Internet Engineering Task Force (IETF)
Request for Comments: 10008
Category: Standards Track
ISSN: 2070-1721

J. Reschke
greenbytes
J.M. Snell
Cloudflare
M. Bishop
Akamai

June 2026

The HTTP QUERY Method

Метод HTTP QUERY

Аннотация

В этой спецификации определяется метод QUERY для HTTP. QUERY запрашивает у целевого объекта обработку вложенного содержимого безопасным и идемпотентным способом с выдачей результата обработки. Это похоже на запросы POST, но QUERY можно автоматически повторять или перезапускать без учёта частичной смены состояния.

Статус документа

Документ содержит проект стандарта 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/rfc10008.

Оглавление

[1] Internet Engineering Task Force - комиссия по решению инженерных задач Internet.
[2] Internet Engineering Steering Group - комиссия по инженерным разработкам Internet.

1. Введение

Эта спецификация определяет метод HTTP QUERY как средство создания безопасного идемпотентного запроса (раздел 9.2 в [HTTP]), содержащего представление, которое описывает, как запрос должен обрабатываться целевым ресурсом. Базовый шаблон запроса имеет вид

```
GET /feed?q=foo&limit=10&sort=-published HTTP/1.1
Host: example.org
```

При большом объёме передаваемых данных их трудно закодировать в URI и этот шаблон становится проблематичным:

- ограничения по размеру зачастую не известны заранее, поскольку запрос может проходить через множество несогласованных систем (отметим, что в параграфе 4.1 [HTTP] отправителям и получателям рекомендуется поддерживать запросы размером по меньшей мере 8000 октетов);
- представление некоторых типов данных в целевом URI неэффективно из-за издержек, связанных с правилами кодирования данных в URI;
- URI запросов заносятся в журнальные файлы с большей вероятностью, чем содержимое запроса, и могут также отображаться в закладках;
- при кодировании запросов напрямую в URI каждая возможная комбинация входных данных запроса преобразуется в отдельные ресурсы.

В качестве альтернативы использованию метода GET многие реализации применяют для запросов метод HTTP POST, как показано ниже. В этом случае входные данные для операции запроса передаются как содержимое запроса, а не компоненты URI. Типичное использование метода HTTP POST для запроса имеет вид

```
POST /feed HTTP/1.1
Host: example.org
Content-Type: application/x-www-form-urlencoded

q=foo&limit=10&sort=-published
```

Однако в этом варианте без специальных знаний о ресурсе и сервере, куда передаётся запрос, не всегда можно понять, что выполняется безопасный идемпотентный запрос.

Метод QUERY предоставляет решение, устраняющее разрыв между использованием GET и POST, а приведённый выше пример можно представить как

```
QUERY /feed HTTP/1.1
Host: example.org
Content-Type: application/x-www-form-urlencoded

q=foo&limit=10&sort=-published
```

Как и в методе POST, входные данные для операции передаются в содержимом запроса, а не в виде части URI. Однако, в отличие от POST, этот метод явно безопасен и идемпотентен, позволяя использовать такие функции, как кэширование и автоматические повторы.

Исходя из принципа разработки, в соответствии с которым каждый важный ресурс должен указываться URI, эта спецификация описывает, как сервер может назначать URI самому запросу и конкретному результату запроса для последующего использования в методе GET.

Таблица 1. Сводка свойств методов.

	GET	***QUERY***	***POST***
Безопасный	да	да	потенциально нет
Идемпотентный	да	да	потенциально нет
URI самого запроса	да (по определению)	возможно (поле отклика Location)	no
URI результата запроса	возможно (поле отклика Content-Location)	возможно (поле отклика Content-Location)	возможно (поле отклика Content-Location)
Кэширование	да	да	да, но только для будущих запросов GET или HEAD
Содержимое (тело)	нет заданной семантики	ожидается (семантика для целевого ресурса)	ожидается (семантика для целевого ресурса)

1.1. Терминология

В этом документе применяются термины, заданные в разделе 3 [HTTP]. Кроме того, применяются термины URI query parameter для параметров в компоненте query в URI (параграф 4.2.2 в [HTTP]) и query content для содержимого запроса (параграф 6.4 в [HTTP]).

1.2. Уровни требований

Ключевые слова **должно** (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.

2. Метод QUERY

Метод QUERY служит для инициирования запроса на стороне сервера. В отличие от метода GET, запрашивающего представления ресурса, указанного целевым URI (см. параграф 7.1 в [HTTP]), метод QUERY применяется для запроса у целевого ресурса выполнения операции в области действия этого целевого ресурса.

Содержимое запроса и тип носителя определяют запрос. Сервер-источник определяет область действия операции на основе целевого ресурса. Серверы **должны** отклонять запрос, если в нем отсутствует поле Content-Type (параграф 8.3 в [HTTP]) или это поле не соответствует содержимому запроса.

Как во всех методах HTTP, компонент query в целевом URI участвует в идентификации запрашиваемого ресурса. Непосредственное влияние этого компонента на результат запроса зависит от конкретного ресурса и не рассматривается в этой спецификации.

