Лаб 1 ВЕБ

Основные принципы HTTP,

Протокол HTTP является протоколом седьмого прикладного уровня модели OSI. HTTP есть основой системы World Wide Web, с помощью которого предоставляется возможность просмотра, доступа к веб-страницам в браузере и обеспечивается работа сети Интернет. В этом и состоит главная цель протокола. Также HTTP может служить в качестве транспорта для других протоколов прикладного уровня, например, SOAP.

Робота HTTP основана на клиент-серверной архитектуре. Где идет “общение” между клиентом и сервером посредством запросов и ответов. Об этом пройдемся более детально. Для этих целей использует протокол транспортного уровня TCP. Сервер работает на 80 порту, а для клиента номер порта генерируется автоматически операционной системой.

Клиенты (в большинстве случаев веб-браузеры, но могут быть и другим ПО) выполняют соединение, формируют и отправляют запрос к серверу. Эти запросы должны иметь стандартизованную структуру.

Сервер в свою очередь получает запрос от клиента, проверяет его, выполняет необходимые действия по полученному запросу, и возвращает ответ с сообщением с результатами выполнения назад клиенту. Аналогично имеет определенную установленную стандартную структуру.

Также **между клиентом и сервером могут существовать посредники**, такие как Proxy, для выполнения транспортных задач.

Структура запроса-ответа

Структура HTTP запроса должна включать в себя:

стартовую строку запроса с указанным методом, URL, и версией HTTP

Например,

GET /promotions HTTP/1.1

Host: hyperhost.ua

Синтаксис ответа сервера очень похож с структурой запроса и включает:

строку состояния, с указанием версии протокола, кодом состояния и его кратким описанием.

Например,

HTTP/1.1 200 OK

Именно по коду состояние клиентское приложение узнаёт о результатах отправленного запроса, анализирует его и определяет какие действия следует совершить дальше. Уже по началу кода состояния можно судить об успешности запроса, так как первая цифра указывает на класс состояния.

Методы

Методом задается само действие, которое нужно выполнить на сервере с ресурсом указанным в URL. URL- путь к запрашиваемому ресурсу. Коротко пройдемся по основным методах: GET, POST, HEAD, PUT, DELETE, TRACE, OPTIONS, CONNECT.

GET - этот метод применяется, чтобы получить данные с сервера по заданном адресе.

HEAD - работает по аналогии, как GET, и используется для получения запрашиваемого ресурса по указанному пути. А отличие в том, что в ответе сервер передает лишь заголовки и статусную строку, но без тела сообщения.

POST - нужен уже наоборот для передачи необходимый данных от клиента на сервер. Например, при заполнении форм регистрации посетителем сайта, отправка введенных данных будет идти этим методом.

PUT - с помощью метода осуществляется обновление ресурса по указанному пути.

DELETE - служит для удаления ресурса указанного в URI.

CONNECT - направляет существующее соединение через прокси.

OPTIONS - чтобы получить характеристики, возможности текущего HTTP соединения для указанного ресурса.

TRACE - позволяет проследить состояние ресурса, какие изменение, кем и когда вносились и т.д.

Есть несколько версий протокола, до недавнего времени чаще всего используется расширенная версия HTTP 1.1, так как без проблем поддерживается всеми браузерами и серверами, но при использовании было ряд нюансов касательно TCP соединений так называемого «медленного старта» (данные отправляются не сразу, а используется алгоритм для определения допустимого объема данных для передачи, чтобы не получить перегрузку сети, этот момент не дает использовать полностью пропускную способность сети), и спустя 10 лет был анонсирована новую версию HTTP/2, где главной особенностью было включение HTTP-потоков (позволяло выполнять несколько запросов через одно TCP-соединение). HTTP/2 частично решал вопрос, но не до конца. Потому с прошлого года внедрилась для тестирования версия HTTP/3, где уже вместо TCP используется QUIC (транспортный протокол разработан Google), для решения проблем с «медленным стартом».

