Internet Engineering Task Force (IETF)
Request for Comments: 9111
STD: 98
Obsoletes: 7234
Category: Standards Track
ISSN: 2070-1721

R. Fielding, Ed.
Adobe
M. Nottingham, Ed.
Fastly
J. Reschke, Ed.
greenbytes

June 2022

HTTP Caching

Кэширование HTTP

Аннотация

Протокол передачи гипертекста (Hypertext Transfer Protocol или HTTP) является протоколом прикладного уровня без поддержки состояний для распределенных гипертекстовых информационных систем коллективной работы. Этот документ определяет кэширование HTTP и связанные с ним поля заголовков, управляющие поведением кэша или указывающие кэшируемые сообщения с откликами.

Этот документ отменяет действие RFC 7234.

Статус документа

Документ относится к категории Internet Standards Track.

Документ является результатом работы IETF[1] и представляет согласованный взгляд сообщества IETF. Документ прошёл открытое обсуждение и был одобрен для публикации IESG[2]. Дополнительную информацию о стандартах Internet можно найти в разделе 2 в RFC 7841.

Информацию о текущем статусе документа, ошибках и способах обратной связи можно найти по ссылке https://www.rfc-editor.org/info/rfc9111.

Оглавление

[1] Internet Engineering Task Force - комиссия по решению инженерных задач Internet.
[2] Internet Engineering Steering Group - комиссия по инженерным разработкам Internet.
[3] В оригинале - IETF Contributions. *Прим. перев.*
[4] В оригинале - IETF Standards Process. *Прим. перев.*

1. Введение

HTTP является протоколом прикладного уровня «запрос-отклик» без поддержки состояний, который использует расширяемую семантику и самоописательные сообщений для гибкого взаимодействия с сетевыми гипертекстовыми информационными системами. Обычно протокол применяется в распределенных информационных системах, где кэширование откликов может повышать производительность. Этот документ задаёт аспекты HTTP, связанные с кэшированием и повторным использованием сообщений с откликами.

Кэш HTTP - это локальное хранилище сообщений с откликами и подсистема управления хранением, извлечением и удалением записей кэша. Кэш сохраняет кэшируемые отклики для снижения времени отклика и расхода пропускной способности при повторении эквивалентных запросов. Любой клиент или сервер может использовать кэш, но не при работе в качестве туннеля (параграф 3.7 в [HTTP]).

Общим (shared) называется кэш, который сохраняет отклики для повторного использования не только одним пользователем. Общий кэш обычно (но не всегда) развёртывается как часть устройства-посредника. Частным (private) называется кэш, выделенный для одного пользователя, что чато реализуется как компонент пользовательского агента.

Целью кэширования HTTP является существенное повышение производительности за счёт неоднократного использования отклика на предшествующее сообщение для выполнения текущего запроса. Кэш считает сохранённый отклик свежим (см. параграф 4.2), если его можно использовать без проверки (пригодности кэшированного отклика для данного запроса). Свежий отклик в кэше может сокращать задержку и загрузку сети при каждом его использовании. Если кэшированный отклик не является свежим, он все равно может использоваться, если проверка может обновить его (параграф 4.3) или источник не доступен (параграф 4.2.4).

Этот документ отменяет RFC 7234, сводка отличий приведена в Приложении B.

1.1. Уровни требований

Ключевые слова **необходимо** (MUST), **недопустимо** (MUST NOT), **требуется** (REQUIRED), **нужно** (SHALL), **не следует** (SHALL NOT), **следует** (SHOULD), **не нужно** (SHOULD NOT), **рекомендуется** (RECOMMENDED), **не рекомендуется** (NOT RECOMMENDED), **возможно** (MAY), **необязательно** (OPTIONAL) в данном документе интерпретируются в соответствии с BCP 14 [RFC2119] [RFC8174] тогда и только тогда, когда они выделены шрифтом, как показано здесь.

В разделе 2 [HTTP] указаны критерии соответствия и рассмотрены вопросы, связанные с обработкой ошибок.

1.2. Обозначения синтаксиса

В этой спецификации применяется нотация дополненных форм Бэкуса-Наура (Augmented Backus-Naur Form или ABNF) [RFC5234], расширенная для поддержки строк с учётом регистра символов [RFC7405]. Используется также набор расширений, заданных в параграфе 5.6.1 [HTTP], который позволяет компактно определять списки с разделением запятыми, используя оператор # (подобно указанию повтора оператором *). В Приложении A представлена сводная грамматика операций со списками в стандартной нотации ABNF.

1.2.1. Импортированные правила

Базовое правило DIGIT (decimal 0-9) указано ссылкой на Приложении B.1 к [RFC5234].

В [HTTP] заданы указанные ниже правила

```
HTTP-date     = <HTTP-date, [HTTP], параграф 5.6.7>
OWS           = <OWS, [HTTP], параграф 5.6.3>
field-name    = <field-name, [HTTP], параграф 5.1>
quoted-string = <quoted-string, [HTTP], параграф 5.6.4>
token         = <token, [HTTP], параграф 5.6.2>
```

1.2.2. Delta-seconds

Правило delta-seconds задаёт неотрицательное целое число, представляющее время в секундах.

```
delta-seconds  = 1*DIGIT
```

Получатель, анализирующий значение delta-seconds и преобразующий его в двоичную форму, должен использовать арифметический тип размером не менее 31 бита из диапазона неотрицательных целых чисел. Если кэш получает значение delta-seconds больше максимального значения, которое он может представить, или при последующих вычислениях возникает переполнение, кэш **должен** принять значение 2147483648 (231) или наибольшее целое число, которое он может удобно представить.

> Примечание. Значение 2147483648 приведено здесь по историческим причинам, оно представляет бесконечность (более 68 лет) и его не обязательно хранить в двоичной форме. Реализация может выдавать его в виде строки в случае переполнения, даже если вычисления выполняются с арифметическим типом, не способным напрямую представить это число. Важно обнаружить переполнение и не считать результат отрицательным значением при последующих расчётах.

2. Обзор операций кэширования

Корректная работа кэша позволяет сохранить семантику передач HTTP, сокращая объем передаваемых по сети данных за счёт использования кэша. Базовая терминология и концепции HTTP описаны в разделе 3 [HTTP].

Хотя кэширование является **необязательным** свойством HTTP, можно предположить, что повторное использование кэшированных откликов желательно и применяется по умолчанию, если какие-либо требования или локальная конфигурация не запрещают этого. Поэтому требования к кэшу HTTP нацелены на предотвращение хранения в кэше неиспользуемых откликов и повторного использования неправомерно сохранённых откликов, а не на то, чтобы кэш всегда сохранял для повторного использования определённые отклики.

Ключ кэша (cache key) - это информация, которую кэш использует для выбора отклика и которая включает, по меньшей мере, метод запроса и URI, использованные для извлечение сохранённого отклика. Метод определяет обстоятельства, при которых этот отклик можно использовать для выполнения последующих запросов. Однако многие из современных кэшей HTTP кэшируют лишь отклики GET и поэтому в качестве ключа применяется лишь URI.

В кэше может храниться несколько откликов для цели запроса, являющейся предметом согласования содержимого. Кэш различает эти отклики путём включения в ключ некоторых полей заголовка исходного запрос, используя сведения из поля заголовка отклика Vary, как описано в параграфе 4.1. Кэш может включать в ключ дополнительный материал. Например, кэш пользовательского агента может включать отождествление ссылающегося сайта, тем самым «удваивая ключ» что позволяет избежать некоторых рисков приватности (см. параграф 7.2).

Чаще всего в кэшах хранятся успешные результаты запросов на извлечение, например, отклики 200 (OK) на запрос GET, содержащие представление целевого ресурса (параграф 9.3.1 в [HTTP]). Однако могут сохраняться перенаправления, отрицательные (например, 404 (Not Found)) и неполные (например, 206 (Partial Content)) результаты, а также отклики для методов, отличных от GET, если определение метода разрешает такое кэширование и задаёт что-либо пригодное в качестве ключа кэша.

Кэш является «отсоединённым», если он не может связаться с сервером-источником или иным способом найти путь пересылки для запроса. Отсоединённый кэш может в некоторых обстоятельствах обслуживать устаревшие отклики (параграф 4.2.4).