Запросы QUERY безопасны по отношению к целевому ресурсу (параграф 9.2.1 в [HTTP]), т. е. клиент не запрашивает и не ожидает какого-либо изменения в состоянии целевого ресурса. Это не мешает серверу создавать ресурсы HTTP, с помощью которых можно получить дополнительную информацию (см. параграфы 2.3 и 2.4). Кроме того, запросы QUERY идемпотентны (параграф 9.2.2 в [HTTP]), их можно повторять или передавать снова, например, при отказе соединения.

В соответствии с параграфом 15.3 в [HTTP], коды отклика 2xx (Successful) указывают, что запрос был получен, понят и воспринят. В частности, отклик 200 (OK) показывает, что запрос был успешно обработан и результаты этой обработки помещены в содержимое отклика.

2.1. Согласование типа носителя и содержимого

Семантика запросов QUERY зависит от содержимого запроса и связанных метаданных, таких как тип носителя ([HTTP], параграф 8.3.1). Как правило, при любой проблеме, связанной с несоответствием содержимого и метаданных, запрос **должен** отвергаться с кодом отклика 4xx (Client Error) (параграф 15.5 в [HTTP]). В приведённом ниже списке указаны различные варианты отказов и рекомендации по выбору кодов статуса.

- Запрос без указания типа носителя некорректен по определению и должен отвергаться с возвратом кода 4xx, такого как 400 (Client Error).
- Если указанный тип носителя не поддерживается ресурсом, подойдёт код 415 (Unsupported Media Type). Это относится, в частности, к случаю, когда тип носителя в принципе известен, но отсутствует семантика, относящаяся к запросу QUERY для целевого ресурса. В обоих случаях поле отклика Accept-Query (раздел 3) можно использовать для информирования клиентов о поддерживаемых типах носителей.
- Если тип носителя указан, но не соответствует фактическому содержимому запроса, можно возвращать код 400 (Bad Request). Т. е. серверу не разрешается определять тип носителя по содержимому запроса, а затем переопределять отсутствующее или «ошибочное» значение (например, «анализ содержимого»).
- Если тип носителя указан и понятен, содержимое соответствует типу, но запрос не может быть обработан из-за фактического содержимого, можно использовать код 422 (Unprocessable Content). Примером может служить синтаксически корректный запрос SQL к отсутствующей таблице.
- Если клиент запрашивает конкретный тип носителя для отклика в поле Accept (параграф 12.5.1 в [HTTP]), но этот тип не поддерживается ресурсом, подойдёт код 406 (Not Acceptable).

2.2. Эквивалентный ресурс

Для любого данного запроса QUERY эквивалентным является ресурс, который отвечает на запросы GET, представляет этот запрос QUERY и его цель, а также воспринимает и учитывает содержимое сообщения и метаданные (раздел 6 в [HTTP]). В частности, это включает метаданные представления (раздел 8 в [HTTP]), такие как тип носителя для содержимого. Иными словами, эквивалентный ресурс выводится из ресурса, реализующего QUERY, путём включения содержимого запроса.

Термин «эквивалентный ресурс» применяется как способ определения поведения для других аспектов HTTP, таких как выбранные представления. Серверы могут, но не обязаны назначать URI таким ресурсам (см. параграф 1.1 в [URI]). Если это делается, ресурс становится доступным для запросов GET.

2.3. Поле отклика Content-Location

Отклик об успехе (2xx, параграф 15.3 в [HTTP]) может включать поле заголовка Content-Location с идентификатором ресурса, соответствующего результатам операции (см. параграф 8.7 в [HTTP]). Это представляет утверждение сервера, что клиент может направить запрос GET по указанному URI для извлечения результатов только что выполненной операции. Указанный ресурс может быть временным. Пример представлен в Приложении A.4.1.

2.4. Поле отклика Location

Сервер может назначить URI эквивалентному ресурсу (параграф 2.2) запроса QUERY. Если сервер делает это, URI такого ресурса может включаться в поле заголовка Location в отклике 2xx (см. параграф 10.2.2 в [HTTP]). Это представляет утверждение сервера, что клиент может направить запрос GET по указанному URI для повторения операции из запроса без повторной отправки содержимого. URI ресурса может быть временным и при неудаче будущего запроса клиент может повторить попытку, используя цель исходного запроса QUERY и отправленное ранее содержимое. Пример представлен в Приложении A.4.2.

2.5. Перенаправление

В некоторых случаях сервер может опосредованно отвечать на запрос QUERY, перенаправляя агент пользователя на другой URI (см. параграф 15.4 в [HTTP]).

Отклики с кодами 301 (Moved Permanently, [HTTP], параграф 15.4.2) или 308 (Permanent Redirect, [HTTP], параграф 15.4.9) указывают, что целевой ресурс перемещён на постоянной основе в другой URI, указанный в поле отклика Location ([HTTP], параграф 10.2.2). Отклики с кодом 302 (Found, [HTTP], параграф 15.4.3) и 307 (Temporary Redirect, [HTTP], параграф 15.4.8) говорят о временном переносе целевого ресурса. Во всех четырёх случаях сервер считает, что агент пользователя может выполнить исходный запрос QUERY, передав аналогичное сообщение QUERY по новому URI цели, указанному в поле Location.

Отметим, что исключения для перенаправления POST как запроса GET после отклика 301 или 302 не применяются к запросам QUERY.

