, CREATE, DELETE, RENAME, SUBSCRIBE, UNSUBSCRIBE, LIST, LSUB, STATUS, APPEND.

[1] Этот отклик на самом деле является одной строкой.

6.3.1. Команда SELECT

Аргументы: имя почтового ящика

Отклик: **обязательно** непомеченный отклик FLAGS, EXISTS, RECENT

дополнительно OK непомеченный отклик UNSEEN, PERMANENTFLAGS

Результат: OK - выбор завершено, состояние selected

NO - отказ при выборе, состояние authenticated (нет почтового ящика или доступа к нему)

BAD - команда не поддерживается или некорректны аргументы

Команда SELECT выбирает почтовый ящик и обеспечивает доступ к хранящимся в нем сообщениям. До возврата клиенту отклика сервер должен передать клиенту следующие непомеченные данные для определения начального состояния почтового ящика:

FLAGS определяет флаги почтового ящика (см. описание отклика FLAGS).

<n> EXISTS число сообщений в почтовом ящике (см. описание отклика EXISTS).

<n> RECENT число сообщений с установленным флагом \Recent (см. описание отклика RECENT).

OK [UIDVALIDITY <n>] уникальный идентификатор корректности (см. описание отклика UID).

Серверу **следует** также передать код UNSEEN в непомеченном отклике OK, показывающий порядковый номер первого непрочитанного сообщения в почтовом ящике.

Если клиенту не разрешается вносить постоянные изменения в те или иные флаги, указанные в непомеченном отклике FLAGS, серверу **следует** передать код PERMANENTFLAGS в непомеченном отклике OK, указывающий какие флаги клиент не может изменить.

В каждом соединении может быть выбран только один почтовый ящик и для работы с несколькими ящиками требуется соответствующее число соединений. При выборе нового почтового ящика команда SELECT автоматически отменяет текущий выбор. Следовательно, если выбран почтовый ящик, то после неудачного использования команды SELECT этот выбор будет отменен.

Если клиенту позволено изменять почтовый ящик, серверу **следует** перед текстом помеченного отклика OK помещать код [READ-WRITE]. Если клиенту не разрешено менять содержимое почтового ящика, но открыт доступ для чтения почты, почтовый ящик выбирается в режиме чтения (read-only) и сервер должен перед текстом отклика OK на команду SELECT поместить код [READ-ONLY]. Доступ для чтения по команде SELECT отличается от использования команды EXAMINE тем, что некоторые почтовые ящики read-only **могут** позволять изменение перманентного состояния отдельным пользователям (при наличии общего запрета. Новости, промаркированные в файле .newsrc на сервере, являются примером такой ситуации (некоторые пользователи могут менять состояние почтовых ящиков с атрибутом read-only).

Пример

```
C: A142 SELECT INBOX
S: * 172 EXISTS
S: * 1 RECENT
S: * OK [UNSEEN 12] Message 12 is first unseen
S: * OK [UIDVALIDITY 3857529045] UIDs valid
S: * FLAGS (\Answered \Flagged \Deleted \Seen \Draft)
S: * OK [PERMANENTFLAGS (\Deleted \Seen \*)] Limited
S: A142 OK [READ-WRITE] SELECT completed
```

6.3.2. Команда EXAMINE

Аргументы: имя почтового ящика

Отклик: **обязательно** непомеченный отклик FLAGS, EXISTS, RECENT

дополнительно OK непомеченный отклик UNSEEN, PERMANENTFLAGS

Результат: OK - проверка завершена, состояние selected

NO - неудачная проверка, состояние authenticated (нет почтового ящика или доступа к нему)

BAD - команда не поддерживается или некорректны аргументы

Команда EXAMINE идентична команде SELECT и возвращает такие же результаты, однако открывает выбранный почтовый ящик только для чтения (read-only). Не допускается перманентного изменения состояния почтового ящика.

Текст отмеченного отклика OK на команду EXAMINE **должен** начинаться с кода [READ-ONLY].

Пример

```
C: A932 EXAMINE blurdybloop
S: * 17 EXISTS
S: * 2 RECENT
S: * OK [UNSEEN 8] Message 8 is first unseen
S: * OK [UIDVALIDITY 3857529045] UIDs valid
S: * FLAGS (\Answered \Flagged \Deleted \Seen \Draft)
S: * OK [PERMANENTFLAGS ()] No permanent flags permitted
S: A932 OK [READ-ONLY] EXAMINE completed
```

6.3.3. Команда CREATE

Аргументы: имя почтового ящика

Отклик: для этой команды не существует специфических откликов

Результат: OK - успешное завершение

NO - невозможно создать почтовый ящик с указанным именем

BAD - команда не поддерживается или некорректны аргументы

Команда CREATE создает почтовый ящик с заданным именем. Отклик OK возвращается только при успешном создании почтового ящика. При попытке создать почтовый ящик INBOX[1] или с другим уже используемым именем возникает ошибка. В результате любой ошибки будет возвращаться помеченный отклик NO.

Если имя почтового ящика имеет суффикс с символом-разделителем уровней иерархии (сервер возвращает его по команде LIST), это означает попытку клиента создать почтовый ящик в иерархии имен. Реализации серверов, не требующие такого объявления, **должны** игнорировать его.

При наличии в имени разделителя уровней иерархии, серверу **следует** создавать все вышележащие уровни иерархии, которые требуются для успешного завершения команды CREATE. Иными словами, при создании имени foo/bar/zap на сервере, использующем символ "/" в качестве разделителя, **следует** создать уровни foo/ и foo/bar/, если их еще нет.

Если создается почтовый ящик с именем ранее удаленного ящика, его уникальные идентификаторы **должны** быть больше уникальных идентификаторов, используемых в прежней инкарнации почтового ящика с таким именем, если новый ящик не имеет другого значения уникального идентификатора корректности (см. описание команды UID).

Пример

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

Отметим, что интерпретация приведенного примера зависит от того, возвращает ли сервер символ "/" по команде LIST. Если "/" служит разделителем, создается новый уровень с именем blurdybloop в иерархии owatagusiam. В противном случае создаются два почтовых ящика на одном уровне иерархии.

6.3.4. Команда DELETE

Аргументы: имя почтового ящика

Отклик: для этой команды не существует специфических откликов

Результат: OK - успешное удаление

NO - невозможно удалить почтовый ящик с указанным именем

BAD - команда не поддерживается или некорректны аргументы

Команда DELETE навсегда удаляет указанный почтовый ящик. При попытке удаления ящика INBOX[1] или отсутствующего почтового ящика возникает ошибка.

Недопустимо с помощью команды DELETE удалять внутренние уровни иерархии имен. Например, почтовый ящик foo является внутренним в иерархии foo.bar (предполагается, что "." служит разделителем уровней), удаление foo **недопустимо** следует удалять foo.bar. Ошибка произойдет и при попытке удаления имени, имеющего подчиненные уровни и атрибут \Noselect (см. описание отклика LIST). Допускается удалять имена с подчиненными уровнями, которые не имеют атрибута \Noselect. В таких случаях все сообщения из почтового ящика, а имя приобретает атрибут \ Noselect.

Максимальное значение уникального идентификатора для удаляемого ящика **должно** сохраняться, чтобы при создании нового ящика с этим именем не возникало повторного использования идентификаторов, если для нового ящика не задается другое значение уникального идентификатора корректности (см. описание команды UID).

Пример

```
C: A682 LIST "" *
S: * LIST () "/" blurdybloop
S: * LIST (\Noselect) "/" foo
S: * LIST () "/" foo/bar
S: A682 OK LIST completed
C: A683 DELETE blurdybloop
S: A683 OK DELETE completed
C: A684 DELETE foo
S: A684 NO Name "foo" has inferior hierarchical names
C: A685 DELETE foo/bar
S: A685 OK DELETE Completed
C: A686 LIST "" *
S: * LIST (\Noselect) "/" foo
S: A686 OK LIST completed
C: A687 DELETE foo
S: A687 OK DELETE Completed
C: A82 LIST "" *
S: * LIST () "." blurdybloop
```

[1] Независимо от регистра символов. *Прим. перев.*

```
S: * LIST () "." foo
S: * LIST () "." foo.bar
S: A82 OK LIST completed
C: A83 DELETE blurdybloop
S: A83 OK DELETE completed
C: A84 DELETE foo
S: A84 OK DELETE Completed
C: A85 LIST "" *
S: * LIST () "." foo.bar
S: A85 OK LIST completed
C: A86 LIST "" %
S: * LIST (\Noselect) "." foo
S: A86 OK LIST completed
```

6.3.5. Команда RENAME

Аргументы: имя существующего почтового ящика, имя нового почтового ящика

Отклик: для этой команды не существует специфических откликов

Результат: OK - успешная смена имени

NO - не удалось изменить имя (нет такого почтового ящика или новое имя уже используется)

BAD - команда не поддерживается или некорректны аргументы

Команда RENAME служит для изменения имени почтового ящика. Помеченный отклик OK возвращается только после успешного изменения имени. Попытка переименовать несуществующий почтовый ящик или задать в качестве нового уже используемое имя приводит к ошибке. В случае любой ошибки возвращается помеченный отклик NO.

При замене имени почтового ящика, имеющего нижележащие уровни, их имена также **должны** изменяться. Например, при смене имени foo на zap имя foo/bar ("/" является разделителем уровней) должно стать zap/bar.

Максимальное значение уникального идентификатора для ящика, имя которого изменяется, **должно** сохраняться, чтобы при создании нового ящика с этим именем не возникало повторного использования идентификаторов, если для нового ящика не задается другое значение уникального идентификатора корректности (см. описание команды UID).

Допускается переименование почтового ящика INBOX при выполнении определенных условий - все сообщения из INBOX перемещаются в почтовый ящик с новым именем и создается заново пустой почтовый ящик INBOX. Если реализация сервера поддерживает иерархию имен в INBOX, на эту иерархию не действует переименование INBOX.

Пример

```
C: A682 LIST "" *
S: * LIST () "/" blurdybloop
S: * LIST (\Noselect) "/" foo
S: * LIST () "/" foo/bar
S: A682 OK LIST completed
C: A683 RENAME blurdybloop sarasoop
S: A683 OK RENAME completed
C: A684 RENAME foo zowie
S: A684 OK RENAME Completed
C: A685 LIST "" *
S: * LIST () "/" sarasoop
S: * LIST (\Noselect) "/" zowie
S: * LIST () "/" zowie/bar
S: A685 OK LIST completed
C: Z432 LIST "" *
S: * LIST () "." INBOX
S: * LIST () "." INBOX.bar
S: Z432 OK LIST completed
C: Z433 RENAME INBOX old-mail
S: Z433 OK RENAME completed
C: Z434 LIST "" *
S: * LIST () "." INBOX
S: * LIST () "." INBOX.bar
S: * LIST () "." old-mail
S: Z434 OK LIST completed
```

6.3.6. Команда SUBSCRIBE

Аргументы: имя почтового ящика.