3. Сохранение откликов в кэше

Отклики **недопустимо** сохранять в кеше, если не выполняются указанные ниже условия:

- метод запроса понятен для кэша;
- код статуса в отклике является окончательным (см. раздел 15 в [HTTP]);

- отклик имеет код статуса 206 или 304 или присутствует директива must-understand (см. параграф 5.2.2.3); кэш понимает код статуса в отклике;
- если в отклике нет директивы no-store (см. параграф 5.2.2.5);
- если кэш является общим и директива private в отклике отсутствует или позволяет общему кэшу сохранять изменённый отклик (см. параграф 5.2.2.7);
- если кэш является общим и поля заголовка Authorization нет в запросе (см. параграф 11.6.2 в [HTTP]) или в отклике имеется директива, явно разрешающая общее кэширование(см. параграф 3.5);
- отклик содержит хотя бы одно из перечисленного ниже:
  - директива public (см. параграф 5.2.2.9);
  - директива private, если кэш не является общим (см. параграф 5.2.2.7);
  - поле заголовка Expires (см. параграф 5.3);
  - директива max-age (см. параграф 5.2.2.1);
  - директива s-maxage, если кэш является общим (см. параграф 5.2.2.10);
  - расширение, разрешающее кэширование(см. параграф 5.2.3); or
  - код статуса, определённый как эвристически кэшируемый (см. параграф 4.2.2).

Отметим, что расширение кэша может переопределять любое из этих требований (см. параграф 5.2.3).

В этом контексте кэш считается понимающим метод запроса или код статуса в отклике, если он распознает и реализует всё связанное с кэшированием поведение.

Отметим, что при нормальной работе некоторые системы кэширования не будут сохранять отклики, не имеющие ни валидатора кэша, ни явного срока действия, поскольку хранить такие отклики обычно бесполезно, но не запрещено.

3.1. Сохранение полей заголовков и трейлеров

Кэш **должен** включать при сохранении все полученные поля заголовка отклика (в том числе, нераспознанные), это гарантирует, что новые поля заголовков HTTP будут внедрены. Однако имеется ряд исключений.

- Поле заголовка Connection и имена указанных в нем полей (в соответствии с параграфом 7.6.1 в [HTTP]) должны удаляться перед пересылкой сообщения. Это **может** быть выполнено перед сохранением.
- Семантика некоторых полей требует их удаления перед пересылкой сообщения и это **может** быть выполнено до сохранения (см. примеры в параграфе 7.6.1 [HTTP]).
- Директивы кэширования no-cache (параграф 5.2.2.4) и private (параграф 5.2.2.7) могут иметь аргументы, препятствующие сохранению полей заголовка всеми или общими кэшами, соответственно.
- Поля заголовков, специфичные для прокси, используемого кэшем при пересылке запроса, сохранять **недопустимо**, если только кэш не включает отождествление прокси в ключ кэша. Фактически это ограничивается полями Proxy-Authenticate (параграф 11.7.1 в [HTTP]), Proxy-Authentication-Info (параграф 11.7.3 в [HTTP]), Proxy-Authorization (параграф 11.7.2 в [HTTP]).

Кэш **может** сохранять поля трейлера отдельно от полей заголовка или отбрасывать их. В кэше **недопустимо** объединять поля трейлера с полями заголовка.

3.2. Обновление сохранённых полей заголовков

В некоторых ситуациях кэш должен обновлять сохранённые поля заголовков по другому (обычно, более новому) отклику, например, в случаях, описанных в параграфах 3.4, 4.3.4, 4.3.5. В таких случаях кэш **должен** добавлять каждое поле заголовка из представленного отклика (за исключением указанных ниже) в сохранённый отклик с заменой уже имеющихся в нем полей.

- Поля заголовка, ожидаемые от хранилища (параграф 3.1).
- Поля заголовка, от которых зависит сохранённый в кэше отклик, как описано ниже.
- Поля заголовка, автоматически обрабатываемые и удаляемые получателем, как описано ниже.
- Поле заголовка Content-Length.

В некоторых случаях в кэше (особенно у пользовательских агентов) хранятся результаты обработки полученного отклика, а не сам отклик, и обновление полей заголовков, влияющих на обработку, может приводить к несогласованному поведению и проблемам безопасности. В таких ситуациях кэш **может** исключать такие поля заголовка из обновления, но **следует** ограничиваться исключением лишь полей, которые влияют на целостность сохранённого отклика. Например, браузер может декодировать содержимое отклика при его получении, что приведёт к разрыву связи между сохраняемыми данными и метаданными исходного отклика. Обновление сохранённых метаданных с использованием другого поля заголовка Content-Encoding может вызвать проблемы. Аналогично, браузер может хранить дерево HTML после разбора, а не содержимое, принятое в отклике и обновление поля заголовка Content-Type в этом случае будет невозможным, поскольку любые допущения о формате, принятые при разборе, окажутся недействительными.

Кроме того, некоторые поля, например, Content-Range, автоматически обрабатываются и удаляются реализацией HTTP. Реализации **могут** автоматически опускать такие поля, даже если они на деле не обрабатываются.

Отметим, что префикс Content-* не является сигналом об исключении поля заголовка из обновления, это соглашение предназначено для полей заголовков MIME, а не HTTP.

3.3. Сохранение неполных откликов

Если метод запроса - GET, код статуса в отклике - 200 (OK) и раздел заголовков запроса получен целиком, в кэше **можно** сохранить неполный отклик (параграф 6.1 в [HTTP]), если при записи неполнота будет указана. Аналогично, **можно** сохранить отклик 206 (Partial Content) как неполный отклик 200 (OK). Однако в кэше **недопустимо** сохранять неполные отклики или отклики с частью содержимого, если в них нет полей заголовка Range и Content-Range или непонятны единицы диапазона в этих полях.

Кэш **может** дополнять сохранённые неполные отклик, выполняя последующий запрос диапазона (параграф 14.2 в [HTTP]) и объединяя отклик о его успехе с сохраненным откликом, как указано в параграфе 3.4. В кэше **недопустимо** использовать неполный отклик на запрос, пока этот отклик не был завершён, или запрос не являлся частичным с указанием диапазона, полностью включённого в неполный отклик. Кэшу **недопустимо** передавать частичный отклик клиенту без явной маркировки его кодом статуса 206 (Partial Content).

3.4. Объединение частичного содержимого

Отклик может передавать лишь часть представления, если соединение разорвано преждевременно или в запросе указано одно или несколько полей Range (параграф 14.2 в [HTTP]). После нескольких таких передач в кэше может оказаться несколько диапазонов одного представления. Кэш **может** объединить эти диапазоны в один сохранённый отклик и использовать его для выполнения будущих запросов, если в них будет тот же строгий валидатор и кэш соответствует требованиям клиента из параграфа 15.3.7.3 в [HTTP]. При создании объединённого отклика из сохранённых откликов кэш **должен** обновить поля заголовка сохраняемого отклика, используя поля из нового отклика, как указано в параграфе 3.2.

3.5. Сохранение откликов на аутентифицированные запросы

В общем кэше **недопустимо** использовать кэшированный отклик на запрос с полем заголовка Authorization (параграф 11.6.2 в [HTTP]) для выполнения последующих запросов, если отклик не содержит поля Cache-Control с директивой (параграф 5.2.2), разрешающей сохранение в общем кэше, и соответствии этого кэша требованиям директивы для данного отклика. В этой спецификации заданы директивы с таким эффектом: must-revalidate (параграф 5.2.2.2), public (параграф 5.2.2.9), s-maxage (параграф 5.2.2.10).

4. Создание откликов из кэша

При представлении запроса кэшу **недопустимо** использовать сохранённый отклик, если не выполняются все указанные ниже условия.

- Представленное значение URI цели (параграф 7.1 в [HTTP]) совпадает с сохраненным откликом.
- Связанный с сохраненным откликом метод запроса позволяет использовать отклик для представленного запроса.
- Поля заголовка запроса, указанные в сохранённом отклике (при наличии) совпадают с представленными полями (см. параграф 4.1).
- В сохранённом отклике нет директивы no-cache (параграф 5.2.2.4), если только он не был успешно проверен (параграф 4.3).
- Сохранённый отклик является одним из указанных ниже:
  - fresh (см. параграф 4.2);
  - разрешён для обслуживания при устаревании (см. параграф 4.2.4);
  - успешно обновлён (см. параграф 4.3).

Отметим, что расширение кэша может переопределять любые из указанных требований (см. параграф 5.2.3).

Когда сохранённый отклик используется для выполнения запроса без проверки, кэш **должен** генерировать поле заголовка Age (параграф 5.1), заменяя любое имеющееся в отклике значение на current_age из сохранённого отклика (см. параграф 4.2.3).