Отклик на QUERY с кодом 303 (See Other, параграф 15.4.4 в [HTTP]) указывает, что исходный запрос можно выполнить с помощью обычного запроса извлечения по URI из поля Location в отклике ([HTTP], параграф 10.2.2). Для HTTP это означает отправку запроса GET по новому целевому URI, как показано в примере Приложения A.4.3.

2.6. Запросы с условиями

Выбранное представление (параграф 3.2 в [HTTP]) для запроса QUERY совпадает с представлением для запроса GET к эквивалентному ресурсу (параграф 2.2). Запрос QUERY с условиями требует, чтобы выбранное представление (т. е. результаты запроса после всех согласований содержимого) возвращалось в отклике только при условиях, указанных в соответствующих полях заголовка, как указано в разделе 13 [HTTP]. Пример представлен в Приложении A.5.

2.7. Кэширование

Отклики для метода QUERY могут кэшироваться и кэш **может** использоваться для выполнения последующих запросов QUERY, как указано в разделе 4 [HTTP-CACHING].

Ключ кэша для запроса QUERY (раздел 2 в [HTTP-CACHING]) **должен** включать содержимое запроса (раздел 6 в [HTTP-CACHING]) и соответствующие метаданные (раздел 8 в [HTTP]).

Для повышения эффективности кэш **может** сначала исключать семантически незначимые отличия в содержимом запроса и связанных с ним метаданных, например:

- удаляя кодировку содержимого (параграф 8.4 в [HTTP]);
- выполняя нормализацию на основе знания соглашений о преобразованиях формата, на что указывает суффикс субтипа носителя в поле Content-Type (например, "+json", см. параграф 4.2.8 в [RFC6838]);
- выполняя нормализацию на основе знания семантики содержимого, указанной в поле запроса Content-Type.

Отметим, что любое из таких преобразований выполняется лишь для создания ключа кэша и не меняет самого запроса.

Клиенты могут указывать нежелательность таких преобразований с использованием директивы кэширования "no-transform" (параграф 5.2.1.6 в [HTTP-CACHING]), но это будет лишь рекомендацией.

Отметим, что кэширование откликов метода QUERY по своей природе сложнее кэширования откликов на запросы GET, поскольку для определения ключа кэша требуется полное прочтение содержимого. Если в отклике на QUERY имеется поле Location (параграф 2.4) для указания URI эквивалентного ресурса (параграф 2.2), клиенты могут в следующих запросах применять метод GET для упрощения обработки.

2.8. Запросы диапазона

Семантика Range Requests для QUERY идентична семантике для GET, заданной в разделе 14 [HTTP]. Однако запросы диапазона в байтах (единственный варниант, определённый на момент создания документа) не оказывают существенного влияния на результаты запросов QUERY. Форматы запросов часто задают свой способ ограничения или разбивки результатов на страницы, например, "FETCH FIRST ... ROWS ONLY" в SQL. Предполагается, что такие встроенные средства будут применяться вместо HTTP Range Requests.

3. Поле заголовка Accept-Query

Поле Accept-Query в заголовке отклика может применяться ресурсом для прямого указания поддержки метода QUERY, одновременно задавая конкретный формат запроса и типы носителя, которые могут использоваться. Accept-Query содержит список диапазонов носителей (параграф 12.5.1 в [HTTP]), используя синтаксис структурированных полей [STRUCTURED-FIELDS]. Диапазоны носителей представляются полем заголовка List Structured в виде маркеров или строк, содержащих диапазоны носителей без параметров.

При наличии параметров типа носителя они отображаются в Structured Field Parameters с типом String или Token. Выбор Token или String семантически малозначителен, т. е. получатель **может** преобразовать Token в String, но **недопустимо** обрабатывать их по-разному в зависимости от полученного типа. Типы носителей не совсем соответствуют маркерам (Token), например, они могут начинаться с цифр. В подобных случаях нужно применять формат String.

Поддерживаются лишь шаблоны */* (соответствует любому типу) и xxxx/* (соответствует любому подтипу xxxx).

Порядок типов в списке не имеет значения.

Значение поля Accept-Query применяется ко всем URI на сервере, имеющим тот же путь, иными словами, компонент query игнорируется. Если запросы к одному ресурсу возвращают разные значения Accept-Query, используется наиболее свежее из них (в соответствии с параграфом 4.2 в [HTTP-CACHING]).

Пример поля дан ниже.

```
Accept-Query: "application/jsonpath", application/sql;charset="UTF-8"
```

Хотя синтаксис этого поля похож на синтаксис других полей, таких как Accept (параграф 12.5.1 в [HTTP]), поле является структурированным и **должно** обрабатываться в соответствии с разделом 4 в [STRUCTURED-FIELDS].

4. Вопросы безопасности

Для метода QUERY применимы те же общие соображения безопасности, что и ко всем методам HTTP, описанным в [HTTP]. Метод можно применять как альтернативу передаче информации в URI (например, в компоненте query). В некоторых случаях это предпочтительно, поскольку URI с большей вероятностью, нежели содержимое запроса, записываются и обрабатываются посредниками. Если запрос собержит деликатные сведения, возможность записи URI в файлы журналов может служить мотивом применения QUERY вместо GET. Если сервер создаёт временный ресурс для представления результатов запроса QUERY (например, для использования в поле Location или Content-Location), назначая ему URI, а запрос содержит деликатную информацию, которую нельзя регистрировать, URI **следует** выбирать так, чтобы не включалась конфиденциальная часть содержимого исходного запроса.