Заголовки запросов и их значения

Содержит дополнительную информацию касательно системных параметров, файлах cookie и другой служебной информации. Нужно обязательно включить заголовок Host с версии HTTP 1.1, где указывается доменное имя. Делается то для того, что на одной IP может располагаться несколько сайтов, а заголовок будет указывать на сайт, для нужно выполнить то или иное действия опираясь на указанный метод в стартовой строке.

Может иметь и другие заголовки, например, Content-Type указывает на тип передаваемого сообщения, а Content-Length на его длину. Поля что начинаются с Accept указывают серверу выдавать только указанные в заголовку форматы данных, которые потом сможет распознать и обработать при получении клиент. Например, Accept - задается список допустимых форматов, Accept-Charset - список поддерживаемых кодировок, Accept-Language - список естественных языков, Accept-Ranges - список единиц измерения. Также можно выделить заголовок User-Agent, где указано программное обеспечение клиента и его компоненты. Существует очень много заголовков.

Заголовки от тела отделяются пустой строкой.

3) тело запроса. Является необязательным элементом и может отсутствовать. Содержит в себе данные, которые будут передаваться в сформированном запросе.

Заголовок ответа

Также дополняет ответ сервера дополнительной информацией, которая необходима для осуществления “общения” между клиентом и сервером. Можно выделить такие заголовки: Vary - предоставлен список всех заголовков запроса, которые были приняты во внимание при обработке запроса и формирования ответа. Server - указаны веб-сервер и его компоненты, именно где был сформирован ответ. А заголовок Via - преподносит список версий протокола, названий и версий прокси-серверов, через которых прошло сообщение. Заголовок Age содержит время в секундах, в течение которого объект находился в прокси-кэше. Allow - указывает на доступные методы, которые могут применяться к ресурсу (такой заголовок отправляется если указанный в запросе метод был недоступен). Public - похож на Allow, но определяет допустимые методы уже на уровне всего сервера. Заголовок Date - дата когда сообщение с сервера было отправлено к клиенту. Last-Modified - определяет дату последний модификации ресурса. Retry-After - указывает на время после которого клиент может осуществить следующий запрос к серверу.

Аналогично с другими заголовками можно ознакомится в документации.

тело ответа также не есть обязательным составляющим ответа, но как вариант, может предоставлять в себе содержание запрашиваемого ресурса.

Говорить об HTTP можно очень долго, в этой статье мы описали некоторые базовые моменты. Но будем и дальше продолжать обращать внимание и оповещать тему развития протокола.

Коды ответов

Список кодов состояния HTTP — Википедия (wikipedia.org)

Информационные 1

В этот класс выделены коды, информирующие о процессе передачи. При работе через протокол версии 1.0 сообщения с такими кодами должны игнорироваться. В версии 1.1 клиент должен быть готов принять этот класс сообщений как обычный ответ, но серверу отправлять что-либо не нужно.

Успех 2

Сообщения данного класса информируют о случаях успешного принятия и обработки запроса клиента. В зависимости от статуса сервер может ещё передать заголовки и тело сообщения.

Перенаправление 3

Коды этого класса сообщают клиенту, что для успешного выполнения операции необходимо сделать другой запрос, как правило, по другому URI. Из данного класса пять кодов 301, 302, 303, 305 и 307 относятся непосредственно к перенаправлениям. Адрес, по которому клиенту следует произвести запрос, сервер указывает в заголовке Location. При этом допускается использование фрагментов в целевом URI.

Ошибка клиента 4

Класс кодов 4xx предназначен для указания ошибок со стороны клиента. При использовании всех методов, кроме HEAD, сервер должен вернуть в теле сообщения гипертекстовое пояснение для пользователя.