Кэш **должен** записывать запросы к серверу-источнику с небезопасными методами (параграф 9.2.1 в [HTTP]), т. е. ему не разрешается генерировать отклик на такой запрос до пересылки запроса и получения соответствующего отклика. Следует также отметить, что небезопасные запросы могут аннулировать уже сохранённые отклики (см. параграф 4.4).

Кэш может использовать сохранённый или сохраняемый отклик для выполнения нескольких запросов при условии разрешения на повторное использование данного отклика для этих запросов. Это позволяет кэшу сворачивать (collapse) запросы, т. е. объединять несколько входящих запросов в один пересылаемый запрос при отсутствии записи в кэше, снижая тем самым нагрузку на сеть и сервер-источник. Однако, если кэш не может использовать возвращённый отклик для части или всех свёрнутых запросов, ему потребуется переслать запросы для их выполнения, что может приводить к дополнительным задержкам.

Если хранится более одного подходящего отклика, кэш **должен** выбрать последний (по полю заголовка Date) и может также переслать запрос с Cache-Control: max-age=0 или Cache-Control: no-cache для выбора используемого отклика.

Кэш без часов (параграф 5.6.7 в [HTTP]) **должен** перепроверять сохранённые отклики при каждом их использовании.

4.1. Расчёт ключей кэша с полем заголовка Vary

При получении кэшем запроса, который можно выполнить по сохранённому отклику, и этот отклик включает поле заголовка Vary (параграф 12.5.5 в [HTTP]), **недопустимо** использовать сохранённый отклик без проверки совпадения полей нового запроса, указанных Vary, с соответствующими полями исходного запроса (вызвавшего сохранение отклика). Поля двух запросов считаются совпадающими лишь в том случае, когда поля первого запроса могут быть преобразованы в поля второго с использованием любого из указанных ниже методов:

- добавление или удаление пробелов, разрешённое синтаксисом поля заголовка;
- объединение нескольких строк заголовка с одноимённым полем (см. параграф 5.2 в [HTTP]);
- нормализация полей обоих заголовков так, чтобы они имели идентичную семантику в соответствии со спецификацией поля (например, смена порядка значений, если он не важен, смена регистра, когда он не учитывается).

Если поле заголовка (после указанной нормализации) отсутствует в запросе, оно может соответствовать только отсутствию поля и в другом запросе. Сохранённый запрос с полем заголовка Vary, содержащим *, никогда не будет соответствовать.

Если совпадает несколько сохранённых откликов, кэш должен выбрать один из них. Если для указанного поля заголовка имеется механизм ранжирования (например, qvalue в Accept или похожих полях), этот механизм **можно** использовать для выбора предпочтительного отклика. Если такого механизма нет или предпочтения одинаковы, выбирается наиболее свежий отклик (по полю Date), как указано в разделе 4.

Некоторые ресурсы ошибочно опускают поле Vary в принятом по умолчанию отклике (т. е. в отклике на запрос без предпочтений), в результате чего этот отклик выбирается для последующих запросов даже при наличии более предпочтительного отклика. Когда в кэше имеется несколько сохранённых откликов для URI цели и в каких-либо из них нет поля Vary, кэшу **следует** выбирать наиболее свежий (см. параграф 4.2.3) отклик с действительным полем Vary.

Если нет соответствующего сохранённого отклика, кэше не может выполнить представленный запрос. Обычно такой запрос пересылается серверу-источнику, возможно, с добавлением предварительных условий для описания откликов, имеющихся в кэше (параграф 4.3).

4.2. fresh

«Свежим» считается отклик, возраст которого ещё не превысил срок свежести. Иные отклики считаются старыми. Сроком свежести отклика считается интервал времени между генерацией отклика сервером-источником и завершением срока действия. Явным сроком действия ACK называют время, по истечении которого сервер-источник считает, что сохранённый отклик больше не может использоваться кэшем без дополнительной проверки, а эвристический срок действия задаётся кэшем при недоступности явного срока действия. Возрастом отклика называют время, прошедшее с момента его создания или успешной проверки сервером-источником. Свежий отклик может применяться для ответов на последующие запросы без обращения к северу-источнику, что повышает эффективность.

Основным механизмом проверки свежести отклика является явно указанный сервером-источником срок свежести в поле заголовка Expires (параграф 5.3) или директиве отклика max-age (параграф 5.2.2.1). Обычно сервер-источник назначает откликам явное время завершения срока действия в будущем, полагая, что представление вряд ли изменится семантически значимым образов в течение срока действия. Если сервер-источник хочет заставить кэш проверять каждый запрос, он может указать завершение срока действия в прошлом, чтобы указать, что отклик уже устарел. Соответствующий спецификации кэш обычно проверяет устаревшие отклики перед применением для последующих запросов (см. параграф 4.2.4).

Поскольку серверы-источники не всегда указывают явный срок действия, кэш может использовать эвристику для определения срока действия в некоторых обстоятельствах (см. параграф 4.2.2). Расчет свежести отклика имеет вид

```
response_is_fresh = (freshness_lifetime > current_age)
```

Параметр freshness_lifetime задан в параграфе 4.2.1, current_age - в параграфе 4.2.3.

Клиенты могут передавать запросы с директивами max-age или min-fresh (параграф 5.2.1) для указания ограничений при расчёте свежести соответствующего отклика. Однако кэш не обязан соблюдать их. При расчёте свежести следует избегать общих проблем анализа дат, указанных ниже.

- Хотя все форматы дат задают учёт регистра символов, получателю кэша **следует** сравнивать значения полей без учёта регистра.
- Если внутренняя реализация часов у получателя кэша имеет меньшее разрешение, чем значение HTTP-date, получатель **должен** представить дату Expires как ближайшее время не позже полученного значения.
- Получателю кэша **недопустимо** разрешать влияние локального часового пояса при расчёте и сравнении возраста или срока действия.
- Получателю кэша **следует** считать даты с обозначением часового пояса, отличным от GMT, недействительными для расчёта срока действия.

Отметим, что свежесть относится только к работе кэша и не может служить для того, чтобы вынудить агент пользователя обновить своё отображение или заново загрузить ресурс. Различия между кэшированием и механизмами истории рассмотрены в разделе 6.

4.2.1. Расчёт срока свежести

Кэш может рассчитывать срок свежести (freshness_lifetime) отклика по приведённым ниже правилам до совпадения.

- Если кэш является общим и имеется директива s-maxage (параграф 5.2.2.10) применяется её значение.
- При наличии в отклике директивы max-age (параграф 5.2.2.1) применяется её значение.
- При наличии поля заголовка Expires (параграф 5.3) применяется его значение за вычетом значения поля Date в отклике (времени приёма сообщения, если этого поля нет, как указано в параграфе 6.6.1 [HTTP])
- В иных случаях явный срок свежести не указан и можно применять эвристическое значение (параграф 4.2.2).

Отметим, что этот расчёт предназначен для снижения расхождения часов путём использования сведений о времени от сервера-источника, когда это возможно.

При наличии более одного значения (например, двух строк поля Expires или нескольких директив Cache-Control: max-age) применяется первое из них или отклик считается устаревшим. Если директивы конфликтуют (например, имеются max-age и no-cache), следует применять более ограничительную из них. Кэшу рекомендуется считать устаревшими отклики с недействительным сроком свежести (например, max-age с нецелочисленным значением).

4.2.2. Расчёт эвристической свежести

Серверы-источники не всегда явно указывают срок свежести, поэтому кэш **может** назначать в таких случаях эмпирическое значение срока на основе алгоритмов, использующих значения других полей (например, Last-Modified) для правдоподобной оценки срока свежести. Эта спецификация не задаёт конкретный алгоритм, но вносит ограничения на результат для худших случаев.

Кэшу **недопустимо** применять эвристику для определения свежести при наличии в сохранённом отклике явного срока свежести. В соответствии с требованием раздела 3 эвристика может применяться лишь для откликов без явного срока свежести, в который коды статуса указаны как эвристически кэшируемые (см. параграф 15.1 в [HTTP]), и откликов без явного срока свежести, помеченных как кэшируемые (например, с помощью директивы public).

Отметим, что в прежних спецификациях эвристически кэшируемые отклики назывались кэшируемыми по умолчанию.

Если в отклике имеется поле заголовка Last-Modified (параграф 8.8.2 в [HTTP]), кэшу рекомендуется использовать значение эвристического срока свежести, которое не превышает некую долю прошедшего с этого момента времени (обычно используется 10%).

> Примечание. Предыдущая версия спецификации HTTP (параграф 13.9 в [RFC2616]) запрещала кэшам вычислять эвристический срок свежести для URI с компонентами запроса (т. е. включающими ?). На практике это не получило широкого распространения, поэтому серверам-источникам рекомендуется передавать явные директивы (например, Cache-Control: no-cache) для предотвращения кэширования.