Средства кэширования, некорректно выполняющие нормализацию QUERY или делающие это с существенными отличиями от обработки содержимого ресурсом, могут возвращать некорректный отклик, если нормализация ведёт к ложному срабатыванию.

Запрос QUERY от пользовательского агента, реализующего совместное использование ресурсов разных источников (Cross-Origin Resource Sharing или CORS), потребует предварительного запроса, поскольку QUERY не входит в число безопасных методов CORS (см. [FETCH]).

5. Взаимодействие с IANA

5.1. Регистрация метода QUERY

Агентство IANA добавило метод QUERY в реестр Hypertext Transfer Protocol (HTTP) Method <http://www.iana.org/assignments/http-methods> (см. параграф 16.3.1 в [HTTP]).

Таблица 2. Определение метода QUERY.

Имя метода	***Безопасный***	***Идемпотентный***	***Спецификация***
QUERY	да	да	раздел 2 в RFC 10008

5.2. Регистрация поля Accept-Query

Агентство IANA добавило поле Accept-Query в реестр Hypertext Transfer Protocol (HTTP) Field Name <https://www.iana.org/assignments/http-fields> (см. параграф 16.1.1 в [HTTP]).

Таблица 3. Определение поля Accept-Query.

Имя поля	***Статус***	***Структурированный тип***	***Документ***	***Комментарии***
Accept-Query	Постоянное	Список	Раздел 3 в RFC 10008	

6. Литература

6.1. 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>.

[HTTP-CACHING] Fielding, R., Ed., Nottingham, M., Ed., and J. Reschke, Ed., "HTTP Caching", STD 98, RFC 9111, DOI 10.17487/RFC9111, June 2022, <https://www.rfc-editor.org/info/rfc9111>.

[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>.

[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>.

[STRUCTURED-FIELDS] Nottingham, M. and P. Kamp, "Structured Field Values for HTTP", RFC 9651, DOI 10.17487/RFC9651, September 2024, <https://www.rfc-editor.org/info/rfc9651>.

[URI] Berners-Lee, T., Fielding, R., and L. Masinter, "Uniform Resource Identifier (URI): Generic Syntax", STD 66, RFC 3986, DOI 10.17487/RFC3986, January 2005, <https://www.rfc-editor.org/info/rfc3986>.

6.2. Дополнительная литература

[FETCH] WHATWG, "FETCH", WHATWG Living Standard, <https://fetch.spec.whatwg.org>. Commit snapshot: <https://fetch.spec.whatwg.org/commit-snapshots/3bab31a55154bda73f25b45a23df718616f2f64e/>.

[RFC3253] Clemm, G., Amsden, J., Ellison, T., Kaler, C., and J. Whitehead, "Versioning Extensions to WebDAV (Web Distributed Authoring and Versioning)", RFC 3253, DOI 10.17487/RFC3253, March 2002, <https://www.rfc-editor.org/info/rfc3253>.

[RFC4918] Dusseault, L., Ed., "HTTP Extensions for Web Distributed Authoring and Versioning (WebDAV)", RFC 4918, DOI 10.17487/RFC4918, June 2007, <https://www.rfc-editor.org/info/rfc4918>.

[RFC5323] Reschke, J., Ed., Reddy, S., Davis, J., and A. Babich, "Web Distributed Authoring and Versioning (WebDAV) SEARCH", RFC 5323, DOI 10.17487/RFC5323, November 2008, <https://www.rfc-editor.org/info/rfc5323>.

[RFC6838] Freed, N., Klensin, J., and T. Hansen, "Media Type Specifications and Registration Procedures", BCP 13, RFC 6838, DOI 10.17487/RFC6838, January 2013, <https://www.rfc-editor.org/info/rfc6838>.

[RFC8259] Bray, T., Ed., "The JavaScript Object Notation (JSON) Data Interchange Format", STD 90, RFC 8259, DOI 10.17487/RFC8259, December 2017, <https://www.rfc-editor.org/info/rfc8259>.

[RFC9535] Gössner, S., Ed., Normington, G., Ed., and C. Bormann, Ed., "JSONPath: Query Expressions for JSON", RFC 9535, DOI 10.17487/RFC9535, February 2024, <https://www.rfc-editor.org/info/rfc9535>.

[URL] WHATWG, "URL", WHATWG Living Standard, <https://url.spec.whatwg.org>. Commit snapshot: <https://url.spec.whatwg.org/commit-snapshots/52526653e848c5a56598c84aa4bc8ac9025fb66b/>.

[XSLT] Kay, M., Ed., "XSL Transformations (XSLT) Version 3.0", W3C Recommendation, 8 June 2017, <https://www.w3.org/TR/2017/REC-xslt-30-20170608/>. Latest version available at https://www.w3.org/TR/xslt-30/.

Приложение A. Примеры

Приведённые ниже примеры служат лишь иллюстрациями. Если кому-то реально нужны столь короткие запросы, лучше использовать GET.

В большинстве примеров применяется тип носителя application/x-www-form-urlencoded, как в запросах POST от браузеров, заданных application/x-www-form-urlencoded в [URL] (https://url.spec.whatwg.org/#application/x-www-form-urlencoded). Поля Content-Length для краткости опущены.