Отклик: для этой команды не существует специфических откликов

Результат: OK - успешное завершение

NO - отказ - невозможна подписка для указанного имени

BAD - команда не поддерживается или некорректны аргументы

Команда SUBSCRIBE добавляет указанное имя почтового ящика в число «активных» или «подписанных» ящиков, список которых возвращается по команде LSUB. Отмеченный отклик OK возвращается только при успешной подписке.

Сервер **может** проверять существование почтового ящика, указанного в команде SUBSCRIBE. Однако для сервера **недопустимо** по своей воле удалять имена почтовых ящиков из списка даже в тех случаях, когда ящика уже нет[1].

Пример

```
C: A002 SUBSCRIBE #news.comp.mail.mime
S: A002 OK SUBSCRIBE completed
```

6.3.7. Команда UNSUBSCRIBE

Аргументы: имя почтового ящика

Отклик: для этой команды не существует специфических откликов

Результат: OK - успешный отказ от подписки

NO - неудача

BAD - команда не поддерживается или некорректны аргументы

Команда UNSUBSCRIBE служит для удаления указанного почтового ящика из числа «активных» или «подписанных» ящиков, список которых возвращается по команде LSUB. Отмеченный отклик OK возвращается только при успешном удалении из списка.

Пример

```
C: A002 UNSUBSCRIBE #news.comp.mail.mime
S: A002 OK UNSUBSCRIBE completed
```

6.3.8. Команда LIST

Аргументы: база, имя почтового ящика (возможны шаблоны)

Отклик: непомеченный отклик LIST

Результат: OK - список получен

NO - отказ - невозможно вернуть список с заданной базой или именем

BAD - команда не поддерживается или некорректны аргументы

Команда LIST возвращает заданное подмножество полного набора имен, доступных клиенту. В ответ на команду возвращается некоторое число (возможно, 0) помеченных откликов LIST, содержащих атрибуты имени, разделитель уровней иерархии и имя (см. описание отклика LIST).

Данные по команде LIST **следует** возвращать быстро лишних задержек. Например, **не следует** выполнять расчет состояний \Marked или \Unmarked и выполнять другие действия - если каждое имя потребует 1 сек., то на обработку 1200 будет потрачено 20 минут!

Пустая база ("") reference name argument indicates that the имя почтового ящика is interpreted as by SELECT. Возвращаемые имена почтовых ящиков **должны** соответствовать указанному шаблону имени. Непустая ссылка задает имя почтового ящика или уровень иерархии и контекст, в котором интерпретируется имя почтового ящика (в зависимости от реализации).

Пустое ("") имя почтового ящика задает специальный случай для возврата разделителя уровней иерархии и корневого имени для заданной базы. Возвращаемое корневое значение **может** быть пустым, если база не имеет корня или задана пустая база. Разделитель уровней возвращается в любом случае. Это позволяет клиенту определить разделитель даже при отсутствии почтового ящика с указанным именем.

База и имя почтового ящика преобразуются (в зависимости от реализации) в каноническую форму с однозначным представлением иерархии слева направо. Возвращаемое имя почтового ящика будет иметь эту форму.

Любую часть базы в преобразованной форме **следует** указывать как префикс, сохраняя указанную в аргументе команды форму базы для этого префикса. Это правило позволяет клиентам определять относится ли возвращенное имя почтового ящика к контексту базы и обнаружить пересечение имени с базой. Без этого правила клиент будет получать искаженную информацию о контексте имен. Для примера в таблице показано несколько вариантов интерпретации базы и имени почтового ящика серверами на основе UNIX.

Первые три строки показывают реализацию в контексте базы. Отметим, что ~smith/Mail **не следует** преобразовать во что-либо иное, типа /u2/users/smith/Mail, поскольку клиент не сможет определить, что интерпретация происходила в контексте базы.

База	Имя почтового ящика	Интерпретация
~smith/Mail/	foo.*	~smith/Mail/foo.*
archive/	%	archive/%
#news.	comp.mail.*	#news.comp.mail.*
~smith/Mail/	/usr/doc/foo	/usr/doc/foo
archive/	~fred/Mail/*	~fred/Mail/*

Символ "*" является шаблоном и соответствует любым символам (в произвольном количестве) в этой позиции. Символ "%" подобен "*", но не соответствует разделителям уровней иерархии. Если символ "%" указан последним в шаблоне имени почтового ящика, возвращаются также соответствующие уровни иерархии. Если эти уровни не являются почтовыми ящиками, которые можно выбрать, возвращается атрибут \ Noselect (см. описание отклика LIST).

Серверам разрешается «прятать» доступные почтовые ящики при использовании шаблонов в некоторых ситуациях. Например, серверы на основе UNIX могут ограничивать действие шаблона "*" так, чтобы начальный символ "/" не давал соответствия.

[1] Это требование связано с тем, что некоторые серверы могут удалять почтовые ящики с хорошо известными именами (например, system-alerts) после истечения срока действия его содержимого и потом восстанавливать почтовый ящик при необходимости.

Специальное имя INBOX включается в выходной список команды LIST, если ящик INBOX поддерживается сервером для данного пользователя и строка INBOX (в верхнем регистре) соответствует базе и имени почтового ящика с заданными в команде шаблонами. Критерием включения ящика INBOX является возможность его выбора по команде SELECT.

Пример

```
C: A101 LIST "" ""
S: * LIST (\Noselect) "/" ""
S: A101 OK LIST Completed
C: A102 LIST #news.comp.mail.misc ""
S: * LIST (\Noselect) "." #news.
S: A102 OK LIST Completed
C: A103 LIST /usr/staff/jones ""
S: * LIST (\Noselect) "/" /
S: A103 OK LIST Completed
C: A202 LIST ~/Mail/ %
S: * LIST (\Noselect) "/" ~/Mail/foo
S: * LIST () "/" ~/Mail/meetings
S: A202 OK LIST completed
```

6.3.9. Команда LSUB

Аргументы: база, имя почтового ящика (возможны шаблоны)

Отклик: непомеченный отклик LSUB

Результат: OK - успешное завершение

NO - неудача

BAD - команда не поддерживается или некорректны аргументы

Команда LSUB возвращает подмножество имен из набора, объявленного пользователем «активным» или «подписанным». LSUB возвращает 0 или больше помеченных откликов. Аргументы LSUB имеют такую же форму, как аргументы команды LIST.

Сервер **может** проверить существование подписанных имен. Если имя не существует, его **следует** возвращать с флагом \Noselect в отклике LSUB. Для серверов **недопустимо** самовольное удаление имени почтового ящика из списка активных, даже если ящик больше не существует.

Пример

```
C: A002 LSUB "#news." "comp.mail.*"
S: * LSUB () "." #news.comp.mail.mime
S: * LSUB () "." #news.comp.mail.misc
S: A002 OK LSUB completed
```

6.3.10. Команда STATUS

Аргументы: имя почтового ящика, имена элементов состояния

Отклик: непомеченный отклик STATUS

Результат: OK - успешное завершение

NO - неудача - для заданного имени нет данных о состоянии

BAD - команда не поддерживается или некорректны аргументы

Команда STATUS запрашивает сведения о состоянии указанного почтового ящика. Команда не меняет выбранный почтовый ящик и не влияет на состояние каких-либо сообщений в ящике, для которого запрашивается состояние (в частности, при использовании команды STATUS **недопустим** сброс флага \Recent).

Команда STATUS является альтернативой открытию второго соединения IMAP4rev1 и использованию команды EXAMINE, позволяя получать данные о состоянии без организации специального соединения IMAP4rev1.

В отличие от LIST команда STATUS не гарантирует быстрого отклика. В некоторых реализациях сервер должен открыть почтовый ящик в режиме read-only (только чтение) для получения некоторых данных о состоянии. Кроме того, в отличие от LIST команда STATUS не допускает использования шаблонов.

Определенные на сегодняшний день элементы состояния включают:

MESSAGES число сообщений в почтовом ящике.

RECENT число сообщений с флагом \Recent.

UIDNEXT значение UID, которое будет использовано для следующего сообщения в почтовом ящике. Гарантируется неизменность этого значения пока в ящик не будет добавлено новое сообщение. Гарантируется также изменение значения при получении нового письма, даже если это сообщение будет потом исключено.

UIDVALIDITY значение уникального идентификатора корректности для почтового ящика.

UNSEEN число сообщений, для которых не установлен флаг \Seen.

Пример

```
C: A042 STATUS blurdybloop (UIDNEXT MESSAGES)
S: * STATUS blurdybloop (MESSAGES 231 UIDNEXT 44292)
```

```
S: A042 OK STATUS completed
```

6.3.11. Команда APPEND

Аргументы: имя почтового ящика, [заключенный в скобки список флагов][1], [строка даты/времени], сообщение

Отклик: для этой команды не существует специфических откликов

Результат: OK - успешное добавление

NO - ошибка - невозможно добавить сообщение в указанный ящик, ошибка в флагах, дате или тексте

BAD - команда не поддерживается или некорректны аргументы

Команда APPEND добавляет переданный ей аргумент как новое сообщение в конце указанного почтового ящика. Для аргументов **следует** использовать синтаксис [RFC-822]. В сообщении допускаются 8-битовые символы. Серверы, неспособные корректно сохранять 8-битовые символы, **должны** обеспечивать обратимое преобразование 8-битовых данных команды APPEND в 7-битовые с использованием транспортного кодирования содержимого [MIME-IMB]. Отметим, что эти правила могут иметь исключения - например, при использовании команды APPEND для сохранения черновиков сообщений можно опустить строки заголовка [RFC-822]. Однако использоваться такие исключения **должны** с осторожностью.

Если присутствует список флагов в скобках, в добавленном сообщении **следует** установить заданные флаги (по умолчанию флаги отсутствуют).

Если указана дата и время, для добавленного сообщения **следует** установить заданные значения даты и времени (по умолчанию устанавливаются текущая дата и время).

Если при добавлении произошла какая-то ошибка, почтовый ящик **должен** быть восстановлен в исходном виде (как до команды APPEND) - частичная добавка сообщения недопустима.

Если указанного почтового ящика не существует, сервер **должен** возвратить сообщение об ошибке; автоматическое создание почтового ящика **недопустимо**. Кроме тех случаев, когда указанный почтовый ящик не может быть создан, сервер **должен** добавлять код [TRYCREATE] перед текстом помеченного отклика NO, намекая клиенту на возможность использования команды CREATE и повтора APPEND при удачном создании почтового ящика.

Если почтовый ящик в данный момент выбран, **следует** выполнить действия как при обычном получении нового сообщения. В частности, серверу **следует** незамедлительно уведомить клиента с помощью непомеченного отклика EXISTS. Если сервер не делает этого, клиент **может** ввести команду NOOP (или CHECK) после одной или нескольких команд APPEND.