Ошибка сервера 5

Пример ошибки 502 Bad Gateway

Коды 5xx выделены под случаи необработанных исключений при выполнении операций на стороне сервера. Для всех ситуаций, кроме использования метода HEAD, сервер должен включать в тело сообщения объяснение, которое клиент отобразит пользователю.

Коды ответов подробнее

Информационные

В этот класс выделены коды, информирующие о процессе передачи. При работе через протокол версии 1.0 сообщения с такими кодами должны игнорироваться. В версии 1.1 клиент должен быть готов принять этот класс сообщений как обычный ответ, но серверу отправлять что-либо не нужно.

100 Continue — сервер удовлетворён начальными сведениями о запросе, клиент может продолжать пересылать заголовки. Появился в HTTP/1.1.

101 Switching Protocols — сервер выполняет требование клиента и переключает протоколы в соответствии с указанием, данным в поле заголовка Upgrade. Сервер отправляет заголовок ответа Upgrade, указывая протокол, на который он переключился. Появился в HTTP/1.1.

102 Processing — запрос принят, но на его обработку понадобится длительное время. Используется сервером, чтобы клиент не разорвал соединение из-за превышения времени ожидания. Клиент при получении такого ответа должен сбросить таймер и дожидаться следующей команды в обычном режиме. Появился в WebDAV.

103 Early Hints — используется для раннего возврата части заголовков, когда заголовки полного ответа не могут быть быстро сформированы.

Успех

200 OK — успешный запрос. Если клиентом были запрошены какие-либо данные, то они находятся в заголовке и/или теле сообщения.

201 Created — в результате успешного выполнения запроса был создан новый ресурс. Сервер может указать адреса (их может быть несколько) созданного ресурса в теле ответа, при этом предпочтительный адрес указывается в заголовке Location. Серверу рекомендуется указывать в теле ответа характеристики созданного ресурса и его адреса, формат тела ответа определяется заголовком Content-Type. При обработке запроса новый ресурс должен быть создан до отправки ответа клиенту, иначе следует использовать ответ с кодом 202. Появился в HTTP/1.0.

202 Accepted — запрос был принят на обработку, но она не завершена. Клиенту не обязательно дожидаться окончательной передачи сообщения, так как может быть начат очень долгий процесс. Появился в HTTP/1.0.

203 Non-Authoritative Information — аналогично ответу 200, но в этом случае передаваемая информация была взята не из первичного источника (резервной

копии, другого сервера и т. д.) и поэтому может быть неактуальной. Появился в HTTP/1.1.

204 No Content — сервер успешно обработал запрос, но в ответе были переданы только заголовки без тела сообщения. Клиент не должен обновлять содержимое документа, но может применить к нему полученные метаданные. Появился в HTTP/1.0.

205 Reset Content — сервер обязывает клиента сбросить введённые пользователем данные. Тела сообщения сервер при этом не передаёт и документ обновлять не обязательно. Появился в HTTP/1.1.

206 Partial Content — сервер удачно выполнил частичный GET-запрос, возвратив только часть сообщения. В заголовке Content-Range сервер указывает байтовые диапазоны содержимого. Особое внимание при работе с подобными ответами следует уделить кэшированию. Появился в HTTP/1.1. (подробнее…)

207 Multi-Status — сервер передаёт результаты выполнения сразу нескольких независимых операций. Они помещаются в само тело сообщения в виде XML-документа с объектом multistatus. Не рекомендуется размещать в этом объекте статусы из серии 1xx из-за бессмысленности и избыточности. Появился в WebDAV.

208 Already Reported — члены привязки DAV уже были перечислены в предыдущей части (multistatus) ответа и не включаются снова.

226 IM Used — заголовок A-IM от клиента был успешно принят и сервер возвращает содержимое с учётом указанных параметров. Введено в RFC 3229 для дополнения протокола HTTP поддержкой дельта-кодирования.