A.1. Простой запрос

Пример простого запроса с непосредственным откликом:

```
QUERY /contacts HTTP/1.1
Host: example.org
Content-Type: application/x-www-form-urlencoded
Accept: application/json

select=surname,givenname,email&limit=10&match=%22email=*@example.*%22
```

Отклик

```
HTTP/1.1 200 OK
Content-Type: application/json
[
  { "surname": "Smith",
    "givenname": "John",
    "email": "smith@example.org" },
  { "surname": "Jones",
    "givenname": "Sally",
    "email": "sally.jones@example.com" },
  { "surname": "Dubois",
    "givenname": "Camille",
    "email": "camille.dubois@example.net" }
]
```

A.2. Обнаружение поддержки QUERY

Простой способ обнаружения поддержки QUERY обеспечивает метод OPTIONS (параграф 9.3.7 в [HTTP]).

```
OPTIONS /contacts HTTP/1.1
Host: example.org
```

Отклик

```
HTTP/1.1 200 OK
Allow: GET, QUERY, OPTIONS, HEAD
```

Поле Allow в отклике (параграф 10.2.1 в [HTTP]) указывает набор поддерживаемых ресурсом методов.

Имеются и другие варианты. Например, запрос QUERY можно выполнить, не зная о поддержке метода сервером. В таком случае сервер обработает запрос или ответит с кодом статуса 4xx, таким как 405 (Method Not Allowed, параграф 15.5.6 в [HTTP]), включая в отклик поле Allow.

A.3. Обнаружение форматов QUERY

Поддерживаемые в методе QUERY типы носителей можно узнать из поля Accept-Query в отклике (раздел 3).

```
HEAD /contacts HTTP/1.1
Host: example.org
```

Отклик

```
HTTP/1.1 200 OK
Content-Type: application/xhtml
Accept-Query: application/x-www-form-urlencoded, application/sql
```

Отклики с Accept-Query зависят от ресурса, которому направлен запрос. Другим вариантом является отправка запроса QUERY, а затем - в случае получения кода 4xx, такого как 415 (Unsupported Media Type, параграф 15.5.16 в [HTTP]), - проверка поля Accept в отклике (параграф 12.5.1 в [HTTP]):

```
HTTP/1.1 415 Unsupported Media Type
Content-Type: application/xhtml
Accept: application/x-www-form-urlencoded, application/sql
```

A.4. Content-Location, Location и опосредованные отклики

Как указано в параграфах 2.3 и 2.4, поля Content-Location и Location в откликах об успехе (2xx, параграф 15.3 в [HTTP]) указывают способ идентификации дополнительных ресурсов, которые будут отвечать на запросы GET, для получения результатов запроса или будущих запросов на выполнение той же операции.

```
QUERY /contacts HTTP/1.1
Host: example.org
Content-Type: application/x-www-form-urlencoded
```

```
Accept: application/json

select=surname,givenname,email&limit=10&match=%22email=*@example.*%22
```

Отклик

```
HTTP/1.1 200 OK
Content-Type: application/json
Content-Location: /contacts/stored-results/17
Location: /contacts/stored-queries/42
Last-Modified: Sat, 25 Aug 2012 23:34:45 GMT
Date: Sun, 17 Nov 2024, 16:10:24 GMT

[
  { "surname": "Smith",
    "givenname": "John",
    "email": "smith@example.org" },
  { "surname": "Jones",
    "givenname": "Sally",
    "email": "sally.jones@example.com" },
  { "surname": "Dubois",
    "givenname": "Camille",
    "email": "camille.dubois@example.net" }
]
```

A.4.1. Использование Content-Location

Поле Content-Location в отклике на запрос QUERY указывает ресурс, содержащий результат для этого запроса.

```
GET /contacts/stored-results/17 HTTP/1.1
Host: example.org
Accept: application/json
```

Отклик

```
HTTP/1.1 200 OK
Last-Modified: Sat, 25 Aug 2012 23:34:45 GMT
Date: Sun, 17 Nov 2024, 16:10:25 GMT

[
  { "surname": "Smith",
    "givenname": "John",
    "email": "smith@example.org" },
  { "surname": "Jones",
    "givenname": "Sally",
    "email": "sally.jones@example.com" },
  { "surname": "Dubois",
    "givenname": "Camille",
    "email": "camille.dubois@example.net" }
]
```

Следует отметить, что не гарантируется постоянное сохранение сервером этого ресурса, поэтому после получения отклика с ошибкой клиенту потребуется повторить исходный запрос QUERY для определения нового местоположения.

A.4.2. Использование Location

Поле Location в отклике указывает ресурс, который будет отвечать на запрос GET текущим результатом для того же процесса и параметров, которые были указаны в исходном запросе QUERY.

```
GET /contacts/stored-queries/42 HTTP/1.1
Host: example.org
Accept: application/json
```

В этом примере одна запись была удалена в момент 2024-11-17T16:12:01Z (указан в поле Last-Modified), поэтому отклик содержит лишь две записи.

```
HTTP/1.1 200 OK
Content-Type: application/json
Last-Modified: Sun, 17 November 2024, 16:12:01 GMT
ETag: "42-1"
Date: Sun, 17 Nov 2024, 16:13:17 GMT

[
  { "surname": "Smith",
    "givenname": "John",
    "email": "smith@example.org" },
  { "surname": "Dubois",
    "givenname": "Camille",
    "email": "camille.dubois@example.net" }
]
```

В предположении, что сервер по-прежнему предоставляет ресурс и результат запроса не изменился, последующий запрос GET с условием

```
If-None-Match: "42-1"
```

возвратит код 304 (Not Modified, параграф 15.4.5 в [HTTP]).