Пример

```
C: A003 APPEND saved-messages (\Seen) {310}
C: Date: Mon, 7 Feb 1994 21:52:25 -0800 (PST)
C: From: Fred Foobar <foobar@Blurdybloop.COM>
C: Subject: afternoon meeting
C: To: mooch@owatagu.siam.edu
C: Message-Id: <B27397-0100000@Blurdybloop.COM>
C: MIME-Version: 1.0
C: Content-Type: TEXT/PLAIN; CHARSET=US-ASCII
C:
C: Hello Joe, do you think we can meet at 3:30 tomorrow?
C:
S: A003 OK APPEND completed
```

Отметим, что команда APPEND не используется для доставки почты, поскольку она не обеспечивает механизма передачи информации из конвертов [SMTP].

6.4. Команды клиента - состояние Selected

В состоянии selected поддерживаются команды для работы с сообщениями в почтовом ящике.

В дополнение к универсальным командам (CAPABILITY, NOOP, LOGOUT), и командам после аутентификации (SELECT, EXAMINE, CREATE, DELETE, RENAME, SUBSCRIBE, UNSUBSCRIBE, LIST, LSUB, STATUS, APPEND) поддерживаются команды CHECK, CLOSE, EXPUNGE, SEARCH, FETCH, STORE, COPY, UID.

6.4.1. Команда CHECK

Аргументы: не используются

Отклик: для этой команды не существует специфических откликов

Результат: OK - проверка завершена успешно

BAD - команда не поддерживается или некорректны аргументы

Команда CHECK запрашивает выполнение контрольной операции (checkpoint) для выбранного почтового ящика. Контрольная операция может быть любым действием, связанным с почтовым ящиком) и определяется используемой реализацией (примером такой контрольной операции может служить сброс содержимого почтового ящика из памяти сервера на диск). Выполнение контрольной операции **может** занимать достаточно много времени. Если сервер не поддерживает контрольных операций, команда CHECK становится эквивалентом команды NOOP.

Нет никакой гарантии, что после команды CHECK будет передаваться непомеченный отклик EXISTS - для гарантированного получения такого отклика **следует** использовать команду NOOP.

[1] В квадратных скобках указаны необязательные параметры. *Прим. перев.*

Пример

```
C: FXXZ CHECK
S: FXXZ OK CHECK Completed
```

6.4.2. Команда CLOSE

Аргументы: не используются

Отклик: для этой команды не существует специфических откликов

Результат: OK - выбор ящика отменен, текущее состояние authenticated

NO - отказ - не выбрано почтового ящика

BAD - команда не поддерживается или некорректны аргументы

Команда CLOSE уничтожает в выбранном почтовом ящике все сообщения с установленным флагом \Deleted и обеспечивает возврат в состояние authenticated из состояния. В ответ на команду передается без тега отклик EXPUNGE.

Сообщения не удаляются и сообщения об ошибке не возвращается, если почтовый ящик был выбран командой EXAMINE или открыт в режиме read-only (только чтение).

Если почтовый ящик выбран, команды SELECT, EXAMINE, LOGOUT **могут** вводиться без предшествующей команды CLOSE. Команды SELECT, EXAMINE и LOGOUT явно отменяют текущий выбор почтового ящика без удаления сообщений из него. Однако при наличии большого числа сообщений с флагом \Deleted последовательности CLOSE-LOGOUT и CLOSE-SELECT заметно быстрее, чем EXPUNGE-LOGOUT или EXPUNGE-SELECT, поскольку они не передают непомеченных откликов EXPUNGE (которые клиент обычно игнорирует).

Пример

```
C: A341 CLOSE
S: A341 OK CLOSE completed
```

6.4.3. Команда EXPUNGE

Аргументы: не используются

Отклик: непомеченный отклик EXPUNGE

Результат: OK - удаление завершено

NO - отказ при удалении (например, нет прав доступа)

BAD - команда не поддерживается или некорректны аргументы

Команда EXPUNGE уничтожает из выбранного почтового ящика все сообщения с флагом \Deleted. Прежде, чем клиенту будет возвращен отклик OK, ему передается непомеченный отклик EXPUNGE для каждого удаленного письма.

Пример

```
C: A202 EXPUNGE
S: * 3 EXPUNGE
S: * 3 EXPUNGE
S: * 5 EXPUNGE
S: * 8 EXPUNGE
S: A202 OK EXPUNGE completed
```

В приведенном примере сообщения 3, 4, 7, 11 имеют флаг \Deleted (см. описание отклика EXPUNGE).

6.4.4. Команда SEARCH

Аргументы: [CHARSET][1], критерии поиска (один или несколько)

Отклик: **Обязательный** непомеченный отклик SEARCH

Результат: OK - удачное завершение поиска

NO - ошибка - невозможно : can't search that [CHARSET] or criteria

BAD - команда не поддерживается или некорректны аргументы

Команда SEARCH обеспечивает поиск в почтовом ящике сообщений, удовлетворяющих заданным критериям. Критерий поиска содержит один или несколько ключей. Непомеченный отклик SEARCH от сервера содержит список порядковых номеров сообщений, которые соответствуют критерию поиска.

При использовании множества ключей поиска они комбинируются с помощью логической операции AND (и). Например, критерий DELETED FROM "SMITH" SINCE 1-Feb-1994 задает поиск всех удаленных сообщений от Smith, помещенных в ящик после 1 февраля 1994 года. При задании ключей поиска могут использоваться логические операции OR (или) и NOT (отрицание), а также скобки.

Сервер **может** исключать части [MIME-IMB] в теле сообщения с типами, отличными от TEXT или MESSAGE при выполнении команды SEARCH.

Необязательный параметр [CHARSET] состоит из ключевого слова CHARSET, за которым следует зарегистрированное значение [CHARSET]. Этот параметр задает набор символов в строке критерия поиска. Транспортное кодирование [MIME-IMB] и строки [MIME-HDRS] в заголовках [RFC-822]/[MIME-IMB] **должны** декодироваться перед сравнением текста, если [CHARSET] отличается от US-ASCII. Кодировка US-ASCII **должна**

[1]Необязательный параметр. *Прим. перев.*

поддерживаться, другие варианты [CHARSET] также можно поддерживать. Если сервер не поддерживает заданное значение [CHARSET], он **должен** возвращать помеченный отклик NO (а не BAD).

Во всех ключах поиска, содержащих строки текста, условие считается выполненным, если ключ поиска является частью строки в сообщении. Регистр символов при поиске не учитывается.

Ниже перечислен список поддерживаемых ключей, а их синтаксические определения приведены в главе 9. Формальный синтаксис.

<message set>	сообщения с порядковым номером, соответствующим указанному набору номеров.
ALL	все сообщения в почтовом ящике; принято по умолчанию как стартовый ключ для AND.
ANSWERED	сообщения с установленным флагом \Answered.
BCC <string>	сообщения, содержащие указанную строку в структуре поля BCC в конверте.
BEFORE <date>	сообщения, в которых внутренняя дата раньше указанной.
BODY <string>	сообщения, содержащие указанную строку в теле письма.
CC <string>	сообщения, содержащие указанную строку в структуре поля CC в конверте.
DELETED	сообщения с установленным флагом \Deleted.
DRAFT	сообщения с установленным флагом \Draft.
FLAGGED	сообщения с установленным флагом \Flagged.
FROM <string>	сообщения, содержащие указанную строку в структуре поля FROM в конверте.
HEADER <field-name> <string>	сообщения, в заголовке которых содержится поле с указанным именем (определено в [RFC-822]) и содержимым.
KEYWORD <flag>	сообщения с установленным флагом, который указан.
LARGER <n>	сообщения с размером [RFC-822], превышающим заданное число октетов.
NEW	сообщения с установленным флагом \Recent, но без флага \Seen (эквивалент RECENT UNSEEN - получено недавно и не прочитано)".
NOT <search-key>	сообщения, не соответствующие заданному ключу поиска.
OLD	сообщения без флага \Recent (функциональный эквивалент NOT RECENT).
ON <date>	сообщения, для которых внутренняя дата совпадает с заданной.
OR	<search-key1> <search-key2> сообщения, соответствующие хотя бы одному ключу поиска.
RECENT	сообщения с флагом \Recent.
SEEN	сообщения с флагом \Seen flag.
SENTBEFORE <date>	сообщения, для которых дата [RFC-822] Date: раньше указанной.
SENTON <date>	сообщения, для которых дата [RFC-822] Date: совпадает с указанной.
SENTSINCE <date>	сообщения, для которых дата [RFC-822] Date: совпадает с указанной или позже ее.
SINCE <date>	сообщения, для которых внутренняя дата совпадает с указанной или позже ее.
SMALLER <n>	сообщения размером [RFC-822] меньше указанного числа октетов.
SUBJECT <string>	сообщения, содержащие указанную строку в поле SUBJECT конверта.
TEXT <string>	сообщения, содержащие указанную строку в заголовке или теле письма.
TO <string>	сообщения, содержащие указанную строку в поле TO конверта.
UID <message set>	сообщения с уникальными идентификаторами, соответствующими указанному набору.
UNANSWERED	сообщения без флага \Answered.
UNDELETED	сообщения без флага \Deleted.
UNDRAFT	сообщения без флага \Draft.
UNFLAGGED	сообщения без флага \Flagged.
UNKEYWORD <flag>	сообщения без указанного ключом флага.
UNSEEN	сообщения без флага \Seen.

Пример

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

6.4.5. Команда FETCH

Аргументы: набор сообщений, имена элементов данных в сообщениях

Отклик: непомеченный отклик FETCH

Результат: OK - успешная выборка

NO - неудача при попытке выборки

BAD - команда не поддерживается или некорректны аргументы

Команда FETCH отыскивает данные, связанные с сообщениями в почтовом ящике. Отыскиваемые данные могут задаваться с помощью атома или списка в скобках.

В настоящее время поддерживаются выборки для следующих типов данных:

ALL макроопределение для (FLAGS INTERNALDATE RFC822.SIZE ENVELOPE)

BODY нерасширяемая форма BODYSTRUCTURE.

BODY[<section>]<<partial>> текст в конкретной секции тела сообщения. Спецификация секций сообщения (если она присутствует) задается номером или одним из идентификаторов: HEADER, HEADER.FIELDS, HEADER.FIELDS.NOT, MIME, TEXT. Пустая спецификация секции задает поиск по всему сообщению, включая заголовки.

Каждое сообщение имеет по крайней мере один номер секции (части сообщения). Сообщения, не относящиеся к [MIME-IMB], и сообщения [MIME-IMB] без инкапсуляции имеют только часть 1.

В многосекционных сообщениях каждая часть имеет порядковый номер (начиная с 1). Если отдельная часть имеет тип сообщения или содержит в себе отдельные части, нумерация **должна** задаваться двумя числами (через запятую) - сначала номер основной части, затем - вложенной. Части типа MESSAGE/RFC822 также имеют номера вложенных частей, указывающие на части тела сообщения.