Перенаправление

Коды этого класса сообщают клиенту, что для успешного выполнения операции необходимо сделать другой запрос, как правило, по другому URI. Из данного класса пять кодов 301, 302, 303, 305 и 307 относятся непосредственно к перенаправлениям. Адрес, по которому клиенту следует произвести запрос, сервер указывает в заголовке Location. При этом допускается использование фрагментов в целевом URI.

По последним стандартам клиент может производить перенаправление без запроса пользователя только если второй ресурс будет запрашиваться методом GET или HEAD[7]. В предыдущих спецификациях говорилось, что для избежания круговых переходов пользователя следует спрашивать после 5-го подряд перенаправления[16]. При всех перенаправлениях, если метод

запроса был не HEAD, то в тело ответа следует включить короткое гипертекстовое сообщение с целевым адресом, чтобы в случае ошибки пользователь смог сам произвести переход.

Разработчики HTTP отмечают, что многие клиенты при перенаправлениях с кодами 301 и 302 ошибочно применяют метод GET ко второму ресурсу, несмотря на то, что к первому запрос был с иным методом (чаще всего PUT)[17]. Чтобы избежать недоразумений, в версии HTTP/1.1 были введены коды 303 и 307 и их рекомендовано использовать вместо 302. Изменять метод нужно только если сервер ответил 303. В остальных случаях следующий запрос производить с исходным методом.

Поведение клиентов при различных перенаправлениях описано в таблице:

Статус ответа Кэширование Если метод не GET или HEAD

301 Moved Permanently Можно как обычно. Спросить у пользователя подтверждения и запросить другой ресурс исходным методом.

307 Temporary Redirect Можно только если указан заголовок Cache-Control или Expires.

302 Found (HTTP/1.1)

302 Moved Temporarily (HTTP/1.0)

303 See Other Нельзя. Перейти автоматически, но уже методом GET.

300 Multiple Choices — по указанному URI существует несколько вариантов предоставления ресурса по типу MIME, по языку или по другим характеристикам. Сервер передаёт с сообщением список альтернатив, давая возможность сделать выбор клиенту автоматически или пользователю. Появился в HTTP/1.0.

301 Moved Permanently — запрошенный документ был окончательно перенесен на новый URI, указанный в поле Location заголовка. Некоторые клиенты некорректно ведут себя при обработке данного кода. Появился в HTTP/1.0.

302 Found, 302 Moved Temporarily — запрошенный документ временно доступен по другому URI, указанному в заголовке в поле Location. Этот код может быть использован, например, при управляемом сервером согласовании

содержимого. Некоторые[какие?] клиенты некорректно ведут себя при обработке данного кода. Введено в HTTP/1.0.

303 See Other — документ по запрошенному URI нужно запросить по адресу в поле Location заголовка с использованием метода GET несмотря даже на то, что первый запрашивался иным методом. Этот код был введён вместе с кодом 307 для избежания неоднозначности, чтобы сервер был уверен, что следующий ресурс будет запрошен методом GET. Например, на веб-странице есть поле ввода текста для быстрого перехода и поиска. После ввода данных браузер делает запрос методом POST, включая в тело сообщения введённый текст. Если обнаружен документ с введённым названием, то сервер отвечает кодом 303, указав в заголовке Location его постоянный адрес. Тогда браузер гарантировано его запросит методом GET для получения содержимого. В противном случае сервер просто вернёт клиенту страницу с результатами поиска. Введено в HTTP/1.1.

304 Not Modified — сервер возвращает такой код, если клиент запросил документ методом GET, использовал заголовок If-Modified-Since или If-None-Match и документ не изменился с указанного момента. При этом сообщение сервера не должно содержать тела. Появился в HTTP/1.0.

305 Use Proxy — запрос к запрашиваемому ресурсу должен осуществляться через прокси-сервер, URI которого указан в поле Location заголовка. Данный код ответа могут использовать только исходные HTTP-сервера (не прокси). Введено в HTTP/1.1.