A.4.3. Опосредованные отклики

Серверы могут передавать «опосредованные» отклики (параграф 2.5), используя код 303 (See Other, параграф 15.4.4 в [HTTP]). На запрос в начале параграфа A.4 сервер может ответить:

```
HTTP/1.1 303 See Other
```

```
Content-Type: text/plain
Date: Sun, 17 Nov 2024, 16:13:17 GMT
Location: /contacts/stored-queries/42
```

Сохранённый ресурс находится в /contacts/stored-queries/42. Это похоже на включение Location в прямой отклик, но результат в этом случае не возвращается. Это позволяет серверу лишь создать дополнительный ресурс или повторно использовать ранее созданный. Ресурс можно использовать, как показано в параграфе A.4.2.

A.5. Запросы с условием

Рассмотрим ресурс, реализующий QUERY с поддержкой application/sql и application/xslt+xml [XSLT] в качестве типов носителя для запроса и способный создавать отклики text/csv. Запрашивается набор данных, содержащий сведения об RFC, и возвращается информация, сгруппированная по десятилетиям.

```
QUERY /rfc-index.xml HTTP/1.1
Host: example.org
Date: Sun, 7 Sep 2025, 00:00:00 GMT
Content-Type: application/xslt+xml
Accept: text/csv

...Содержимое запроса в XSLT...
```

Отклик

```
HTTP/1.1 200 OK
Date: Sun, 7 Sep 2025, 00:00:00 GMT
Location: /stored-queries/4815162342
Content-Type: text/csv
Accept-Query: "application/sql", "application/xslt+xml"
Last-Modified: Sun, 31 Aug 2025, 08:44:00 GMT
Vary: Accept-Query, Content-Encoding, Content-Type

decade, total, with errata, % with errata, average page count
1960, 26, 5, 19.2, 5.3
1970, 666, 18, 2.7, 6.1
1980, 376, 44, 11.7, 23.4
1990, 1593, 269, 16.9, 25.5
2000, 2888, 1048, 36.3, 27.3
2010, 2954, 895, 30.3, 26.1
2020, 1133, 230, 20.3, 26.2
```

Сервер указал путь /stored-queries/4815162342 к эквивалентному ресурсу (параграф 2.4) для последующего запроса GET. Клиент повторяет запрос, указывая, что результаты следует возвращать лишь при их изменении

```
QUERY /rfc-index.xml HTTP/1.1
Host: example.org
Date: Mon, 8, Sep 2025, 11:00:00 GMT
Content-Type: application/sql
Accept: text/csv
If-Modified-Since: Sun, 31 Aug 2025, 08:44:00 GMT
Vary: Accept-Query, Content-Type

...Тот же запрос в SQL...
```

Запрошенные данные не изменились, поэтому сервер возвращает:

```
HTTP/1.1 304 Not Modified
Date: Mon, 8 Sep 2025, 11:00:00 GMT
Content-Type: text/csv
Location: /stored-queries/4815162342
Accept-Query: "application/sql", "application/xslt+xml"
Last-Modified: Sun, 31 Aug 2025, 08:44:00 GMT
Vary: Accept-Query, Content-Type
```

Поскольку сервер указал URI эквивалентного ресурса, к этому ресурсу можно обратиться с помощью GET. Это, в частности, избавляет от повторной передачи содержимого запроса.

```
GET /stored-queries/4815162342 HTTP/1.1
Host: example.org
Date: Sun, 21, Sep 2025, 12:08:00 GMT
Accept: text/csv
If-Modified-Since: Sun, 31 Aug 2025, 00:00:00 GMT
```

В этом случае состояние набора данных поменялось и возвращается новое содержимое:

```
HTTP/1.1 200 OK
Date: Sun, 21, Sep 2025, 12:08:00 GMT
Content-Type: text/csv
Last-Modified: Thu, 18 Sep 2025, 19:56:00 GMT
Vary: Accept-Query, Content-Encoding, Content-Type

decade, total, with errata, % with errata, average page count
1960, 26, 5, 19.2, 5.3
1970, 666, 18, 2.7, 6.1
1980, 376, 44, 11.7, 23.4
1990, 1593, 269, 16.9, 25.5
2000, 2888, 1048, 36.3, 27.3
2010, 2954, 895, 30.3, 26.1
2020, 1133, 230, 20.3, 26.2
```

(обратите внимание на изменение строки для этого десятилетия[1]).

[1] По видимому в оригинале допущена ошибка, т. к. наборы строк в обоих примерах не отличаются. *Прим. перев.*

На рисунках ниже показано применение запросов с условием и из возможные различия, когда назначается URI эквивалентного ресурса и клиент использует его в своих интересах. Для демонстрации применяется вымышленное имя Validator.