Идентификаторы HEADER, HEADER.FIELDS, HEADER.FIELDS.NOT, TEXT могут указывать секцию сообщения сами по себе или использоваться с числовым префиксом, указывающим номер части типа MESSAGE/RFC822. Идентификатор MIME **должен** использоваться с одним или несколькими числовыми префиксами.

Идентификаторы HEADER, HEADER.FIELDS, HEADER.FIELDS.NOT указывают на заголовок сообщения [RFC-822] или инкапсулированного сообщения [MIME-IMT] MESSAGE/RFC822. Идентификаторы HEADER.FIELDS и HEADER.FIELDS.NOT сопровождаются списком имен полей (определены в [RFC-822]) и возвращают часть заголовка. Идентификатор HEADER.FIELDS задает возврат включены в список, а при использовании HEADER.FIELDS.NOT возвращаются поля, которые отсутствуют в списке. При сравнении имен полей регистр символов не учитывается, а в остальных случаях сравнение ведется с учетом регистра. Во всех случаях пустая строка между заголовком и телом сообщения включается в результат поиска.

Идентификатор MIME указывает на заголовок [MIME-IMB] для этой части.

Идентификатор TEXT указывает на текст тела сообщений без заголовка [RFC-822].

Ниже приведен пример сложного сообщения с указателями на его отдельные части:

HEADER (заголовок [RFC-822] для сообщения)

```
TEXT        MULTIPART/MIXED
1           TEXT/PLAIN
2           APPLICATION/OCTET-STREAM
3           MESSAGE/RFC822
3.HEADER    (заголовок [RFC-822])
3.TEXT      (текст тела сообщения [RFC-822])
3.1         TEXT/PLAIN
3.2         APPLICATION/OCTET-STREAM
4           MULTIPART/MIXED
4.1         IMAGE/GIF
4.1.MIME    (заголовок [MIME-IMB] для IMAGE/GIF)
4.2         MESSAGE/RFC822
4.2.HEADER (заголовок [RFC-822])
4.2.TEXT    (текст тела сообщения [RFC-822])
4.2.1       TEXT/PLAIN
4.2.2       MULTIPART/ALTERNATIVE
4.2.2.1     TEXT/PLAIN
4.2.2.2     TEXT/RICHTEXT
```

Возможна выборка подстроки обозначенного текста, осуществляемая путем добавления конструкции **<позиция первого желаемого. максимальное число возвращаемых октетов>** к спецификатору выборки. Если стартовый октет указывает за пределы текста, возвращается пустая строка. Любая попытка частичной выборки за пределами текста усекается. Частичная выборка с позиции всегда дает результат, даже если происходит усекновение[1].

Отметим, что выборка подстроки для HEADER.FIELDS или HEADER.FIELDS.NOT рассчитывается после выделения из заголовка нужных полей.

Флаг \Seen устанавливается явно; если это ведет к изменению флагов, их **следует** включать как часть откликов FETCH.

BODY.PEEK[<section>]<<partial>> Другая форма BODY[<section>] , не устанавливающая явно флаг \Seen.

BODYSTRUCTURE Структура [MIME-IMB] в теле сообщения, рассчитываемая сервером путем разбора полей заголовка [MIME-IMB] в заголовке [RFC-822] и заголовков [MIME-IMB].

[1]Это означает, что BODY[]<0.2048> для сообщения размером в 1500 октетов будет возвращать BODY[]<0> с буквальным текстом размером 1500 октетов, а не просто BODY[].

ENVELOPE	Структура конверта в сообщении, рассчитываемая сервером путем разбора заголовка [RFC-822] на составные части с установкой при необходимости принятых по умолчанию значений полей.
FAST	Макроопределение для (FLAGS INTERNALDATE RFC822.SIZE)
FLAGS	Флаги, установленные для сообщения.
FULL	Макроопределение для (FLAGS INTERNALDATE RFC822.SIZE ENVELOPE BODY)
INTERNALDATE	Внутренняя дата сообщения.
RFC822	Функциональный эквивалент BODY[], отличающийся синтаксисом непомеченных данных FETCH (возвращается RFC822).
RFC822.HEADER	Функциональный эквивалент BODY.PEEK[HEADER] , отличающийся синтаксисом непомеченных данных FETCH (возвращается RFC822.HEADER).
RFC822.SIZE	The [RFC-822] size of the message.
RFC822.TEXT	Функциональный эквивалент BODY[TEXT], отличающийся синтаксисом непомеченных данных FETCH (возвращается RFC822.TEXT).
UID	Уникальный идентификатор сообщения.

Пример

```
C: A654 FETCH 2:4 (FLAGS BODY[HEADER.FIELDS (DATE FROM)])
S: * 2 FETCH ....
S: * 3 FETCH ....
S: * 4 FETCH ....
S: A654 OK FETCH completed
```

6.4.6. Команда STORE

Аргументы: набор сообщений, значение для элемента данных сообщения

Отклик: непомеченный отклик FETCH

Результат: OK - успешная запись

NO - неудача

BAD - команда не поддерживается или некорректны аргументы

Команда STORE изменяет данные для указанного сообщения в почтовом ящике. Обычно STORE возвращает измененное значение данных в непомеченном отклике FETCH. Суффикс ".SILENT"[1] в имени элемента данных предотвращает возврат непомеченных откликов FETCH и серверу **следует**, что клиент способен сам определить обновленное значение или оно не требуется клиенту.

В настоящее время могут изменяться следующие данные:

FLAGS <flag list>	замена флагов сообщения; новые значения флагов возвращаются в непомеченном отклике FETCH.
FLAGS.SILENT <flag list>	эквивалент FLAGS, но без возврата новых флагов.
+FLAGS <flag list>	добавление флага; новые значения флагов возвращаются в непомеченном отклике FETCH.
+FLAGS.SILENT <flag list>	эквивалент +FLAGS, но без возврата новых флагов.
-FLAGS <flag list>	удаление флага для сообщения; новые значения флагов возвращаются в непомеченном отклике FETCH.
-FLAGS.SILENT <flag list>	эквивалент -FLAGS, но без возврата новых флагов.

Пример

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

6.4.7. Команда COPY

Аргументы: набор сообщений, имя почтового ящика

Отклик: для этой команды не существует специфических откликов

Результат: OK - успешное копирование

NO - неудача

BAD - команда не поддерживается или некорректны аргументы

Команда COPY копирует указанные сообщения в конец заданного почтового ящика. Для копии **следует** сохранять внутреннюю дату и значения флагов.

[1]Независимо от наличия суффикса ".SILENT" серверу **следует** передавать непомеченный отклик FETCH при изменении флагов внешним источником.

Если указанного почтового ящика не существует, серверу **следует** возвращать сообщение об ошибке и **не следует** автоматически создавать почтовый ящик. Если нет очевидной невозможности создания почтового ящика с таким именем, сервер должен передавать код [TRYCREATE] в качестве префикса к тексту помеченного отклика NO. Это подсказывает клиенту возможность попытки создания почтового ящика с помощью команды CREATE и после этого использовать команду COPY, если почтовый ящик удалось создать.

При любой ошибке команды COPY сервер **должен** восстановить исходное состояние почтового ящика.

Пример

```
C: A003 COPY 2:4 MEETING
S: A003 OK COPY completed
```

6.4.8. Команда UID

Аргументы: имя команды, аргументы команды

Отклик: непомеченный отклик FETCH, SEARCH

Результат: OK - успешное завершение

NO - ошибка

BAD - команда не поддерживается или некорректны аргументы

Команда UID имеет две формы. В первом варианте используются команды COPY, FETCH, STORE и аргументы соответствующей команды. Однако в качестве номеров при указании набора сообщений используются уникальные идентификаторы сообщений, а не их порядковые номера. Во второй форме команда UID использует команду SEARCH и ее набор аргументов. Интерпретация аргументов такая же, как для команды SEARCH, однако вместо порядковых номеров сообщений возвращаются их уникальные идентификаторы (UID). Например, команда UID SEARCH 1:100 UID 443:557 будет возвращать уникальные идентификаторы, соответствующие пересечению сообщений с порядковыми номерами от 1 до 100 и UID в диапазоне443:557.

Допустимо включение диапазона сообщений, однако не гарантируется, что уникальные идентификаторы для диапазона образуют непрерывную последовательность. Несуществующие уникальные идентификаторы в заданном диапазоне будут игнорироваться без генерации сообщений об ошибках.

Число после "*" в непомеченном отклике FETCH всегда указывает порядковый номер сообщения, а не его уникальный идентификатор (даже в откликах на команду UID). Однако серверы **должны** явно включать значения UID как часть откликов FETCH на команду UID, независимо от указания UID в качестве элемента данных для FETCH.

Пример

```
C: A999 UID FETCH 4827313:4828442 FLAGS
S: * 23 FETCH (FLAGS (\Seen) UID 4827313)
S: * 24 FETCH (FLAGS (\Seen) UID 4827943)
S: * 25 FETCH (FLAGS (\Seen) UID 4828442)
S: A999 UID FETCH completed
```

6.5. Команды клиента - экспериментальные/расширения

6.5.1. Команда X<atom>

Аргументы: зависит от реализации

Отклик: зависит от реализации

Результат: OK - успешное завершение

NO - неудача

BAD - команда не поддерживается или некорректны аргументы

Все команды с префиксом X являются экспериментальными. Команды, которые не включены в данную спецификацию, ее расширения/пересмотры в стандартах и предложенных стандартах или одобренные IESG экспериментальные протоколы, **должны** использовать префикс X.

Все добавляемые экспериментальными командами непомеченные отклики должны начинаться с префикса X. Для серверов **недопустима** передача таких непомеченных откликов, если клиент не использует связанных экспериментальных команд.

Пример

```
C: a441 CAPABILITY
S: * CAPABILITY IMAP4rev1 AUTH=KERBEROS_V4 XPIG-LATIN
S: a441 OK CAPABILITY completed
C: A442 XPIG-LATIN
S: * XPIG-LATIN ow-nay eaking-spay ig-pay atin-lay
S: A442 OK XPIG-LATIN ompleted-cay
```

7. Отклики сервера

Отклики сервера делятся на три типа - информация о состоянии, данные и запросы на продолжение команд. Информация, содержащаяся в откликах, помечена словом "Содержимое:" при описании отклика (но не его синтаксиса). Синтаксис откликов описан в главе 9. Формальный синтаксис.

Клиент **должен** сохранять постоянную готовность к получению откликов от сервера.

Статусные отклики могут быть отмеченными (tagged) или неотмеченными (untagged). Отмеченные отклики показывают результат (OK, NO, BAD) выполнения команды клиента и включают тег, соответствующий команде.

Все данные и некоторые статусные отклики передаются без тегов. Непомеченные отклики идентифицируются маркером "*" взамен тега. Непомеченные статусные отклики содержат приветствия сервера, или данные о состоянии, которые не показывают завершения команд (например, уведомление об отключении сервера - shutdown alert). В силу исторических причин неотмеченные статусные отклики сервера называют также unsolicited data (представленные без запроса данные), хотя, строго говоря, они не являются таковыми.