306 (зарезервировано) — использовавшийся раньше код ответа, в настоящий момент зарезервирован. Упомянут в RFC 2616 (обновление HTTP/1.1).

307 Temporary Redirect — запрашиваемый ресурс на короткое время доступен по другому URI, указанный в поле Location заголовка. Метод запроса (GET/POST) менять не разрешается. Например, POST запрос должен быть отправлен по новому URI тем же методом POST. Этот код был введён вместе с 303 вместо 302-го для избежания неоднозначности. Введено в RFC 2616 (обновление HTTP/1.1).

308 Permanent Redirect — запрашиваемый ресурс был окончательно перенесен на новый URI, указанный в поле Location заголовка. Метод запроса (GET/POST) менять не разрешается. Например, POST запрос должен быть отправлен по новому URI тем же методом POST. Этот код был введён вместо 301-го для избежания неоднозначности. Введено в RFC 7238 (RFC 7538).

Ошибка клиента

Класс кодов 4xx предназначен для указания ошибок со стороны клиента. При использовании всех методов, кроме HEAD, сервер должен вернуть в теле сообщения гипертекстовое пояснение для пользователя.

400 Bad Request — сервер обнаружил в запросе клиента синтаксическую ошибку. Появился в HTTP/1.0.

401 Unauthorized — для доступа к запрашиваемому ресурсу требуется аутентификация. В заголовке ответ должен содержать поле WWW-Authenticate с перечнем условий аутентификации. Иными словами, для доступа к запрашиваемому ресурсу клиент должен представиться, послав запрос, включив при этом в заголовок сообщения поле Authorization с требуемыми для аутентификации данными. Если запрос уже включает данные для авторизации, ответ 401 означает, что в авторизации с ними отказано.

402 Payment Required — предполагается использовать в будущем[когда?]. В настоящий момент[когда?] не используется. Этот код предусмотрен для платных пользовательских сервисов, а не для хостинговых компаний. Имеется в виду, что эта ошибка не будет выдана хостинговым провайдером в случае просроченной оплаты его услуг. Зарезервирован, начиная с HTTP/1.1.

Сервер вернул ошибку 403 при попытке просмотра каталога «cgi-bin», доступ к которому был запрещён

403 Forbidden[18] — сервер понял запрос, но он отказывается его выполнять из-за ограничений в доступе для клиента к указанному ресурсу. Иными словами, клиент не уполномочен совершать операции с запрошенным ресурсом. Если для доступа к ресурсу требуется аутентификация средствами HTTP, то сервер вернёт ответ 401, или 407 при использовании прокси. В противном случае ограничения были заданы администратором сервера или разработчиком веб-приложения и могут быть любыми в зависимости от возможностей используемого программного обеспечения. В любом случае серверу следует сообщить причины отказа в обработке запроса. Наиболее вероятными причинами ограничения может послужить попытка доступа к системным ресурсам веб-сервера (например, файлам .htaccess или .htpasswd) или к файлам, доступ к которым был закрыт с помощью конфигурационных файлов, требование аутентификации не средствами HTTP, например, для доступа к системе управления содержимым или разделу для

зарегистрированных пользователей либо сервер не удовлетворён IP-адресом клиента, например, при блокировках. Появился в HTTP/1.0.

404 Not Found[19] — самая распространённая ошибка при пользовании Интернетом, основная причина — ошибка в написании адреса Web-страницы. Сервер понял запрос, но не нашёл соответствующего ресурса по указанному URL. Если серверу известно, что по этому адресу был документ, то ему желательно использовать код 410. Ответ 404 может использоваться вместо 403, если требуется тщательно скрыть от посторонних глаз определённые ресурсы. Появился в HTTP/1.0.