4.2.3. Расчёт возраста

Поле заголовка Age служит для передачи оценки возраста сообщения с откликом, полученного из кэша. Age содержит оценку кэшем числа секунд с момента генерации или проверки отклика сервером-источником, которая определяется суммой времени пребывания этого отклика в каждом кэше на пути от сервера-источника и времени, затраченного на передачу через сеть. При расчёте Age применяются указанные ниже данные.

age_value
: Значение поля заголовка Age (параграф 5.1) в пригодной для арифметических действий форме или 0 (при недоступности.

date_value
: Значение поля заголовка Date в пригодной для арифметических действий форме. Определение поля и требования к откликам без него приведены в параграфе 6.6.1 [HTTP].

now
: Текущее время по часам реализации (параграф 5.6.7 в [HTTP]).

request_time
: Значение часов в момент запроса, вызвавшего сохранённый отклик.

response_time
: Значение часов в момент получения отклика.

Возврast отклика можно рассчитать двумя полностью независимыми способами.

1. Видимый возраст apparent_age = response_time и date_value, если часы реализации достаточно точно синхронизированы с часами сервера-источника. При получении отрицательного значения принимается 0.

2. Уточнённый возраст corrected_age_value, если все кэши на пути реализуют HTTP/1.1 или выше. Кэш **должен** интерпретировать это значение относительно времени инициирования запроса, а не времени приёма отклика.

```
apparent_age = max(0, response_time - date_value);
response_delay = response_time - request_time;
corrected_age_value = age_value + response_delay;
```

Значение corrected_age_value может применяться как corrected_initial_age. В ситуациях, где имеются очень старые реализации кэшей, некорректно устанавливающие Age, значение corrected_initial_age может вычисляться более консервативно как

```
corrected_initial_age = max(apparent_age, corrected_age_value);
```

Значение current_age для сохранённого отклика можно после этого рассчитать путём сложения corrected_initial_age с числом секунд с момента последнего обновления сохранённого отклика сервером источником.

```
resident_time = now - response_time;
current_age = corrected_initial_age + resident_time;
```

4.2.4. Работа с устаревшими откликами

Устаревшим (stale) считается отклик, явно содержащий сведения об истечении срока действия или разрешающий эвристическую оценку срока действия, которая указывает его несвежесть в соответствии с расчётами из параграфа 4.2.

Кэшу **недопустимо** генерировать устаревший отклик, если это запрещено явной директивой в протоколе (например, no-cache, must-revalidate, применимой директивой s-maxage или proxy-revalidate, см. параграф 5.2.2). Кэшу **недопустимо** генерировать устаревший отклик, если кэш не отсоединён или это не разрешено явно клиентом или сервером-источником (например, директивой max-stale из параграфа 5.2.1, директивами расширения вроде описанных в [RFC5861] или конфигурацией в соответствии с отдельным (out-of-band) соглашением.

4.3. Проверка

При наличии в кэше хотя бы одного сохранённого отклика для запрошенного URI и невозможности применить его (например, по причине несвежести или невозможности выбрать, см. параграф 4.1), можно использовать механизм условного запроса (раздел 13 в [HTTP]) при пересылке, чтобы дать следующему приёмному серверу возможность

выбрать пригодный для использования сохранённый отклик, обновляющий обрабатываемые сохранённые метаданные или заменяющий сохранённый отклик(и). Это процесс называется проверкой или перепроверкой сохранённого отклика.

4.3.1. Передача запроса на проверку

При создании условного запроса на проверку кэш начинает с запроса, который он пытается выполнить, или (если запрос создаётся независимо) синтезирует запрос, используя сохранённый отклик путём копирования метода, URI цели и полей заголовка, указанных полем заголовка Vary (параграф 4.1). Затем в запрос включается одно или несколько полей заголовка с условиями. Эти поля содержат валидатор данных, полученный из сохранённых откликов с тем же URI. Обычно включаются лишь сохранённые отклики с тем же ключом кэша, хотя разрешается проверять отклики, которые невозможно выбрать с помощью передаваемых полей заголовка запроса (см. параграф 4.1).

Поля условий из заголовка сравниваются получателями для определение наличия сохранённого отклика, эквивалентного текущему представлению ресурса. Одним из валидаторов является метка времени из поля заголовка Last-Modified (параграф 8.8.2 в [HTTP]), которую можно указать в поле заголовка If-Modified-Since для проверки отклика или в поле If-Unmodified-Since или If-Range для выбора представления (т. е. клиент обращается к конкретному полученному ранее представлению по этой метке). Другим валидатором является тег сущности в поле заголовка ETag (параграф 8.8.3 в [HTTP]). Один или несколько тегов сущностей, указывающих сохранённые отклики, можно использовать в поле заголовка If-None-Match для проверки отклика или в поле If-Match или If-Range для выбора представления (т. е. клиент указывает конкретно одно или несколько ранее полученных представлений с теми же тегами сущности).

При генерации условного запроса для проверки:

- **должны** передаваться соответствующие теги сущностей (в If-Match, If-None-Match, If-Range), если такие теги были представлены в проверяемых сохранённых откликах;
- **следует** передавать значение Last-Modified (в If-Modified-Since), если запрос не относится к проверяемому поддиапазону и отклик содержит значение Last-Modified;
- **можно** передавать значение Last-Modified (в If-Unmodified-Since или If-Range), если запрос относится к поддиапазону, проверяется один сохранённый отклик и он содержит только Last-Modified (без тега сущности).

В большинстве случаев оба валидатора включаются в запрос проверки кэша (даже при явном преобладании тегов сущностей), чтобы старые посредники, не понимающие условий с тегами сущностей, могли реагировать должным образом.

4.3.2. Обработка полученных запросов на проверку

Каждый клиент в цепочке запросов может иметь свой кэш, поэтому обычно кэш посредника получает условные запросы от других (передающих - outbound) кэшей. Некоторые пользовательские агенты тоже применяют условные запросы для ограничения передачи данных, чтобы принимать лишь недавно обновлённые представления или завершать передачу полученного частично представления.

Если кэш получает запрос, который можно выполнить путём повторного использования сохранённого отклика 200 (OK) или 206 (Partial Content) в соответствии с параграфом 4, ему **следует** оценить и применить поля заголовка с условиями по имеющимся валидаторам, содержащимся в сохранённом запросе. Кэшу **недопустимо** оценивать условные поля заголовков, применимые лишь к серверу-источнику, а также поля, встречающиеся в запросе с семантикой, невыполнимой с кэшированным откликом, или в запросе с целевым ресурсом, для которого нет сохранённых откликов. Такие условия, вероятно, предназначены для другого принимающего (inbound) сервера.

Корректная оценка условных запросов кэшем зависит от полученных полей заголовка с условиями и их приоритета. Поля заголовка If-Match и If-Unmodified-Since не применимы к кэшу, поэтому запрос **должен** пересылаться источнику, а поле If-None-Match имеет преимущество перед If-Modified-Since. Полная спецификация приоритетов условий приведена в параграфе 13.2.2 [HTTP][1].

Запрос с полем If-None-Match (параграф 13.1.2 в [HTTP]) указывает, что клиент хочет сравнить один или несколько сохранённых у него откликов с сохраненным откликом, выбранным кэшем (в соответствии с параграфом 4). Если этого поля нет, но имеется поле If-Modified-Since (параграф 13.1.3 в [HTTP]), запрос показывает, что клиент хочет проверить один или несколько сохранённых им откликов по датам изменения.

Если запрос содержит поле заголовка If-Modified-Since, а в сохранённом отклике нет поля Last-Modified, для проверки условий **следует** использовать поле Date из сохранённого отклика (или время получения этого отклика, если Date нет).

Кэш, поддерживающий частичные отклики на запросы с диапазоном в соответствии с параграфом 14.2 в [HTTP], должен сравнивать полученное поле заголовка If-Range (параграф 13.1.5 в [HTTP]) с выбранным им откликом.

Когда кэш решает переслать запрос на перепроверку своих сохранённых откликов для запроса со списком тегов сущностей If-None-Match, он **может** объединить полученный список со списком тегов сущностей из сохранённого им набора откликов (свежих и старых) и передать совокупный список как замену поля If-None-Match в пересылаемом запросе. Если сохранённый отклик включает лишь часть содержимого, кэшу **недопустимо** включать свой тег сущности в объединение, если только запрос с диапазоном не был полностью выполнен сохраненным частичным откликом. Если отклик на пересланный запрос имеет код 304 (Not Modified) и содержит поле ETag с тегом сущности, отсутствующим в списке клиента, кэш **должен** генерировать отклик 200 (OK) для клиента, используя соответствующий сохранённый отклик, обновленный метаданными из отклика 304 (параграф 4.3.4).