Некоторые данные от сервера **должны** записываться клиентом - такие случаи явно указываются в описании данных. Данные этого типа содержат критически важную информацию, которая влияет на интерпретацию последующих команд и откликов (например, обновлений о создании или удалении сообщений).

Другие типы **данных** следует сохранять для использования в будущем; если клиенту не нужно записывать эти данные или смысл записи данных неясен (например, отклик SEARCH при отсутствии выполняемых команд SEARCH), такие данные **следует** игнорировать.

Непомеченные данные по инициативе сервера могут передаваться, когда соединение IMAP находится в состоянии selected. В этом состоянии сервер проверяет почтовый ящик на предмет появления в нем новых сообщений в результате выполнения команд. Обычно это является частью выполнения всех команд, следовательно, команда NOOP также инициирует проверку наличия новых сообщений. Если такие сообщения найдены, сервер передает непомеченные отклики EXISTS и RECENT, отражающие новый размер почтового ящика. Серверам, поддерживающим множественный доступ к почтовому ящику, **следует** также передавать подходящие непомеченные отклики FETCH и EXPUNGE, если другой агент изменяет состояние любого из флагов сообщений или удаляет какое-либо сообщение.

Запросы на продолжение команд используют маркер "+" взамен тега. Такие отклики передаются сервером для индикации восприятия сервером неполной команды клиента и готовности к получению оставшейся части команды.

7.1. Отклики сервера - состояние

К статусным относятся отклики OK, NO, BAD, PREAUTH и BYE. Отклики OK, NO, BAD могут передаваться с тегами или без таковых, отклики PREAUTH и BYE всегда являются неотмеченными.

Статусные отклики **могут** включать **дополнительно** «код отклика» (response code), состоящий из данных в квадратных скобках в форме атома, за которыми может следовать пробел и аргументы. Код отклика содержит дополнительную информацию или коды состояния для клиентской программы (сверх результата OK/NO/BAD) и определяются для тех случаев, когда клиент на основе этой информации может предпринять дополнительные действия.

В настоящем документе определены следующие коды откликов:

ALERT — текст для пользователя, содержащий специальное предупреждение, которое **должно** предостеречь пользователя понятным ему способом.

NEWNAME — с последующим именем нового почтового ящика. Команды SELECT и EXAMINE будут давать ошибку в связи с тем, что указанный почтовый ящик больше не существует в результате смены имени. Это является советом клиенту повторить операцию SELECT или EXAMINE с новым именем почтового ящика.

PARSE — понятный пользователю текст, представляющий ошибку при разборе заголовка [RFC-822] или заголовков [MIME-IMB] в сообщении.

PERMANENTFLAGS — вместе с последующим списком в скобках показывает какие из известных флагов клиент может поменять окончательно. Флаги, включенные в непомеченный отклик FLAGS, но отсутствующие в списке PERMANENTFLAGS нельзя изменить окончательно. Если клиент пытается сохранить с помощью команды STORE флаги, не указанные в списке PERMANENTFLAGS, сервер будет отвергать команду, возвращая отклик NO или сохранит флаги только до конца текущего сеанса. Список PERMANENTFLAGS может включать также специальный флаг *, который показывает возможность создания новых ключевых слов за счет попытки сохранения этих флагов в почтовом ящике.

READ-ONLY — почтовый ящик выбран в режиме «только для чтения» или доступ после выбора в режиме «чтения-записи» был изменен на read-only.

READ-WRITE — почтовый ящик выбран для чтения и записи или после выбора в режиме «только чтение» доступ был изменен на read-write.

TRYCREATE — неудачная попытка использования APPEND или COPY в результате отсутствия указанного почтового ящика (в отличие от других ошибок); этот отклик является советом клиенту сначала создать ящик с помощью команды CREATE, а потом повторить попытке.

UIDVALIDITY — вместе со следующим десятичным числом показывает уникальный идентификатор корректности.

UNSEEN — вместе со следующим десятичным числом показывает количество сообщений без флага \Seen.

Дополнительные коды откликов, используемые клиентами и серверами, **должны** начинаться с префикса "X", пока они не будут включены в новый вариант спецификации этого протокола. Клиентам **следует** игнорировать непонятные им коды откликов.

7.1.1. Отклик OK

Содержимое: **необязательный** текст для пользователя

Отклик OK показывает информационное сообщение от сервера. Отклик с тегом говорит об успешном завершении указанной тегом команды. В отклик **может** дополнительно включаться текст, понятный пользователю. Отклик без тега

указывает на информационное сообщение - природа информации **может** указываться кодом отклика. Отклики без тега используются также как одно из трех возможных приветствий при организации соединения, показывая, что аутентификации еще не прошла и требуется команда LOGIN.

Пример

```
S: * OK IMAP4rev1 server ready
C: A001 LOGIN fred blurdybloop
S: * OK [ALERT] System shutdown in 10 minutes
S: A001 OK LOGIN Completed
```

7.1.2. Отклик NO

Содержимое: **необязательный** текст для пользователя

Отклик NO говорит о возникновении ошибки на сервере. Отклик с тегом говорит о неудачной попытке выполнения указанной тегом команды. Неотмеченная форма используется как предупреждение о том, что выполнение команды еще не завершено. В отклик может добавляться понятный пользователю текст.

Пример

```
C: A222 COPY 1:2 owatagusiam
S: * NO Disk is 98% full, please delete unnecessary data
S: A222 OK COPY completed
C: A223 COPY 3:200 blurdybloop
S: * NO Disk is 98% full, please delete unnecessary data
S: * NO Disk is 99% full, please delete unnecessary data
S: A223 NO COPY failed: disk is full
```

7.1.3. Отклик BAD

Содержимое: **необязательный** текст для пользователя

Отклик BAD говорит об ошибке на сервере. Отклики с тегом сообщают об ошибке протокольного уровня при обработке сервером указанной тегом команды. Непомеченные отклики показывают ошибки протокольного уровня, которые не удалась связать с определенной командой. Отклик может содержать текст, понятный пользователю.

Пример

```
C: ...very long command line...
S: * BAD Command line too long
C: ...empty line...
S: * BAD Empty command line
C: A443 EXPUNGE
S: * BAD Disk crash, attempting salvage to a new disk!
S: * OK Salvage successful, no data lost
S: A443 OK Expunge completed
```

7.1.4. Отклик PREAUTH

Содержимое: **необязательный** текст для пользователя

Отклики PREAUTH всегда передаются без тега и служат в качестве одного из трех приветствий, передаваемых сервером при организации соединения. Отклик показывает, что аутентификация уже успешно проведена внешними силами и нет необходимости использовать команду LOGIN.

Пример

```
S: * PREAUTH IMAP4rev1 server logged in as Smith
```

7.1.5. Отклик BYE

Содержимое: **необязательный** текст для пользователя

Отклики BYE всегда передаются без тега и показывают, что сервер закрывает соединение. Для пользователя в отклик **может** добавляться текстовое сообщение. Отклик BYE может передаваться в четырех случаях:

1) Часть нормальной процедуры завершения сеанса. Сервер будет закрывать соединение после передачи помеченного отклика OK для команды LOGOUT.

2) Часть анонса неожиданного завершения работы сервера (panic shutdown). Соединение закрывается незамедлительно.

3) Анонс завершения сеанса по тайм-ауту. Соединение закрывается незамедлительно.

4) Одно из трех возможных приветствий при организации соединения, показывающее, что сервер не желает открывать сеанс для данного клиента. Соединение закрывается незамедлительно.

Различие между BYE при нормальной процедуре LOGOUT (случай 1) и в результате сбоя или отказа (остальные три ситуации) состоит в том, что в трех последних случаях сеанс закрывается немедленно.

Пример

```
S: * BYE Autologout; idle for too long
```

7.2. Отклики сервера - состояние сервера и почтового ящика

Эти отклики всегда передаются без тегов и служат для передачи клиенту данных о состоянии сервера и почтового ящика. Большинство этих откликов возвращается в результате вызова одноименных команд.

7.2.1. Отклик CAPABILITY

Содержимое: список возможностей

Отклик CAPABILITY возвращается в результате использования команды CAPABILITY. Список возможностей содержит поддерживаемые сервером возможности с использованием пробела в качестве разделителя. Список возможностей **должен** включать атом IMAP4rev1.

Имена возможностей, начинающиеся с AUTH=, показывают, что сервер поддерживает тот или иной механизм аутентификации.

Другие имена возможностей показывают, что сервер поддерживает расширения, пересмотренные варианты или замену протокола IMAP4rev1. Отклики сервера **должны** соответствовать данному документу, пока клиент не использует команд, связанных с соответствующими расширенными возможностями.

Имена возможностей должны быть стандартными расширениями, пересмотрами или заменами протокола IMAP4rev1, зарегистрированными в IANA, если они не начинаются с префикса "X".

Клиентам **не следует** требовать поддержки имен, отличающихся от IMAP4rev1 и они **должны** игнорировать неизвестные имена возможностей.

Пример

```
S: * CAPABILITY IMAP4rev1 AUTH=KERBEROS_V4 XPIG-LATIN
```

7.2.2. Отклик LIST

Содержимое: атрибуты имени, разделитель уровней иерархии, имя

Отклик LIST возвращается в результате вызова команды LIST и содержит одно имя, соответствующее спецификации в команде LIST. На одну команду LIST может возвращаться множество откликов LIST.

Определены 4 атрибута имен:

\Noinferiors для этого уровня иерархии невозможно существование любых дочерних уровней - их нет сейчас и не может быть в будущем.

\Noselect это имя нельзя использовать как выбранный почтовый ящик.

\Marked почтовый ящик отмечен сервером как «интересный» - возможно он содержит сообщения, добавленные с момента его предыдущего выбора.

\Unmarked почтовый ящик не содержит сообщений, добавленных после предыдущего обращение к нему.

Если сервер не может определить «интересен» ли данный почтовый ящик, или имя имеет атрибут \Noselect, серверу **не следует** передавать атрибуты \Marked или \Unmarked.

Разделитель уровней иерархии представляет собой символ, используемый в качестве границы между соседними уровнями иерархии имен почтовых ящиков. Клиент может использовать разделитель для создания дочерних почтовых ящиков или поиска других уровней в иерархии имен. На всех уровнях иерархии **должен** использоваться одинаковый разделитель. Значение NIL в качестве разделителя уровней означает отсутствие иерархии (плоская модель именования).

Имена представляются в иерархии слева направо и **должны** быть приемлемы для использования в командах LIST и LSUB. Если не указан атрибут \Noselect имя также **должно** быть корректно как аргумент для команд, принимающих в качестве аргумента имя почтового ящика (например, SELECT).