405 Method Not Allowed — указанный клиентом метод нельзя применить к текущему ресурсу. В ответе сервер должен указать доступные методы в заголовке Allow, разделив их запятой. Эту ошибку сервер должен возвращать, если метод ему известен, но он не применим именно к указанному в запросе ресурсу, если же указанный метод не применим на всём сервере, то клиенту нужно вернуть код 501 (Not Implemented). Появился в HTTP/1.1.

406 Not Acceptable — запрошенный URI не может удовлетворить переданным в заголовке характеристикам. Если метод был не HEAD, то сервер должен вернуть список допустимых характеристик для данного ресурса. Появился в HTTP/1.1.

407 Proxy Authentication Required — ответ аналогичен коду 401 за исключением того, что аутентификация производится для прокси-сервера. Механизм аналогичен идентификации на исходном сервере. Появился в HTTP/1.1.

408 Request Timeout — время ожидания сервером передачи от клиента истекло. Клиент может повторить аналогичный предыдущему запрос в любое время. Например, такая ситуация может возникнуть при загрузке на сервер объёмного файла методом POST или PUT. В какой-то момент передачи источник данных перестал отвечать, например, из-за повреждения компакт-диска или потери связи с другим компьютером в локальной сети. Пока клиент ничего не передаёт, ожидая от него ответа, соединение с сервером держится. Через некоторое время сервер может закрыть соединение со своей стороны, чтобы дать возможность другим клиентам сделать запрос. Этот ответ не возвращается, когда клиент принудительно остановил передачу по команде пользователя или соединение прервалось по каким-то иным причинам, так как ответ уже послать невозможно. Появился в HTTP/1.1.

409 Conflict — запрос не может быть выполнен из-за конфликтного обращения к ресурсу. Такое возможно, например, когда два клиента пытаются изменить ресурс с помощью метода PUT. Появился в HTTP/1.1.

410 Gone — такой ответ сервер посылает, если ресурс раньше был по указанному URL, но был удалён и теперь недоступен. Серверу в этом случае неизвестно и местоположение альтернативного документа (например копии). Появился в HTTP/1.1.

411 Length Required — для указанного ресурса клиент должен указать Content-Length в заголовке запроса. Без указания этого поля не стоит делать повторную попытку запроса к серверу по данному URI. Такой ответ естественен для запросов типа POST и PUT. Например, если по указанному URI производится загрузка файлов, а на сервере стоит ограничение на их объём. Тогда разумней будет проверить в самом начале заголовок Content-Length и сразу отказать в загрузке, чем провоцировать бессмысленную нагрузку, разрывая соединение, когда клиент действительно пришлёт слишком объёмное сообщение. Появился в HTTP/1.1.

412 Precondition Failed — возвращается, если ни одно из условных полей заголовка (If-Match и др., см. RFC 7232) запроса не было выполнено. Появился в HTTP/1.1.

413 Payload Too Large — возвращается в случае, если сервер отказывается обработать запрос по причине слишком большого размера тела запроса. Сервер может закрыть соединение, чтобы прекратить дальнейшую передачу запроса. Если проблема временная, то рекомендуется в ответ сервера включить заголовок Retry-After с указанием времени, по истечении которого можно повторить аналогичный запрос. Появился в HTTP/1.1. Ранее назывался «Request Entity Too Large».

414 URI Too Long — сервер не может обработать запрос из-за слишком длинного указанного URI. Такую ошибку можно спровоцировать, например, когда клиент пытается передать длинные параметры через метод GET, а не POST. Появился в HTTP/1.1. Ранее назывался «Request-URI Too Long».

415 Unsupported Media Type — по каким-то причинам сервер отказывается работать с указанным типом данных при данном методе. Появился в HTTP/1.1.

416 Range Not Satisfiable — в поле Range заголовка запроса был указан диапазон за пределами ресурса и отсутствует поле If-Range. Если клиент передал байтовый диапазон, то сервер может вернуть реальный размер в поле Content-Range заголовка. Данный ответ не следует использовать при передаче типа multipart/byteranges[источник не указан 3177 дней]. Введено в RFC 2616 (обновление HTTP/1.1). Ранее назывался «Requested Range Not Satisfiable».