4.3.3. Обработка откликов о проверке

Обработка откликов на условные запросы зависит от полученного в них кода статуса.

- Код 304 (Not Modified) указывает, что сохранённый отклик можно обновить и использовать снова (параграф 4.3.4).

[1] В оригинале это предложение несколько отличается, см. https://www.rfc-editor.org/errata/eid7695. *Прим. перев.*

- Полный отклик (включающий содержимое) показывает, что не подходит ни один из сохранённых откликов, отмеченных в условном запросе. Кэш **должен** использовать для выполнения запроса полный отклик и может сохранить этот отклик в соответствии со своими ограничениями (см. раздел 3).

- Если кэш получает код 5xx (Server Error) при попытке проверить отклик, он может переслать полученный отклик запрашивающему клиенту или действовать как при отсутствии отклика от сервера. Во втором случае кэш может передать сохранённый ранее отклик с учётом ограничений (параграф 4.2.4) или повторить запрос на проверку.

4.3.4. Обновление сохранённых откликов после проверки

Когда кэш получает отклик 304 (Not Modified), ему необходимо идентифицировать сохранённые отклики, которые можно обновить по этому отклику, и выполняет обновления. В исходный набор для обновления входят отклики, которые могли быть выбраны для данного запроса, т. е. соответствующие требованиям раздела 4, за исключением последнего требования быть свежими, подходящими для обслуживания несвежих или просто обновлёнными. Затем отклики исходного набора фильтруются по приведённым ниже условиям (до первого совпадения).

- Если новый отклик содержит хотя бы один строгий валидатор (см. параграф 8.8.1 в [HTTP]), каждый из этих валидаторов указывает выбранное представление для обновления. Обновление применяется ко всем сохраненным откликам, имеющим один из таких строгих валидаторов. Если ни в одном из откликов исходного набора нет таких строгих валидаторов, кэшу **недопустимо** использовать новый отклик для обновления.

- Если в новом отклике нет строгих валидаторов, но имеется хотя бы один слабый валидатор и эти валидаторы соответствуют одному из откликов исходного набора, для обновления указывается последний из соответствующих сохранённых откликов.

- Если в новом отклике совсем нет валидаторов (например, когда клиент генерирует запрос If-Modified-Since по источнику, отличному от поля Last-Modified в заголовке отклика) и в исходном наборе имеется лишь один сохранённый отклик, в котором тоже нет валидатора, сохранённый отклик указывается для обновления.

Для каждого идентифицированного сохранённого отклика кэш **должен** обновить поля заголовка значениями полей из отклика 304 (Not Modified), как указано в параграфе 3.2.

4.3.5. Обновление откликов с HEAD

Отклик для метода HEAD идентичен отклику на запрос GET, но не передаёт содержимого. Это свойство откликов HEAD может служить для аннулирования или обновления кэшированных откликов GET при недоступности более эффективного механизма условных запросов (из-за отсутствия валидаторов в сохранённом отклике) или передача содержимого нежелательна даже при его изменении.

Когда кэш выполняет входящий запрос HEAD для URI цели и получает отклик 200 (OK), ему **следует** обновить или аннулировать каждый из сохранённых откликов GET, которые могли быть выбраны для этого запроса (параграф 4.1). Для каждого из сохранённых откликов, который мог быть выбран, кэшу **следует** обновить сохранённый отклик, как описано ниже, если сохранённый отклик и отклик HEAD имеют совпадающие значения любых полученных полей валидаторов (ETag и Last-Modified) и отклик HEAD имеет поле заголовка Content-Length, значение которого соответствует Content-Length в сохранённом отклике. В противном случае кэшу **следует** считать сохранённый отклик устаревшим.

Если кэш обновляет сохранённый отклик по метаданным из отклика HEAD, он **должен** использовать для обновления поля заголовка из отклика HEAD (см. параграф 3.2).

4.4. Аннулирование сохранённых откликов

Поскольку небезопасные методы запросов (параграф 9.2.1 в [HTTP]), такие как PUT, POST и DELETE могут изменять состояние сервера-источника, промежуточные кэши должны аннулировать сохранённые отклики для поддержки актуальности их содержимого. Кэш **должен** аннулировать URI цели (параграф 7.1 в [HTTP]) при получении не связанного с ошибкой кода статуса в отклике на небезопасный метод (включая методы, о безопасности которых не известно) и **может** аннулировать другие URI в таких случаях. В частности, кандидатами на аннулирование являются URI в полях заголовка Location и Content-Location (при наличии), другие URI могут обнаруживаться механизмами, не заданными в этом документе. Однако кэшу **недопустимо** инициировать аннулирование при этих условиях, если источник (параграф 4.3.1 в [HTTP]) из URI для аннулирования отличается от URI цели (параграф 7.1 в [HTTP]). Это помогает предотвратить DoS[1]-атаки.

Аннулирование означает, что кэш будет удалять сохранённые отклики с URI цели, совпадающим с данным URI или помечать их как недействительные и нуждающиеся в обязательной проверке перед отправкой в ответ на запрос.

Не связанными с ошибками откликами считаются отклики с кодами статуса 2xx (Successful) и 3xx (Redirection).

Отметим, что это не гарантирует глобальное аннулирование всех соответствующих откликов и меняющие состояние запросы будут аннулировать лишь отклики в кэшах, через которые эти запросы проходят.

5. Определения полей

В этом разделе определяется синтаксис и семантика полей HTTP, связанных с кэшированием.

5.1. Age

Поле заголовка откликов Age передаёт оценку отправителем времени, прошедшего с момента генерации или успешной проверки сервером-источником. Значение Age рассчитывается в соответствии с параграфом 4.2.3.

```
Age = delta-seconds
```

Поле Age содержит неотрицательное целое число, указывающее время в секундах (см. параграф 1.2.2).

[1]Denial-of-service - отказ в обслуживании.

Хотя поле определено как одиночное, кэшу, встретившему сообщение со списочным полем Age, **следует** взять из списка первой значение и отбросить остальные. Если значение поля (после отбрасывания лишнего) является недействительным (например, не является неотрицательным целым числом), кэшу **следует** игнорировать поле.

Наличие поля Age подразумевает, что этот отклик не был создан или проверен сервером для данного запроса. Однако отсутствие Age не подразумевает контакта с источником.

5.2. Cache-Control

Поле заголовка Cache-Control служит для перечисления директив для кэшей в цепочке запрос-отклик. Директивы кэша являются однонаправленными в том смысле, что наличие директивы в запросе не предполагает её включения в отклик. Сведения об обработке директив Cache-Control, определённых в других местах, приведены в параграфе 5.2.3.

Прокси (независимо от реализации в них кэша) **должны** передавать директивы кэширования в пересылаемых откликах, независимо от их значимости для данного приложения, поскольку директивы могут применяться для всех получателей в цепочке запрос-отклик. Невозможно задать директиву только для конкретного кэша.

Директивы кэширования задаются маркером (без учёта регистра символов) и могут иметь аргумент, который может быть задан маркером или строкой в кавычках. Для определённых ниже директив с аргументами получатель должен воспринимать обе формы, даже если при генерации требуется определённая форма.

```
Cache-Control   = #cache-directive
cache-directive = token [ "=" ( token / quoted-string ) ]
```

Для описанных ниже директив аргументы не определены (не разрешены), если явно не указано иное.

5.2.1. Директивы запроса

Далее определены директивы кэширования для запроса. Они не обязательны и кэш **может** не реализовать их.

5.2.1.1. max-age

Синтаксис аргумента

```
delta-seconds (см. параграф 1.2.2)
```

Директива запроса max-age указывает, что клиент предпочитает отклике с возрастом не более заданного аргументом директивы числа секунд. Отсутствие директивы запроса max-stale указывает, что клиент не хочет получать устаревшие отклики. Аргумент этой директивы использует маркерную форму, например, max-age=5, а не max-age="5". Отправителю **недопустимо** использовать строку в кавычках.

5.2.1.2. max-stale

Синтаксис аргумента

```
delta-seconds (см. параграф 1.2.2)
```

Директива запроса max-stale указывает, что клиент воспримет отклик с истёкшим сроком свежести. Присутствие значения говорит о том, что клиент готов воспринимать отклики, срок свежести которых истёк не более указанного числа секунд назад. Если значение max-stale не задано, клиент будет воспринимать устаревшие отклики с любым возрастом. Аргумент этой директивы использует маркерную форму, например, max-stale=10, а не max-stale="10". Отправителю **недопустимо** использовать строку в кавычках.