Пример

```
S: * LIST (\Noselect) "/" ~/Mail/foo
```

7.2.3. Отклик LSUB

Содержимое: атрибуты имени, разделитель уровней иерархии, имя

Отклик LSUB возвращается в результате использования команды LSUB и содержит одно имя, соответствующее спецификации в команде LSUB. На одну команду LSUB может возвращаться множество откликов LSUB. Формат данных идентичен формату в откликах LIST.

Пример

```
S: * LSUB () "." #news.comp.mail.misc
```

7.2.4 Отклик STATUS

Содержимое: имя, заключенный в скобки список состояний

Отклик STATUS возвращается в результате вызова клиентом команды STATUS и содержит имя почтового ящика, соответствующее спецификации в команде STATUS, а также информацию о состоянии этого почтового ящика.

Пример

```
S: * STATUS blurdybloop (MESSAGES 231 UIDNEXT 44292)
```

7.2.5. Отклик SEARCH

Содержимое: 0 или более чисел

Отклики SEARCH возвращаются в результате вызова команд SEARCH или UID SEARCH. Возвращаемые числа содержат порядковые номера (команда SEARCH) или уникальные идентификаторы (UID SEARCH) удовлетворяющих критериям поиска сообщений. В качестве разделителя используется символ пробела.

Пример

```
S: * SEARCH 2 3 6
```

7.2.6. Отклик FLAGS

Содержимое: заключенный в скобки список флагов

Отклики FLAGS возвращаются в результате использования клиентом команд SELECT или EXAMINE и содержат заключенный в скобки список флагов (как минимум, системные флаги), которые применимы к указанному почтовому ящику. Отличные от системных флаги используются в зависимости от реализации сервера.

Обновления по сравнению с откликом FLAGS **должны** записываться клиентом.

Пример

```
S: * FLAGS (\Answered \Flagged \Deleted \Seen \Draft)
```

7.3. Отклики сервера - размер почтового ящика

Отклики этой группы всегда передаются без тегов и показывают клиенту изменения размера почтового ящика. Вслед за маркером "*" сразу же указывается число сообщений в почтовом ящике.

7.3.1. Отклик EXISTS

Содержимое: отсутствует

Отклики EXISTS показывают число сообщений в почтовом ящике и передаются в результате использования команд SELECT и EXAMINE или при изменении почтового ящика (например, доставка нового письма).

Обновления числа сообщений **должны** записываться клиентом.

Пример

```
S: * 23 EXISTS
```

7.3.2. Отклик RECENT

Содержимое: отсутствует

Отклики RECENT говорят о числе сообщений с флагом \Recent и возвращаются в результате команд SELECT и EXAMINE или при изменении размера почтового ящика (например, в результате доставки нового письма).

Отметим, что непрерывность порядковых номеров свежих сообщений не гарантируется - они не обязательно занимают подряд n старших номеров в почтовом ящике (значение n возвращается откликом RECENT). Примером ситуации отсутствия такой непрерывности может быть одновременная работа множества клиентов работают с одним почтовым ящиком (первая уведомляемая сессия будет видеть свежие сообщения, остальные могут этого и не увидеть), а также случаи изменения порядка сообщений агентами, отличными от IMAP.

Надежными способами идентификации свежих сообщений являются только просмотр флагов и поиск или флага \ Recent или использование команды SEARCH RECENT.

Обновление откликов RECENT **должно** записываться клиентом.

Пример

```
S: * 5 RECENT
```

7.4. Отклики сервера - сообщение о состоянии

Статусные отклики всегда передаются без тегов. Отклики часто передаются сервером в результате использования клиентом одноименной с откликом команды. Сразу после маркера "*" в откликах содержится число, указывающее порядковый номер сообщения.

7.4.1. Отклик EXPUNGE

Содержимое: отсутствует

Отклик EXPUNGE информирует об уничтожении сообщения с указанным порядковым номером. Порядковый номер каждого следующего за уничтоженным сообщения уменьшается на 1 и это уменьшение отражается в нумерации для последующих откликов (включая и другие непомеченные отклики EXPUNGE).

Как результат правила немедленного уменьшения номеров, порядковые номера в последовательных откликах EXPUNGE зависят от порядка уничтожения сообщений (от старших номеров к младшим или наоборот). Например, при удалении последних 5 сообщений из ящика с 9 письмами, начиная с младшего номера, сервер будет передавать 5 последовательных непомеченных откликов EXPUNGE для сообщения с порядковым номером 5, тогда как при удалении со старшего номера отклики EXPUNGE будут возвращаться для сообщений 9, 8, 7, 6, 5.

Недопустимо передавать отклик EXPUNGE при отсутствии обрабатываемых команд или при обработке FETCH, STORE, SEARCH. Это правило позволяет избежать потери синхронизации порядковых номеров сообщений между клиентом и сервером.

Обновления откликов EXPUNGE **должны** записываться клиентом.

Пример

```
S: * 44 EXPUNGE
```

7.4.2. Отклик FETCH

Содержимое: данные сообщения

Отклик FETCH возвращает клиенту данные о сообщении. Данные представляются в виде пар имя - значение, заключенных в скобки. Этот отклик возвращается по команде в результате команд FETCH и STORE или по инициативе сервера (например, в результате обновления флагов).

Настоящая спецификация определяет следующие типы данных:

BODY — форма BODYSTRUCTURE без расширения данных.

BODY[<section>]<<origin_octet>> — Строковое выражение содержимого тела указанной части сообщения. Клиенту **следует** интерпретировать строку в соответствии с типом транспортного кодирования содержимого, типом и подтипом тела.

Если указан начальный октет, эта строка является подстрокой всего содержимого тела, начиная с начального октета. Это значит, что BODY[]<0> **можно** усечь, а BODY[] усекать **недопустимо**.

Допустимо использование 8-битовых текстовых данных, если идентификатор [CHARSET] является частью заключенного в скобки списка параметров тела для этой секции. Отметим, что заголовки (спецификаторы частей HEADER и MIME или части заголовка MESSAGE/RFC822) **должны** быть 7-битовыми (8-битовые символы недопустимы в заголовках). Отметим также, что в данные заголовка всегда включается пустая строка в конце.

Нетекстовые (бинарные) данные **должны** передаваться с использованием преобразования в текст (например, BASE64) до передачи клиенту. Для восстановления исходных двоичных данных клиент **должен** декодировать строку транспортного кодирования.

BODYSTRUCTURE — заключенный в скобки список, который описывает структуру тела сообщения [MIME-IMB]. Этот список создается сервером в результате разбора полей заголовка [MIME-IMB] и включения при необходимости принятых по умолчанию значений. Например, простое текстовое сообщение из 48 строк и 2279 октетов имеет структуру тела: ("TEXT" "PLAIN" ("CHARSET" "US-ASCII") NIL NIL "7BIT" 2279 48). При наличии множества частей они указываются вложенными списками в скобках. Вместо типа тела первого элемента заключенного в скобки списка помещается вложенное тело. Вторым элементов списка в скобках является подтип multipart (mixed, digest, parallel, alternative и т. п.). Например, сообщение из 2 частей, содержащее текст и данные в формате BASE64 может иметь структуру тела: (("TEXT" "PLAIN" ("CHARSET" "US-ASCII") NIL NIL "7BIT" 1152 23)("TEXT" "PLAIN" ("CHARSET" "US-ASCII" "NAME" "cc.diff") "<960723163407.20117h@cac.washington.edu>" "Compiler diff" "BASE64" 4554 73) "MIXED"))

После подтипа multipart следует расширение данных. Это расширение никогда не возвращается в выборках BODY но может содержаться в выборках BODYSTRUCTURE. При наличии расширенных данных они **должны** размещаться в приведенном ниже порядке.

Данные расширения в следующем порядке:

body parameter parenthesized list

Заключенный в скобки список пар «атрибут-значение» (например, ("foo" "bar" "baz" "rag"), где "bar" является значением атрибута "foo", а "rag" - значением "baz") в соответствии с [MIME-IMB].

body disposition

Заключенный в скобки список, который содержит строку типа размещения, за которой следует заключенный в скобки список пар «атрибут размещения - значение». Типы размещения и имена атрибутов будут определены в будущей версии предложенного стандарта [DISPOSITION].

body language

Строка или заключенный в скобки список, указывающие язык тела в соответствии с [LANGUAGE-TAGS].

Все последующие данные еще не определены в данной версии протокола. Такие расширения данных могут содержать 0 или более значений NIL, строк, чисел и потенциально вложенных заключенных в скобки списков таких данных. Реализации клиентов, использующие выборки BODYSTRUCTURE **должны** быть готовы к восприятию расширенных данных такого типа. Для серверов **недопустима** передача таких расширений, пока они не будут определены при пересмотре данного протокола.

Базовые поля для тел, не содержащих множества частей (non-multipart body) имеют следующий порядок:

body type

Строка, указывающая название типа среды в соответствии с [MIME-IMB].

body subtype

Строка, указывающая имя подтипа в соответствии с [MIME-IMB].

body parameter parenthesized list

Заключенный в скобки список пар «атрибут-значение» (например, ("foo" "bar" "baz" "rag"), где "bar" является значением атрибута "foo", а "rag" - значением "baz") в соответствии с [MIME-IMB].

body id

Строка, указывающая идентификатор содержимого в соответствии с [MIME-IMB].

body description

Строка, описывающая содержимое в соответствии с [MIME-IMB].

body encoding

Строка, указывающая транспортное кодирование в соответствии с [MIME-IMB].

body size

Число, показывающее размер тела в октетах. Отметим, что это число показывает размер для транспортного кодирования, а не размер после декодирования.

Тело типа MESSAGE или подтипа RFC822 содержит сразу же после основных полей структуру конверта, структуру тела и размер текстовых строк инкапсулированного сообщения.

Тело типа TEXT содержит сразу же после основных полей размер и собственно тело в форме текстовых строк. Отметим, что поле размера задается для транспортного кодирования, а не для содержимого после декодирования.

После основных полей и перечисленных выше полей, связанных с конкретными типами сообщений, следуют поля расширения. Эти поля никогда не возвращаются в выборке BODY, но могут возвращаться выборкой BODYSTRUCTURE. При использовании расширенных данных они должны размещаться в приведенном ниже порядке (для тел, не содержащих множества частей):

body MD5

Строка, дающая значение MD5 в соответствии с определением [MD5].

body disposition

Список в скобках таким же содержимым и функциями, как расположение тела для multipart body part.

body language

Строка или заключенный в скобки список, указывающие язык тела сообщения в соответствии с [LANGUAGE-TAGS].

Все последующие расширения данных еще не определены в этой версии протокола и описываются как данные multipart-расширений.

ENVELOPE — заключенный в скобки список, описывающий структуру конверта в сообщении. Этот список создается сервером путем разбора заголовка [RFC-822] на составные части и включения при необходимости принятых по умолчанию значений. Поля в структуре конверта располагаются в следующем порядке: date, subject, from, sender, reply-to, to, cc, bcc, in-reply-to, and message-id. Поля date, subject, in-reply-to и message-id представляют собой текстовые строки, поля from, sender, reply-to, to, cc, bcc - заключенные в скобки списки адресных структур.