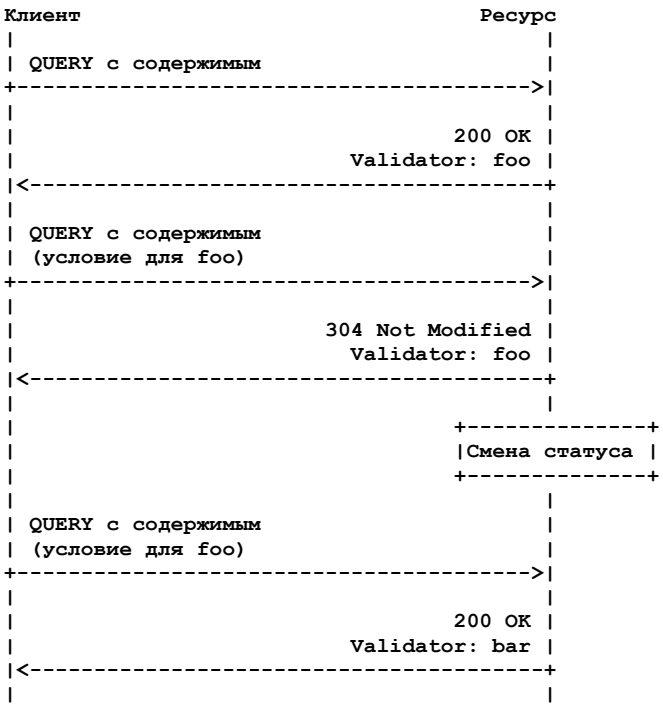

Рисунок 1. Поток данных только для QUERY.

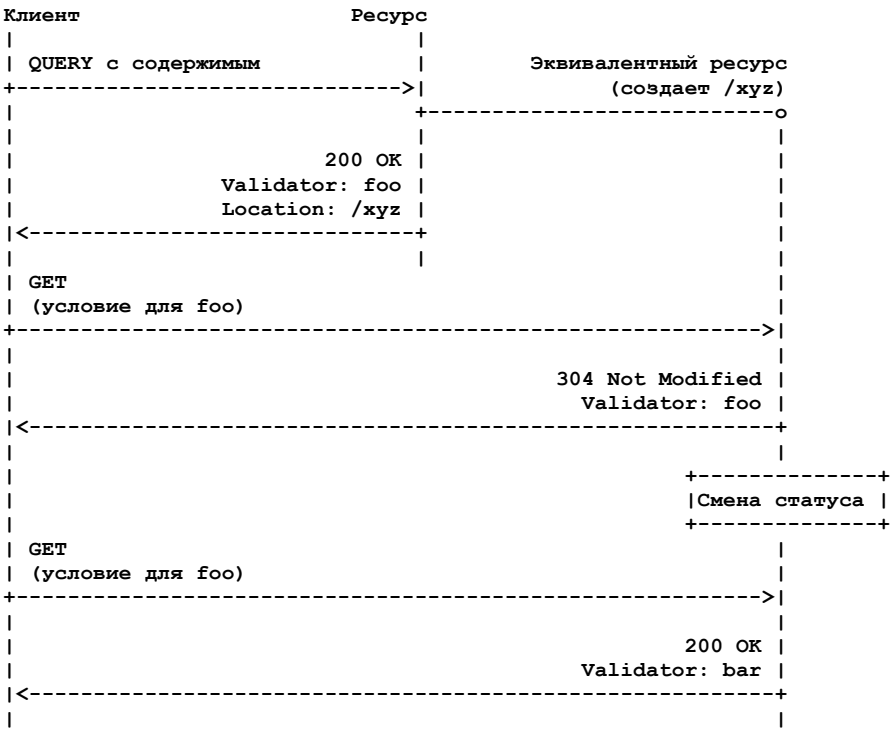

Рисунок 2. Поток данных с GET к эквивалентному ресурсу.

A.6. Дополнительные форматы Query

Ниже показаны запросы к базе данных RFC errata (ошибки) в формате JSON [RFC8259]. В приведённом запросе применяется формат XSLT (eXtensible Stylesheet Language Transformations) для извлечения сводных данных об ошибках по годам и определённых типах ошибок.

```
QUERY /errata.json HTTP/1.1
Host: example.org
Content-Type: application/xslt+xml
Accept: application/xml, text/csv

<transform xmlns="http://www.w3.org/1999/XSL/Transform"
  xmlns:j="http://www.w3.org/2005/xpath-functions"
  version="3.0">

  <output method="text"/>

  <param name="input"/>
```

```
    <variable name="json"
      select="json-to-xml(unparsed-text($input))"/>

    <variable name="sc">errata_status_code</variable>
    <variable name="sd">submit_date</variable>

    <template match="/">
      <text>year, total, rejected, verified, hdu, reported</text>
      <text>&#10;</text>
      <variable name="en" select="$json//j:map"/>
      <for-each-group select="$en"
        group-by="substring-before(j:string[@key=$sd],'-')">
        <sort select="current-grouping-key()"/>
        <variable name="year" select="current-grouping-key()"/>
        <variable name="errata" select=
          "$en[$year=substring-before(j:string[@key=$sd],'-')]"/>
        <value-of select="concat(
          $year,
          ', ',
          count($errata),
          ', ',
          count($errata['Rejected'=j:string[@key=$sc]]),
          ', ',
          count($errata['Verified'=j:string[@key=$sc]]),
          ', ',
          count(
            $errata['Held for Document Update'=j:string[@key=$sc]]),
          ', ',
          count($errata['Reported'=j:string[@key=$sc]]),
          '&#10;')"/>
      </for-each-group>
    </template>

  </transform>
```