5.2.1.3. min-fresh

Синтаксис аргумента

```
delta-seconds (см. параграф 1.2.2)
```

Директива запроса min-fresh указывает, что клиент предпочитает отклики со сроком свежести не меньше суммы текущего возраста и указанного аргументом числа секунд, т. е. отклики, которые будут сохранять свежесть ещё в течение указанного числа секунд. Аргумент этой директивы использует маркерную форму, например, min-fresh=20, а не min-fresh="20". Отправителю **недопустимо** использовать строку в кавычках.

5.2.1.4. no-cache

Директива запроса no-cache говорит, что клиент предпочитает не использовать для выполнения запроса сохранённые отклики без успешной проверки сервером-источником.

5.2.1.5. no-store

Директива запроса no-store указывает, что кэшу **недопустимо** сохранять какую-либо часть запроса и отклика на него. Директива применяется как для приватных, так и для общих кэшей. **Недопустимость** сохранения в этом контексте означает, что кэшу **недопустимо** намеренно записывать информацию в энергонезависимое хранилище и он **должен** приложить максимум усилий для удаления информации из временного хранилища как можно быстрее после её пересылки. Эта директива не является надёжным и достаточным средством обеспечения приватности. В частности, вредоносные или скомпрометированные кэши могут не распознавать или не исполнять эту директиву, а коммуникационные сети могут быть доступны для прослушивания.

Отметим, что при выполнении запроса с такой директивой из кэша она не применяется к уже сохранённому отклику.

5.2.1.6. no-transform

Директива запроса no-transform указывает, что клиент просит посредников избегать преобразования содержимого (параграф 7.7 в [HTTP].

5.2.1.7. only-if-cached

Директива запроса only-if-cached указывает, что клиент хочет получить только сохранённый отклик. Кэшу, соблюдающему эту директиву **следует** при её получении отвечать сохраненным откликом, соответствующим другим ограничениям для запроса или кодом 504 (Gateway Timeout).

5.2.2. Директивы отклика

Далее определены директивы кэширования для отклика. Кэш **должен** подчиняться директивам Cache-Control, заданным здесь.

5.2.2.1. max-age

Синтаксис аргумента

```
delta-seconds (см. параграф 1.2.2)
```

Директива отклика max-age указывает, что отклик будет сочтён устаревшим, когда его возраст превысит заданное значение (в секундах). Аргумент этой директивы использует маркерную форму, например, max-age=5, а не max-age="5". Отправителю **недопустимо** использовать строку в кавычках.

5.2.2.2. must-revalidate

Директива отклика must-revalidate указывает, что после признания отклика устаревшим кэшу **недопустимо** использовать его для выполнения другого запроса до успешной проверки источником, как указано в параграфе 4.3.

Директива must-revalidate нужна для поддержки надёжной работы некоторых функций протокола. При любых обстоятельствах кэшу **недопустимо** игнорировать must-revalidate. В частности, если кэш отсоединён, он **должен** возвращать ошибку вместо использования старого отклика. В отклике **следует** указывать код статуса 504 (Gateway Timeout), если нет более подходящего кода.

Серверы должны использовать директиву must-revalidate тогда и только тогда, когда неспособность проверить запрос может вызвать некорректные действия, например, неисполнение финансовой транзакции.

Директива must-revalidate позволяет общему кэшу повторно использовать отклик на запрос с полем Authorization (параграф 11.6.2 в [HTTP]) при соблюдении указанного выше требования о проверке (параграф 3.5).

5.2.2.3. must-understand

Директива отклика must-understand ограничивает кэширование отклика случаями, когда кэш понимает и выполняет требования к этому коду статуса в отклике.

Отклику с директивой must-understand **следует** включать также директиву no-store. Когда кэш, реализующий директиву must-understand, получает отклик с такой директивой, ему **следует** игнорировать no-store, если он понимает и реализует требования к кэшированию для кода статуса.

5.2.2.4. no-cache

Синтаксис аргумента

```
#field-name
```

Директива отклика no-cache в неполном формате (без аргумента) указывает, что отклик **недопустимо** использовать для выполнения любого другого запроса без его пересылки для проверки и получения отклика об успехе (см. параграф 4.3). Это позволяет серверу-источнику предотвратить использование отклика кэшем без связи с сервером, даже если кэш настроен на отправку устаревших откликов.

Полная форма директивы no-cache с аргументом, содержащим список имён полей, указывает, что кэш **может** использовать отклик для выполнения последующего запроса (при соблюдении других ограничений на кэширование), если перечисленные в аргументе поля заголовков исключены из последующего отклика или этот отклик успешно перепроверен сервером-источником (обновление или удаление этих полей). Это позволяет серверу-источнику предотвратить повторное использование некоторых полей заголовка в отклике, разрешая кэшировать остальное. Имена полей в аргументе не ограничены заданным этой спецификацией набором. Регистр символов не учитывается.

Директива использует для аргумента строку в кавычках и отправителю **не следует** создавать маркерную форму, даже когда кавычки представляются ненужными для списка с одним элементом.

> Примечание. Кэши часто обрабатываю полную директиву no-cache как неполную, т. е. специальная обработка полной формы не получила широкого распространения.

5.2.2.5. no-store

Директива отклика no-store указывает, что кэшу **недопустимо** сохранять какую либо часть отклика или соответствующего запроса, а также **недопустимо** использовать этот отклик для выполнения другого запроса.

Директива применяется как для приватных, так и для общих кэшей. **Недопустимость** сохранения в этом контексте означает, что кэшу **недопустимо** намеренно записывать информацию в энергонезависимое хранилище и он **должен** приложить максимум усилий для удаления информации из временного хранилища как можно быстрее после её пересылки. Эта директива не является надёжным и достаточным средством обеспечения приватности. В частности, вредоносные или скомпрометированные кэши могут не распознавать или не исполнять эту директиву, а коммуникационные сети могут быть доступны для прослушивания.

Отметим, что директива must-understand в некоторых случаях отменяет no-store (см. параграф 5.2.2.3).

5.2.2.6. no-transform

Директива отклика no-transform указывает, что посредникам (независимо от поддержки кэширования) **недопустимо** преобразовывать содержимое (см. параграф 7.7 в [HTTP]).

5.2.2.7. private

Синтаксис аргумента

```
#field-name
```

Неполная (без аргумента) директива отклика private указывает, что общему кэшу **недопустимо** сохранять отклик (т. е. он предназначен для одного пользователя). Приватный кэш **может** сохранять отклик с учётом ограничений, заданных в

разделе 3, даже если в ином отклик не может быть эвристически кэшируемым в приватно кэше. Полная форма директивы с аргументом в виде списка полей заголовка, указывает, что эти поля предназначены лишь для одного пользователя и общему кэшу **недопустимо** сохранять указанные поля при их наличии в исходном отклике, но можно сохранить часть отклика без этих полей с учётом ограничений, заданных в разделе 3.

Имена полей в аргументе не ограничены заданным этой спецификацией набором. Регистр символов не учитывается.

Директива использует для аргумента строку в кавычках и отправителю **не следует** создавать маркерную форму, даже когда кавычки представляются ненужными для списка с одним элементом.

> Примечание. Приватность в этом контексте относится лишь к местам сохранения отклика и не связана с сохранением приватности содержимого сообщения. Кэши часто обрабатываю полную директиву private как неполную, т. е. специальная обработка полной формы не получила широкого распространения.

5.2.2.8. proxy-revalidate

Директива отклика proxy-revalidate указывает, что после признания отклика устаревшим кэшу **недопустимо** использовать его для выполнения другого запроса до успешной проверки источником, как указано в параграфе 4.3. Это аналогично must-revalidate (параграф 5.2.2.2), но директива proxy-revalidate не применяется к приватным кэшам.

Отметим, что директива proxy-revalidate сама по себе не предполагает, что отклик является кэшируемым. Например, она может применяться вместе с директивой public (параграф 5.2.2.9), позволяя кэшировать отклик и требуя лишь перепроверки устаревших откликов в общем кэше.

5.2.2.9. public

Директива отклика public указывает, что кэш **может** сохранять отклик, даже если это запрещено иными средствами, с учётом ограничений, указанных в разделе 3. Иными словами, директива public явно помечает отклик как кэшируемый. Например, она разрешает общему кэшу повторно использовать отклик на запрос с полем заголовка Authorization (параграф 3.5).