Адресная структура представляет собой заключенный в скобки список, который описывает адрес электронной почты. Поля структуры располагаются в следующем порядке: personal name, [SMTP] at-domain-list (source route), mailbox name, host name.

Групповой синтаксис [RFC-822] указывается специальной формой адресной структуры, в которой имя хоста имеет значение NIL. Если имя почтового ящика также имеет значение NIL, это говорит о маркере окончания группы (точка с запятой в синтаксисе RFC 822). Если имя почтового ящика отличается от NIL, это говорит о маркере начала группы и поле имени почтового ящика содержит имя группы.

Любые неприменимые поля конверта или адресной структуры содержат значение NIL. Отметим, что сервер **должен** по умолчанию брать значения полей reply-to и sender из поля from; предполагается, что клиент не обязан их знать.

FLAGS — заключенный в скобки список флагов, установленных для сообщения.

INTERNALDATE — строка, представляющая внутреннюю дату сообщения.

RFC822 — эквивалент BODY[].

RFC822.HEADER — эквивалент BODY.PEEK[HEADER].

RFC822.SIZE — число, выражающее размер [RFC-822] для сообщения.

RFC822.TEXT — эквивалент BODY[TEXT].

UID — число, выражающее уникальный идентификатор сообщения.

Пример

```
S: * 23 FETCH (FLAGS (\Seen) RFC822.SIZE 44827)
```

7.5. Отклики сервера - запрос продолжения команды

Запросы на продолжение команды указываются маркером "+" взамен тега. Эта форма откликов показывает, что сервер готов воспринять от клиента продолжение команды. Остальная часть отклика представляет собой текстовую строку.

Этот отклик используется в команде AUTHORIZATION для передачи сервером данных клиенту и запроса дополнительных данных от клиента. Этот же отклик применяется с любой командой, аргументом которой является литерал.

Клиенту не разрешается передавать октеты литералов, пока сервер не укажет готовность к их приему. Это позволяет серверу построчно обрабатывать команды и отвергать ошибки. Оставшаяся часть команды, включая завершающую последовательность CRLF передается в форме октетов. При наличии дополнительных аргументов они должны отделяться пробелом.

Пример

```
C: A001 LOGIN {11}
S: + Ready for additional command text
C: FRED FOOBAR {7}
S: + Ready for additional command text
C: fat man
S: A001 OK LOGIN completed
C: A044 BLURDYBLOOP {102856}
S: A044 BAD No such command as "BLURDYBLOOP"
```

8. Пример соединения IMAP4rev1

Ниже приведен пример соединения IMAP4rev1 (длинные строки разделены на части только в целях форматирования).

```
S:   * OK IMAP4rev1 Service Ready
C:   a001 login mrc secret
S:   a001 OK LOGIN completed
C:   a002 select inbox
S:   * 18 EXISTS
S:   * FLAGS (\Answered \Flagged \Deleted \Seen \Draft)
S:   * 2 RECENT
S:   * OK [UNSEEN 17] Message 17 is the first unseen message
S:   * OK [UIDVALIDITY 3857529045] UIDs valid
S:   a002 OK [READ-WRITE] SELECT completed
C:   a003 fetch 12 full
S:   * 12 FETCH (FLAGS (\Seen) INTERNALDATE "17-Jul-1996 02:44:25 -0700"
      RFC822.SIZE 4286 ENVELOPE ("Wed, 17 Jul 1996 02:23:25 -0700 (PDT)"
      "IMAP4rev1 WG mtg summary and minutes"
      (("Terry Gray" NIL "gray" "cac.washington.edu"))
      (("Terry Gray" NIL "gray" "cac.washington.edu"))
      (("Terry Gray" NIL "gray" "cac.washington.edu"))
      ((NIL NIL "imap" "cac.washington.edu"))
      ((NIL NIL "minutes" "CNRI.Reston.VA.US")
      ("John Klensin" NIL "KLENSIN" "INFOODS.MIT.EDU")) NIL NIL
      "<B27397-0100000@cac.washington.edu>")
       BODY ("TEXT" "PLAIN" ("CHARSET" "US-ASCII") NIL NIL "7BIT" 3028 92))
S:    a003 OK FETCH completed
C:    a004 fetch 12 body[header]
S:    * 12 FETCH (BODY[HEADER] {350}
S:    Date: Wed, 17 Jul 1996 02:23:25 -0700 (PDT)
S:    From: Terry Gray <gray@cac.washington.edu>
S:    Subject: IMAP4rev1 WG mtg summary and minutes
S:    To: imap@cac.washington.edu
S:    cc: minutes@CNRI.Reston.VA.US, John Klensin <KLENSIN@INFOODS.MIT.EDU>
S:    Message-Id: <B27397-0100000@cac.washington.edu>
S:    MIME-Version: 1.0
S:    Content-Type: TEXT/PLAIN; CHARSET=US-ASCII
S:
S:    )
S:    a004 OK FETCH completed
C:    a005 store 12 +flags \deleted
S:    * 12 FETCH (FLAGS (\Seen \Deleted))
S:    a005 OK +FLAGS completed
C:    a006 logout
S:    * BYE IMAP4rev1 server terminating connection
S:    a006 OK LOGOUT completed
```

9. Формальный синтаксис

В приведенной ниже спецификации синтаксиса используется расширенная форма Бэкуса-Наура (BNF), заданная в [RFC-822], с единственным исключением - символ-ограничитель (delimiter), используемый с "#" порождает один пробел (SPACE), а не одну или несколько запятых.

При наличии альтернативных или дополнительных правил, когда последующее правило отменяет действие указанного ранее правила, приоритет **должно** иметь правило, указанное раньше. Например, "\Seen" при рассмотрении флагов дает имя флага \Seen, а не расширение flag_extension, несмотря на возможность разборки "\Seen" как flag_extension. Некоторые (но не все) случаи использования этого правила отмечены ниже при описании синтаксиса.

Если явно не указано иное, регистр символов не принимается во внимание. Строчные и прописные символы используются в описаниях синтаксиса для лучшего восприятия. Реализации должны трактовать эти строки независимо от регистра символов.

```
address          ::= "(" addr_name SPACE addr_adl SPACE addr_mailbox SPACE addr_host ")"
addr_adl         ::= nstring
                 ;; сохраняет [RFC-822] route-addr (если оно не равно NIL)
addr_host        ::= nstring
                 ;; NIL указывает на групповой синтаксис [RFC-822], в остальных случаях
                 ;; содержит доменное имя [RFC-822]
addr_mailbox     ::= nstring
                 ;; NIL указывает конец группы [RFC-822]; в остальных случаях содержит имя
                 ;; группы [RFC-822] (если addr_host = NIL) или локальную часть [RFC-822]
addr_name        ::= nstring
                 ;; фраза из почтового ящика [RFC-822] или 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"
                 ;; регистр символов различается
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>
atom_specials    ::= "(" / ")" / "{" / SPACE / CTL / list_wildcards / quoted_specials
authenticate     ::= "AUTHENTICATE" SPACE auth_type *(CRLF base64)
auth_type        ::= atom
                 ;; определяется в [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 ")"
                 ;; Будущее расширение. Реализации клиентов должны воспринимать поля
                 ;; body_extension. Для реализаций серверов недопустимо генерировать поля
                 ;; body_extension за исключением тех, которые будут определены в будущих
                 ;; стандартных (или standards-track) вариантах этой спецификации
body_ext_1part   ::= body_fld_md5 [SPACE body_fld_dsp [SPACE body_fld_lang [SPACE
                     1#body_extension]]]
                 ;; недопустимо возвращать нерасширяемые выборки "BODY"
body_ext_mpart   ::= body_fld_param [SPACE body_fld_dsp SPACE body_fld_lang [SPACE
                     1#body_extension]]
                 ;; недопустимо возвращать нерасширяемые выборки "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
                 ;; субтипом MESSAGE недопустимо задавать "RFC822"
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
                 ;; новые возможности должны начинаться с "X" или должны быть зарегистрированы
                 ;; в IANA как стандарт или предложенный стандарт
capability_data  ::= "CAPABILITY" SPACE [1#capability SPACE] "IMAP4rev1" [SPACE 1#capability]
                 ;; серверы IMAP4rev1, предлагающие совместимость с RFC 1730, должны
                 ;; указывать IMAP4 как первую возможность.
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
                ;; доступность команд зависит от состояния
command_any     ::= "CAPABILITY" / "LOGOUT" / "NOOP" / x_command
                ;; доступны во всех состояниях
command_auth    ::= append / create / delete / examine / list / lsub / rename / select /
                    status / subscribe / unsubscribe
                ;; доступны только в состояниях Authenticated и Selected
command_nonauth ::= login / authenticate
                ;; доступны только в состоянии Non-Authenticated
command_select  ::= "CHECK" / "CLOSE" / "EXPUNGE" / copy / fetch / store / uid / search
                ;; доступны только в состоянии Selected
continue_req    ::= "+" SPACE (resp_text / base64)
copy            ::= "COPY" SPACE set SPACE mailbox
CR              ::= <ASCII CR, carriage return, 0x0D>
create          ::= "CREATE" SPACE mailbox
                ;; имя 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
                ;; число месяца
date_day_fixed  ::= (SPACE digit) / 2digit
                ;; date_day в фиксированном формате
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
digit           ::= "0" / digit_nz
digit_nz        ::= "1" / "2" / "3" / "4" / "5" / "6" / "7" / "8" / "9"
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
                ;; Будущее расширение. Реализации клиентов должны воспринимать флаги
                ;; flag_extension. Для серверов недопустимо генерировать флаги flag_extension
                ;; кроме тех, которые будут определены в стандартных вариантах
                ;; 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 ")"
LF              ::= <ASCII LF, line feed, 0x0A>
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
login           ::= "LOGIN" SPACE userid SPACE password
lsub            ::= "LSUB" SPACE mailbox SPACE list_mailbox
mailbox         ::= "INBOX" / astring
                ;; В имени INBOX регистр символов не имеет значения и все варианты
                ;; (например, iNbOx) должны интерпретироваться как INBOX (см. параграф 5.1).
```

```
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
                ;; Определено в [MIME-IMT]
media_message   ::= <"> "MESSAGE" <"> SPACE <"> "RFC822" <">
                ;; Определено в [MIME-IMT]
media_subtype   ::= string
                ;; Определено в [MIME-IMT]
media_text      ::= <"> "TEXT" <"> SPACE media_subtype
                ;; Определено 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
                ;; Беззнаковое 32-битовое целое (0 <= n < 4,294,967,296)
nz_number       ::= digit_nz *digit
                ;; Беззнаковое 32-битовое целое (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
                ;; Использование INBOX как получателя приводит к ошибке NO
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
                ;; сервер немедленно закрывает соединение
response_tagged ::= tag SPACE resp_cond_state CRLF
resp_cond_auth  ::= ("OK" / "PREAUTH") SPACE resp_text
                ;; условие аутентификации
resp_cond_bye   ::= "BYE" SPACE resp_text
resp_cond_state ::= ("OK" / "NO" / "BAD") SPACE resp_text
                ;; состояние
resp_text       ::= ["[" resp_text_code "]" SPACE] (text_mime2 / text)
                ;; текст не должен начинаться с "[" 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
                ;; значение [CHARSET] должно быть зарегистрировано в 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" /
                ;; В [IMAP2] эта строка имела вид
                ;; "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 / "*"
                ;; * - наибольшее используемое значение. Для порядковых номеров сообщений
                ;; это число писем в ящике, для уникальных идентификаторов - уникальный
                ;; идентификатор последнего сообщения в ящике.
set             ::= sequence_num / (sequence_num ":" sequence_num) / (set "," set)
                ;; идентифицирует набор сообщений. Для порядковых номеров от целые числа от 1
```

```
                ;; до числа сообщения в ящике. Запятые разделяют числа, двоеточие указывает
                ;; интервал (включающий). Например, 2,4:7,9,12:* показывает сообщения
                ;; 2,4,5,6,7,9,12,13,14,15 для ящика с 15 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> "?="
                ;; синтаксис определен в [MIME-HDRS]
TEXT_CHAR       ::= <any CHAR except CR and LF>
time            ::= 2digit ":" 2digit ":" 2digit
                ;; часы, минуты, секунды
uid             ::= "UID" SPACE (copy / fetch / search / store)
                ;; уникальные идентификаторы, используемые взамен порядковых номеров
uniqueid        ::= nz_number
                ;; строго возрастает
unsubscribe     ::= "UNSUBSCRIBE" SPACE mailbox
userid          ::= astring
x_command       ::= "X" atom <experimental command arguments>
zone            ::= ("+" / "-") 4digit
                ;; 4-значное число со знаком, представляющее сдвиг от Гринвича в формате hhmm
                ;; (т. е., разницу между данным временем и Universal Time - UT). Вычитание
                ;; zone из данного времени дает значение UT (зона +0000).
```