417 Expectation Failed — по каким-то причинам сервер не может удовлетворить значению поля Expect заголовка запроса. Введено в RFC 2616 (обновление HTTP/1.1).

418 I’m a teapot — Этот код был введен в 1998 году как одна из традиционных первоапрельских шуток IETF в RFC 2324, Hyper Text Coffee Pot Control Protocol. Не ожидается, что данный код будет поддерживаться реальными серверами[20].

419 Authentication Timeout (not in RFC 2616) — Этого кода нет в RFC 2616, используется в качестве альтернативы коду 401, которые прошли проверку подлинности, но лишены доступа к определенным ресурсам сервера. Обычно код отдается, если CSRF-токен устарел или оказался некорректным.

421 Misdirected Request — запрос был перенаправлен на сервер, не способный дать ответ.

422 Unprocessable Entity — сервер успешно принял запрос, может работать с указанным видом данных (например, в теле запроса находится XML-документ, имеющий верный синтаксис), однако имеется какая-то логическая ошибка, из-за которой невозможно произвести операцию над ресурсом. Введено в WebDAV.

423 Locked — целевой ресурс из запроса заблокирован от применения к нему указанного метода. Введено в WebDAV.

424 Failed Dependency — реализация текущего запроса может зависеть от успешности выполнения другой операции. Если она не выполнена и из-за этого нельзя выполнить текущий запрос, то сервер вернёт этот код. Введено в WebDAV.

425 Too Early — сервер не готов принять риски обработки "ранней информации". Введено в RFC 8470 для защиты от атак повторения при использовании 0-RTT в TLS 1.3.

426 Upgrade Required — сервер указывает клиенту на необходимость обновить протокол. Заголовок ответа должен содержать правильно сформированные поля Upgrade и Connection. Введено в RFC 2817 для возможности перехода к TLS посредством HTTP.

428 Precondition Required — сервер указывает клиенту на необходимость использования в запросе заголовков условий, наподобие If-Match. Введено в черновике стандарта RFC 6585.

429 Too Many Requests — клиент попытался отправить слишком много запросов за короткое время, что может указывать, например, на попытку DDoS-атаки. Может сопровождаться заголовком Retry-After, указывающим,

через какое время можно повторить запрос. Введено в черновике стандарта RFC 6585.

431 Request Header Fields Too Large — Превышена допустимая длина заголовков. Сервер не обязан отвечать этим кодом, вместо этого он может просто сбросить соединение. Введено в черновике стандарта RFC 6585.

434 Requested host unavailable — Запрашиваемый адрес недоступен[источник не указан 2614 дней].

449 Retry With — возвращается сервером, если для обработки запроса от клиента поступило недостаточно информации. При этом в заголовок ответа помещается поле Ms-Echo-Request. Введено корпорацией Microsoft для WebDAV. В настоящий момент[когда?] используется как минимум программой Microsoft Money.

451 Unavailable For Legal Reasons — доступ к ресурсу закрыт по юридическим причинам, например, по требованию органов государственной власти или по требованию правообладателя в случае нарушения авторских прав. Введено в черновике IETF за авторством Google[12], при этом код ошибки является отсылкой к роману Рэя Брэдбери «451 градус по Фаренгейту». Был добавлен в стандарт 21 декабря 2015 года[21].

499 Client Closed Request — нестандартный код, предложенный и используемый nginx для случаев, когда клиент закрыл соединение, пока nginx обрабатывал запрос.

Ошибка сервера

Пример ошибки 502 Bad Gateway

Коды 5xx выделены под случаи необработанных исключений при выполнении операций на стороне сервера. Для всех ситуаций, кроме использования метода HEAD, сервер должен включать в тело сообщения объяснение, которое клиент отобразит пользователю.