Отметим, что директиву public не требуется добавлять в отклик, который уже является кэшируемым в соответствии с разделом 3. Если отклик с директивой public не имеет явных сведений о свежести, он будет эвристически кэшируемым (параграф 4.2.2).

5.2.2.10. s-maxage

Синтаксис аргумента

```
delta-seconds (см. параграф 1.2.2)
```

Директива отклика s-maxage response указывает, что для общего кэша заданный этой директивой максимальный возраст переопределяет максимальный возраст, заданный директивой max-age или полем заголовка Expires.

Директива s-maxage принимает семантику директивы отклика proxy-revalidate (параграф 5.2.2.8) для общего кэша. Общему кэшу **недопустимо** повторно использовать устаревший отклик с директивой s-maxage для выполнения другого запроса, пока он не был успешно проверен источником, как указано в параграфе 4.3. Директива также разрешает общему кэшу повторно использовать отклик для запроса с полем Authorization с учётом приведённых выше требования к максимальному возрасту и перепроверке (параграф 3.5).

Аргумент этой директивы использует маркерную форму, например, s-maxage=10, а не s-maxage=10. Отправителю **недопустимо** использовать строку в кавычках.

5.2.3. Директивы расширения

Поле заголовка Cache-Control может быть расширено с помощью одной или нескольких директив расширения. Кэш **должен** игнорировать непонятные директивы кэширования.

Информационные расширения (не требующие смены поведения кэша) можно добавлять без изменения семантики других директив.

Расширения поведения предназначены для изменения имеющихся директив кэширования. Представляются сразу новая и старая директива, чтобы приложения, не понимающие новую директиву, могли вести себя в соответствии со старой, а понимающие новую директиву приложения - менять своё поведение с соответствии с новой директивой. Таким образом, расширения имеющихся директив кэширования можно вносить без нарушения работы существующих кэшей. Рассмотрим, например, гипотетическую директиву community, меняющую поведение для директивы private, разрешая (в дополнение к частным кэшам) кэшировать отклик в кеше, общем для указанного сообщества пользователей. Сервер-источник, разрешить сообществу UCI использовать приватные отклики в своём общем кэше может сделать это, включив в отклик Cache-Control: private, community="UCI". Кэш, понимающий новую директиву, будет вести себя в соответствии с ней, прочие будут игнорировать её и следовать директиве private.

Для новых директив расширения следует рассмотреть определения:

- трактовки неоднократного включения директивы;
- применения директивы с аргументом и без аргумента;
- когда требуется аргумент и к чему ведёт его отсутствие;
- относится директива к запросам, откликам или обоим сразу.

5.2.4. Реестр директив кэширования

Реестр Hypertext Transfer Protocol (HTTP) Cache Directive Registry задаёт пространство имён для директив кэширования. Реестр доступен по ссылке <https://www.iana.org/assignments/http-cache-directives>.

Регистрация **должна** включать поля:

- имя директивы кэширования;
- указатель на текст спецификации.

Значения добавляются в пространство имён по процедуре IETF Review ([RFC8126], параграф 4.8).

5.3. Expires

Поле заголовка Expires указывает дату и время, после которого отклик считается устаревшим. Рассмотрение модели свежести откликов приведено в параграфе 4.2. Наличие поля Expires не предполагает, что исходный ресурс изменится или прекратит существование в указанное время, до или после него.

Значением поля Expires служит временная метка HTTP-date, определённая в параграфе 5.6.7 [HTTP]. Требования к разбору для кэша приведены в параграфе 4.2.

```
Expires = HTTP-date
```

Например,

```
Expires: Thu, 01 Dec 1994 16:00:00 GMT
```

Получатель кэша **должен** интерпретировать недействительные форматы дан (особенно 0) как время в прошлом (уже прошло).

Если отклик содержит поле заголовка Cache-Control с директивой max-age (параграф 5.2.2.1), получатель **должен** игнорировать поле заголовка Expires. Если отклик включает директиву s-maxage (параграф 5.2.2.10), получатель общего кэша **должен** игнорировать поле заголовка Expires. В обоих случаях значение поля Expires предназначено лишь для получателей, которые ещё не реализуют поле заголовка Cache-Control.

Серверу-источнику без часов (параграф 5.6.7 в [HTTP]) **недопустимо** создавать поле Expires, если только его значение не представляет фиксированный момент в прошлом (уже прошло) или не связано с ресурсом системой с часами.

Исторически в HTTP требовалось, чтобы значение поля Expires указывало не больше чем на 1 год в будущее. Хотя больший срок свежести сейчас не запрещается, было показано, что очень большие значения вызывают проблемы (например, переполнение часов, использующих 32-битовые значения времени) и многие кэше будут удалять отклик значительно раньше.

5.4. Pragma

Поле заголовка Pragma определено для кэшей HTTP/1.0, чтобы клиенты могли указывать для запроса no-cache (поле Cache-Control не было определено до HTTP/1.1). Однако поддержка Cache-Control не распространена широко, поэтому данная спецификация упраздняет поле Pragma.

> Примечание. Поскольку смысл Pragma: no-cache в откликах не был задан, это поле не обеспечивает надёжной замены для Cache-Control: no-cache.

5.5. Warning

Поле заголовка Warning служило для передачи дополнительных сведений о статусе или преобразовании сообщения, которые могут не отображаться в коде статуса. Данная спецификация отменяет поле, поскольку оно редко применяется и не показывается пользователям. Те же сведения можно получить из других полей, таких как Age.

6. Связи с приложениями и другими кэшами

Приложения, использующие HTTP, часто задаю дополнительные формы кэширования. Например, Web-баузеры часто имеют механизмы истории, такие как кнопка возврата назад (Back), которые могут служить для повторного отображения представления, просмотренного ранее в сессии. В некоторых браузерах Web реализовано кэширование изображений и других активов в рамках просмотра страницы, которые могут (но не обязаны) поддерживать кэширование HTTP.

Требования этой спецификации не обязательно распространяются на использованием приложением данных после их извлечения из кэша HTTP. Например, механизм истории позволяет отображать предшествующее представление, даже когда срок его действия истёк, а приложение может использовать кэшированные данные по завершении срока их свежести. Эта спецификация не запрещает приложениям принимать кэширование HTTP во внимание, например, механизм истории может сообщать пользователю, что представление устарело или соблюдать директивы кэширования (скажем, Cache-Control: no-store).

Однако при кэшировании приложением данных не очевидным и не контролируемым пользователем способом, настоятельно рекомендуется определять отношение к директивам кэширования HTTP так, чтобы не удивлять авторов, ожидающих соблюдения семантики кэширования. Например, вполне разумное кэширование приложением «поверх» HTTP, позволяющее повторно использовать отклики с Cache-Control: no-store для запросов, которые напрямую связаны с получившим этот отклик запросом (скажем, созданных при загрузке той же страницы), скорей всего будет удивительно и непонятно для пользователей и авторов, если разрешить его применять для несвязанных запросов.

7. Вопросы безопасности

Этот раздел предназначен для информирования разработчиков, поставщиков информации и пользователей об известных проблемах безопасности, связанных с кэшированием HTTP. Более общее рассмотрение вопросов безопасности приведено в HTTP/1.1 (раздел 11 в [HTTP/1.1]) и HTTP Semantics (раздел 17 в [HTTP]).

Кэши открывают дополнительный фронт атак, поскольку содержимое кэша является привлекательной целью для вредоносного использования. Поскольку содержимое кэша сохраняется после завершения запроса HTTP, атаки на кэш могут раскрывать сведения спустя долгое время после того, как пользователь сочтёт его удаленным из сети. Поэтому содержимое кэша следует защищать как конфиденциальные сведения.

В частности, приватный кэш, предназначенный для одного пользователя, можно применить для восстановления действий этого пользователя. Поэтому важно, чтобы пользовательские агенты позволяли контролировать такой кэш, например, разрешая удалять сохранённые отклики от всех или некоторых серверов-источников.

7.1. Отравление кэша

Сохранение вредоносного содержимого в кэше может расширять сферу влияния атакующего, охватывая множество пользователей. Такие атаки с отравлением кэша (cache poisoning) происходят, когда злоумышленник использует недостатки реализации, повышенные привилегии или иные методы внедрения откликов в кэш. Особенно эффективно это при использовании общего кэша для распространения вредоносного содержимого множеству клиентов.

Одним из базовых векторов атак с отравлением кэша является использование различий в разборе сообщений на прокси и пользовательских агентах. Требования для HTTP/1.1 в этой части приведены в параграфе 6.3 [HTTP/1.1].