10. Примечание автора

Этот документ является пересмотром предшествующих документов и переписывает спецификации протокола, включенные в эти документы - RFC 1730, неопубликованный документ IMAP2bis, RFC 1176, RFC 1064.

11. Вопросы безопасности

Транзакции протокола IMAP4rev1, включая данные электронной почты, передаются через сеть в открытом виде, если с помощью команды AUTHENTICATE не согласована защита передаваемой информации.

В сообщение сервера об отказе при выполнении команды AUTHENTICATE, которая произошла вследствие передачи некорректных полномочий (credentials) **не следует** включать детали некорректности представленных полномочий. При использовании команды LOGIN пароль передается в открытом виде. Этого можно избежать путем использования команды AUTHENTICATE взамен LOGIN. Сообщение об отказе при выполнении команды LOGIN **не следует** указывать пару “имя пользователя - пароль”, для которой получен отказ.

Дополнительные аспекты обеспечения безопасности рассмотрены в описании команд AUTHENTICATE и LOGIN.

12. Адрес автора

Mark R. Crispin

Networks and Distributed Computing

University of Washington

4545 15th Aveneue NE

Seattle, WA 98105-4527

Phone: (206) 543-5762

EMail: MRC@CAC.Washington.EDU

Перевод на русский язык

Николай Малых

nmalykh@gmail.com

Приложения

A. Литература

[ACAP] Myers, J. "ACAP -- Application Configuration Access Protocol", Work in Progress[1].

[CHARSET] Reynolds, J., and J. Postel, "Assigned Numbers", STD 2, RFC 1700[2], USC/Information Sciences Institute, October 1994.

[1]Работа завершена и опубликована в RFC 2244. *Прим. перев.*

[2]В соответствии с RFC 3232 документ STD 2 утратил силу. Значения Assigned Numbers следует искать в базе данных, доступной на сайте www.iana.org/numbers.html. *Прим. перев.*

[DISPOSITION] Troost, R., and Dorner, S., "Communicating Presentation Information in Internet Messages: The Content-Disposition Header", RFC 1806[3], June 1995.

[IMAP-AUTH] Myers, J., "IMAP4 Authentication Mechanism", RFC 1731. Carnegie-Mellon University, December 1994.

[IMAP-COMPAT] Crispin, M., "IMAP4 Compatibility with IMAP2bis", RFC 2061, University of Washington, November 1996.

[IMAP-DISC] Austein, R., "Synchronization Operations for Disconnected IMAP4 Clients", Work in Progress.

[IMAP-HISTORICAL] Crispin, M. "IMAP4 Compatibility with IMAP2 and IMAP2bis", RFC 1732, University of Washington, December 1994.

[IMAP-MODEL] Crispin, M., "Distributed Electronic Mail Models in IMAP4", RFC 1733, University of Washington, December 1994.

[IMAP-OBSOLETE] Crispin, M., "Internet Message Access Protocol - Obsolete Syntax", RFC 2062, University of Washington, November 1996.

[IMAP2] Crispin, M., "Interactive Mail Access Protocol - Version 2", RFC 1176, University of Washington, August 1990.

[LANGUAGE-TAGS] Alvestrand, H., "Tags for the Identification of Languages", RFC 1766[4], March 1995.

[MD5] Myers, J., and M. Rose, "The Content-MD5 Header Field", RFC 1864, October 1995.

[MIME-IMB] Freed, N., and N. Borenstein, "MIME (Multipurpose Internet Mail Extensions) Part One: Format of Internet Message Bodies", RFC 2045[5], November 1996.

[MIME-IMT] Freed, N., and N. Borenstein, "MIME (Multipurpose Internet Mail Extensions) Part Two: Media Types", RFC 2046[6], November 1996.

[MIME-HDRS] Moore, K., "MIME (Multipurpose Internet Mail Extensions) Part Three: Message Header Extensions for Non-ASCII Text", RFC 2047[5], November 1996.

[RFC-822] Crocker, D., "Standard for the Format of ARPA Internet Text Messages", STD 11, RFC 822[7], University of Delaware, August 1982.

[SMTP] Postel, J., "Simple Mail Transfer Protocol", STD 10, RFC 821[8], USC/Information Sciences Institute, August 1982.

[UTF-7] Goldsmith, D., and Davis, M., "UTF-7: A Mail-Safe Transformation Format of Unicode", RFC 1642[9], July 1994.

B. Изменения по отношению к RFC 1730

1) Добавлена команда STATUS.
2) Уточнен формальный синтаксис - "#" не может указывать на множественные символы пробела.
3) Устаревший синтаксис вынесен в отдельный документ.
4) Отменена команда PARTIAL.
5) Отменены атрибуты выборки RFC822.HEADER.LINES, RFC822.HEADER.LINES.NOT, RFC822.PEEK, RFC822.TEXT.PEEK.
6) Добавлены атрибуты "<" "." size ">" для текста BODY.
7) Добавлены идентификаторы частей HEADER, HEADER.FIELDS, HEADER.FIELDS.NOT, MIME, TEXT.
8) Добавлена поддержка Content-Disposition и Content-Language.
9) Снято ограничение для выборки во вложенных частях MULTIPART.
10) Отменено использование части тела с номером 0.
11) Поддерживаемые сервером способы аутентификации идентифицируются как возможности сервера.
12) Возможности протокола обозначаются как IMAP4rev1. Серверам, обеспечивающим совместимость с RFC 1730, следует возвращать также возможность IMAP4 (в дополнение к IMAP4rev1) в откликах CAPABILITY. Поскольку RFC-1730 требует указания IMAP4 как первой возможности, этот идентификатор **должен** указываться первым.
13) Добавлено описание соглашений для пространства имен почтовых ящиков.
14) Добавлено описание соглашения по использованию других языков в именах почтовых ящиков.
15) Элементы состояния UID-NEXT и UID-VALIDITY получили названия UIDNEXT и UIDVALIDITY (это изменение связано с незавершенной работой IMAP STATUS, а не с RFC-1730).
16) Добавление - пустое имя почтового ящика в аргументе команды LIST возвращает непомеченный отклик LIST с разделителем уровней иерархии и ссылкой на корень.

---

[3] В RFC 2183 содержится ряд дополнений и уточнений. *Прим. перев.*
[4] Этот документ устарел и заменен RFC 3066. *Прим. перев.*
[5] В RFC 2184, RFC 2231 содержится ряд дополнений и уточнений. *Прим. перев.*
[6] В RFC 2646 содержится ряд дополнений и уточнений. *Прим. перев.*
[7] В RFC 1123, RFC 1138, RFC 1148, RFC 1327, RFC 2156 содержится ряд дополнений и уточнений. *Прим. перев.*
[8] Этот документ устарел и заменен RFC 2821. *Прим. перев.*
[9] Этот документ устарел и заменен RFC 2152. *Прим. перев.*

17) Определены термины MUST, SHOULD, MUST NOT.

18) Добавлен параграф, определяющий атрибуты сообщений и более тонкие детали семантики порядковых номеров, UID и флагов.

19) Разъяснены обстоятельства, когда клиент может передавать множество команд, не дожидаясь отклика, и возможные неоднозначности в таких случаях.

20) Добавлены рекомендации в части поведения сервера по отношению к командам DELETE и RENAME при существовании подчиненных уровней иерархии.

21) Уточнение - имя почтового ящика не может быть произвольно исключено из подписки сервером даже если это имя больше не существует.

22) Уточнение - отклик на команду LIST должен даваться максимально быстро.

23) Уточнение - аргумент date_time функции APPEND устанавливает для сообщения внутреннюю дату.

24) Уточнено описание поведения APPEND для операций с выбранным в данный момент почтовым ящиком.

25) Уточнение - внешние изменения флагов должны всегда анонсироваться с помощью непомеченных откликов FETCH даже при использовании для аргументов команды STORE суффикса ".SILENT".

26) Уточнение - команда COPY копирует сообщения в указанный почтовый ящик.

27) Добавлен код отклика NEWNAME.

28) Переписано описание непомеченного отклика BYE с цель уточнения семантики.

29) Уточнены ссылки на RFC для тел MD5.

30) Уточнение - правила формального синтаксиса могут перекрываться и преимущество имеет первое правило из числа перекрывающихся.

31) Исправлено определение параметра body_fld_param.

32) Расширен формальный синтаксис для for capability_data.

33) Уточнение - любое написание "INBOX" должно трактоваться как INBOX (все буквы заглавные).

34) Уточнение - текст для пользователя в откликах resp_text не следует начинать с "[" или "=".

35) Изменены ссылки на черновые стандарты для MIME.

36) Уточнена семантика флагов \Recent.

37) Добавлены примеры.

C. Предметный указатель