Отклик

```
  HTTP/1.1 200 OK
  Content-Type: text/csv
  Accept-Query: "application/jsonpath", "application/xslt+xml"
  Date: Wed, 19 Feb 2025, 17:10:01 GMT

  year, total, rejected, verified, hdu, reported
  2000, 14, 0, 14, 0, 0
  2001, 72, 1, 70, 1, 0
  2002, 124, 8, 104, 12, 0
  2003, 63, 0, 61, 2, 0
  2004, 89, 1, 83, 5, 0
  2005, 156, 10, 96, 50, 0
  2006, 444, 54, 176, 214, 0
  2007, 429, 48, 188, 193, 0
  2008, 423, 52, 165, 206, 0
  2009, 331, 39, 148, 144, 0
  2010, 538, 80, 232, 222, 4
  2011, 367, 47, 170, 150, 0
  2012, 348, 54, 149, 145, 0
  2013, 341, 61, 169, 106, 5
  2014, 342, 73, 180, 72, 17
  2015, 343, 79, 145, 89, 30
  2016, 295, 46, 122, 82, 45
  2017, 303, 46, 120, 84, 53
  2018, 350, 61, 118, 98, 73
  2019, 335, 47, 131, 94, 63
  2020, 387, 68, 117, 123, 79
  2021, 321, 44, 148, 63, 66
  2022, 358, 37, 198, 40, 83
  2023, 262, 38, 121, 33, 70
  2024, 322, 33, 125, 23, 141
  9999, 1, 0, 0, 1, 0
```

Отметим, что поле отклика Accept-Query указывает поддержку и другого формата запросов - JSONPath [RFC9535]. Приведённый ниже запрос возвращает все отклонённые сообщения об ошибках с 2024 года.

```
  QUERY /errata.json HTTP/1.1
  Host: example.org
  Content-Type: application/jsonpath
  Accept: application/json

  $..[
       ?@.errata_status_code=="Rejected"
       && @.submit_date>"2024"
     ]
     ["doc-id"]
```

Отклик

```
  HTTP/1.1 200 OK
  Content-Type: application/json
  Accept-Query: "application/jsonpath", "application/xslt+xml"
```

```
   Date: Thu, 20 Feb 2025, 09:55:42 GMT
   Last-Modified: Thu, 20 Feb 2025 06:10:01 GMT

   [
     "RFC1185","RFC8407","RFC6350","RFC8467","RFC1157","RFC9543",
     "RFC9076","RFC7656","RFC2822","RFC9460","RFC2104","RFC6797",
     "RFC9499","RFC9557","RFC2131","RFC2328","RFC9001","RFC3325",
     "RFC9438","RFC2526","RFC2985","RFC7643","RFC9132","RFC6376",
     "RFC9110","RFC9460","RFC7748","RFC9497","RFC8463","RFC4035",
     "RFC7239","RFC9083","RFC9537","RFC9537","RFC9420","RFC9000",
     "RFC9656","RFC9110","RFC2324","RFC2549","RFC6797","RFC2549",
     "RFC8894"
   ]
```

Приложение B. Выбор имени метода QUERY

В реестре Hypertext Transfer Protocol (HTTP) Method Registry (<http://www.iana.org/assignments/http-methods>) уже имеются три других метода со свойствами безопасности (safe) и идемпотентности (idempotent): PROPFIND [RFC4918], REPORT [RFC3253] и SEARCH [RFC5323]. Можно было бы использовать любой из них, обновив в соответствии с тем, что в данной спецификации определено как новый метод QUERY. И действительно, на ранних этапах этой спецификации использовался метод SEARCH.

Название метода QUERY в конечном счёте было выбрано по нескольким причинам:

- в других вариантах применяется базовый тип носителя для содержимого запроса (application/xml), а семантика запроса зависит лишь от его содержимого;
- все отмеченные выше методы основаны на действиях WebDAV, по поводу чего у многих возникают смешанные чувства;
- имя QUERY чётко отражает связь с компонентом URI query.

Благодарности

Спасибо всем членам рабочей группы HTTP за их идеи, рецензии и отклики. Отдельной благодарности заслуживают Carsten Bormann, Mark Nottingham, Martin Thomson, Michael Thornburgh, Roberto Polli, Roy Fielding и Will Hawkins.

Участники работы

Ashok Malhotra участвовал в ранних обсуждениях, приведших к этой спецификации.

Ashok Malhotra
Email: malhotrasahib@gmail.com

Обсуждение этого метода HTTP было возобновлено Asbjørn Ulsberg на HTTP Workshop в 2019 году.

Asbjørn Ulsberg
Email: asbjorn@ulsberg.no
URI: https://asbjor.nu/

Адреса авторов

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

James M Snell
Cloudflare
Email: jasnell@gmail.com

Mike Bishop
Akamai
Email: mbishop@evequefou.be

Перевод на русский язык

Николай Малых

nmalykh@protokols.ru