7.2. Атаки по времени

Поскольку одним из основных назначений кэша является оптимизация производительности, использование кэша может приводить к «утечке» сведений о запрошенных ранее ресурсах. Например, если пользователь посетил сайт и его браузер кэшировал некоторые из откликов сайта, а затем пользователь перешёл на другой сайт, тот может попытаться загрузить отклики, которые по его сведениям имеются на первом сайте. Если они загрузятся быстро, можно предположить, что пользователь посетил сайт или даже конкретную страницу на нём. Такие атаки по времени (timing attack) можно смягчить, добавив в ключ кэша дополнительные сведения, такие как отождествление ссылающегося сайта (для предотвращения указанной выше атаки). Это иногда называют двойным ключом (double keying).

7.3. Кэширование деликатных сведений

Недостатки реализации и внедрения (часто вызванные ошибочным пониманием работы кэша) могут приводить к кэшированию деликатных или конфиденциальных сведений (например, свидетельств для аутентификации), открывая их для неуполномоченных сторон.

Отметим, что поле заголовка Set-Cookie [COOKIE] не препятствует кэширования и кэшированный отклик с полем Set-Cookie может использоваться (и это часто бывает) для выполнения последующих запросов к кэшам. Серверам, желающим контролировать кэширование таких откликов, рекомендуется создавать соответствующие поля Cache-Control в заголовках откликов.

8. Взаимодействие с IANA

Контролёром изменений для указанных ниже реестров является IETF (iesg@ietf.org) - Internet Engineering Task Force.

8.1. Регистрация имён полей

Агентство IANA обновило реестр Hypertext Transfer Protocol (HTTP) Field Name Registry <https://www.iana.org/assignments/http-fields>, как указано в параграфе 18.4 в [HTTP], с приведёнными в таблице 1 именами полей.

Таблица 1.

Имя	*Статус*	*Параграф*	*Комментарии*
Age	permanent	5.1	
Cache-Control	permanent	5.2	
Expires	permanent	5.3	
Pragma	deprecated	5.4	
Warning	obsoleted	5.5	

8.2. Регистрация директив кэширования

Агентство IANA обновило реестр Hypertext Transfer Protocol (HTTP) Cache Directive Registry <https://www.iana.org/assignments/http-cache-directives> с процедурой регистрации, указанной в параграфе 5.2.4, и именами директив кэширования, приведёнными в таблице 2.

Таблица 2.

Директива	*Параграф*
max-age	5.2.1.1, 5.2.2.1
max-stale	5.2.1.2
min-fresh	5.2.1.3
must-revalidate	5.2.2.2
must-understand	5.2.2.3
no-cache	5.2.1.4, 5.2.2.4
no-store	5.2.1.5, 5.2.2.5
no-transform	5.2.1.6, 5.2.2.6
only-if-cached	5.2.1.7
private	5.2.2.7
proxy-revalidate	5.2.2.8
public	5.2.2.9
s-maxage	5.2.2.10

8.3. Реестр кодов предупреждений

Агентство IANA добавило в реестр Hypertext Transfer Protocol (HTTP) Warn Codes <https://www.iana.org/assignments/http-warn-codes> приведённое ниже примечание об отмене поля заголовка Warning.

Поле заголовка Warning (и применяемые в нем коды) признано устаревшим для HTTP в соответствии с [RFC9111].

9. Литература

9.1. Нормативные документы

[HTTP] Fielding, R., Ed., Nottingham, M., Ed., and J. Reschke, Ed., "HTTP Semantics", STD 97, RFC 9110, DOI 10.17487/RFC9110, June 2022, <https://www.rfc-editor.org/info/rfc9110>.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <https://www.rfc-editor.org/info/rfc2119>.

[RFC5234] Crocker, D., Ed. and P. Overell, "Augmented BNF for Syntax Specifications: ABNF", STD 68, RFC 5234, DOI 10.17487/RFC5234, January 2008, <https://www.rfc-editor.org/info/rfc5234>.

[RFC7405] Kyzivat, P., "Case-Sensitive String Support in ABNF", RFC 7405, DOI 10.17487/RFC7405, December 2014, <https://www.rfc-editor.org/info/rfc7405>.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/info/rfc8174>.

9.2. Дополнительная литература

[COOKIE] Barth, A., "HTTP State Management Mechanism", RFC 6265, DOI 10.17487/RFC6265, April 2011, <https://www.rfc-editor.org/info/rfc6265>.

[HTTP/1.1] Fielding, R., Ed., Nottingham, M., Ed., and J. Reschke, Ed., "HTTP/1.1", STD 99, RFC 9112, DOI 10.17487/RFC9112, June 2022, <https://www.rfc-editor.org/info/rfc9112>.

[RFC2616] Fielding, R., Gettys, J., Mogul, J., Frystyk, H., Masinter, L., Leach, P., and T. Berners-Lee, "Hypertext Transfer Protocol -- HTTP/1.1", RFC 2616, DOI 10.17487/RFC2616, June 1999, <https://www.rfc-editor.org/info/rfc2616>.

[RFC5861] Nottingham, M., "HTTP Cache-Control Extensions for Stale Content", RFC 5861, DOI 10.17487/RFC5861, May 2010, <https://www.rfc-editor.org/info/rfc5861>.

[RFC7234] Fielding, R., Ed., Nottingham, M., Ed., and J. Reschke, Ed., "Hypertext Transfer Protocol (HTTP/1.1): Caching", RFC 7234, DOI 10.17487/RFC7234, June 2014, <https://www.rfc-editor.org/info/rfc7234>.

[RFC8126] Cotton, M., Leiba, B., and T. Narten, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 8126, DOI 10.17487/RFC8126, June 2017, <https://www.rfc-editor.org/info/rfc8126>.

Приложение A. Сводка ABNF

Ниже приведена сводка правил ABNF, расширенных в соответствии с параграфом 5.6.1 в [HTTP].

```
Age = delta-seconds
Cache-Control = [ cache-directive *( OWS "," OWS cache-directive ) ]
Expires = HTTP-date
HTTP-date = <HTTP-date, see [HTTP], Section 5.6.7>
OWS = <OWS, see [HTTP], Section 5.6.3>
cache-directive = token [ "=" ( token / quoted-string ) ]
delta-seconds = 1*DIGIT
field-name = <field-name, see [HTTP], Section 5.1>
quoted-string = <quoted-string, see [HTTP], Section 5.6.4>
token = <token, see [HTTP], Section 5.6.2>
```

Приложение B. Отличия от RFC 7234

Уточнена обработка дубликатов и конфликтующих директив кэширования (параграф 4.2.1).

Аннулирование URI в полях заголовка Location и Content-Location больше не требуется, но разрешено (параграф 4.4). Такое аннулирование запрещено, если источник отличается (раньше было хост) параграф 4.4).

Уточнена обработка недействительных и множественных значений поля Age (параграф 5.1).

Для некоторых директив кэширования, определённых в этой спецификации, задан более строгий запрет на генерацию значений в кавычках, так как стало известно о связанных с этим проблемах. Потребители расширений директив кэша больше не обязаны воспринимать обе формы (маркер и строка в кавычках), но всё равно должны корректно разбирать их для неизвестных расширений (параграф 5.2).

Уточнены директивы общего и приватного кэша, чтобы они не допускали повторного использования откликов ни при каких условиях (параграф 5.2.2).

Добавлена директива must-understand и от кэшей больше не требуется понимать семантику новых кодов статуса, если они не присутствуют (параграф 5.2.2.3).

Заголовок отклика Warning признан устаревшим. Большая часть сведений, поддерживаемых Warning, может быть получена проверкой отклика, остальные, хотя и могли быть полезными, служили лишь рекомендациями. На практике поле Warning не добавлялось кэшами и посредниками (параграф 5.5).

Благодарности

См. одноимённый раздел в [HTTP].

Предметный указатель

Адреса авторов

Roy T. Fielding (editor)
Adobe
345 Park Ave
San Jose, CA 95110
United States of America
Email: fielding@gbiv.com
URI: https://roy.gbiv.com/

Mark Nottingham (editor)
Fastly
Prahran
Australia
Email: mnot@mnot.net
URI: https://www.mnot.net/

Julian Reschke (editor)
greenbytes GmbH
Hafenweg 16
48155 Münster
Germany
Email: julian.reschke@greenbytes.de
URI: https://greenbytes.de/tech/webdav/

Перевод на русский язык

Николай Малых

nmalykh@protokols.ru