500 Internal Server Error[22] — любая внутренняя ошибка сервера, которая не входит в рамки остальных ошибок класса. Появился в HTTP/1.0.

501 Not Implemented — сервер не поддерживает возможностей, необходимых для обработки запроса. Типичный ответ для случаев, когда сервер не понимает указанный в запросе метод. Если же метод серверу известен, но он не применим к данному ресурсу, то нужно вернуть ответ 405. Появился в HTTP/1.0.

502 Bad Gateway — сервер, выступая в роли шлюза или прокси-сервера, получил недействительное ответное сообщение от вышестоящего сервера. Появился в HTTP/1.0.

503 Service Unavailable — сервер временно не имеет возможности обрабатывать запросы по техническим причинам (обслуживание, перегрузка и прочее). В поле Retry-After заголовка сервер может указать время, через которое клиенту рекомендуется повторить запрос. Хотя во время перегрузки очевидным кажется сразу разрывать соединение, эффективней может оказаться установка большого значения поля Retry-After для уменьшения частоты избыточных запросов. Появился в HTTP/1.0.

504 Gateway Timeout — сервер в роли шлюза или прокси-сервера не дождался ответа от вышестоящего сервера для завершения текущего запроса. Появился в HTTP/1.1.

505 HTTP Version Not Supported — сервер не поддерживает или отказывается поддерживать указанную в запросе версию протокола HTTP. Появился в HTTP/1.1.

506 Variant Also Negotiates — в результате ошибочной конфигурации выбранный вариант указывает сам на себя, из-за чего процесс связывания прерывается. Экспериментальное. Введено в RFC 2295 для дополнения протокола HTTP технологией Transparent Content Negotiation.

507 Insufficient Storage — не хватает места для выполнения текущего запроса. Проблема может быть временной. Введено в WebDAV.

508 Loop Detected — операция отменена, т.к. сервер обнаружил бесконечный цикл при обработке запроса без ограничения глубины. Введено в WebDAV.

509 Bandwidth Limit Exceeded — используется при превышении веб-площадкой отведённого ей ограничения на потребление трафика. В данном случае владельцу площадки следует обратиться к своему хостинг-провайдеру. В настоящий момент данный код не описан ни в одном RFC и используется только модулем «bw/limited», входящим в панель управления хостингом cPanel, где и был введён.

510 Not Extended — на сервере отсутствует расширение, которое желает использовать клиент. Сервер может дополнительно передать информацию о доступных ему расширениях. Введено в RFC 2774 для дополнения протокола HTTP поддержкой расширений.

511 Network Authentication Required — этот ответ посылается не сервером, которому был предназначен запрос, а сервером-посредником — например, сервером провайдера — в случае, если клиент должен сначала

авторизоваться в сети, например, ввести пароль для платной точки доступа к Интернету. Предполагается, что в теле ответа будет возвращена Web-форма авторизации или перенаправление на неё. Введено в черновике стандарта RFC 6585.

520 Unknown Error, возникает когда сервер CDN не смог обработать ошибку веб-сервера; нестандартный код CloudFlare.

521 Web Server Is Down, возникает когда подключения CDN отклоняются веб-сервером; нестандартный код CloudFlare.

522 Connection Timed Out, возникает когда CDN не удалось подключиться к веб-серверу; нестандартный код CloudFlare.

523 Origin Is Unreachable, возникает когда веб-сервер недостижим; нестандартный код CloudFlare.

524 A Timeout Occurred, возникает при истечении тайм-аута подключения между сервером CDN и веб-сервером; нестандартный код CloudFlare.

525 SSL Handshake Failed, возникает при ошибке рукопожатия SSL между сервером CDN и веб-сервером; нестандартный код CloudFlare.

526 Invalid SSL Certificate, возникает когда не удаётся подтвердить сертификат шифрования веб-сервера; нестандартный код CloudFlare.