**HTTP and HTTPS**

HTTP uses TCP

Протокол HTTPs отличается от HTTP тем что данные до их отправки шифруются и протокол HTTPs использует сертификаты. В остальном это тот же HTTP протокол.

Какие протоколы шифрования вы можете назвать. Их последние версии?

**SSL** (latest 3.0)([англ.](http://ru.wikipedia.org/wiki/%D0%90%D0%BD%D0%B3%D0%BB%D0%B8%D0%B9%D1%81%D0%BA%D0%B8%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA" \o "Английский язык) Secure Sockets Layer — уровень защищённых сокетов) — [криптографический протокол](http://ru.wikipedia.org/wiki/%D0%9A%D1%80%D0%B8%D0%BF%D1%82%D0%BE%D0%B3%D1%80%D0%B0%D1%84%D0%B8%D1%87%D0%B5%D1%81%D0%BA%D0%B8%D0%B9_%D0%BF%D1%80%D0%BE%D1%82%D0%BE%D0%BA%D0%BE%D0%BB" \o "Криптографический протокол), который обеспечивает безопасность связи

**TLS** (latest 1.3)([англ.](http://ru.wikipedia.org/wiki/%D0%90%D0%BD%D0%B3%D0%BB%D0%B8%D0%B9%D1%81%D0%BA%D0%B8%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA" \o "Английский язык) Transport Layer Security — безопасность транспортного уровня, как и его предшественник **SSL** ([англ.](http://ru.wikipedia.org/wiki/%D0%90%D0%BD%D0%B3%D0%BB%D0%B8%D0%B9%D1%81%D0%BA%D0%B8%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA) Secure Socket Layers — уровень защищённых сокетов) — [криптографические](http://ru.wikipedia.org/wiki/%D0%9A%D1%80%D0%B8%D0%BF%D1%82%D0%BE%D0%B3%D1%80%D0%B0%D1%84%D0%B8%D1%8F) протоколы, обеспечивающие защищённую передачу данных между узлами в сети [Интернет](http://ru.wikipedia.org/wiki/%D0%98%D0%BD%D1%82%D0%B5%D1%80%D0%BD%D0%B5%D1%82)[[1]](http://ru.wikipedia.org/wiki/TLS#cite_note-1).

Какой порт по умолчанию для HTTP и HTTPS? (80) (443)

****

**Request Headers -** используются только в запросах клиента.

**Cache**

* **If-Modified-Since** – Используется вместе с Last-modified. The If-Modified-Since request HTTP header makes the request conditional: the server sends back the requested resource, with a 200 status, only if it has been last modified after the given date. If the resource has not been modified since, the response is a 304 without any body; the Last-Modified response header of a previous request contains the date of last modification. Unlike If-Unmodified-Since, If-Modified-Since can only be used with a GET or HEAD.
* **If-Unmodified-Since**
* **If-None-Match**  - Используется вместе в ETag, что бы проверить обновлялся ли ресурс или нет (что бы загружать его по новой или использовать из кеша)

**Cookie**

* **Cookie** - The Cookie HTTP request header contains stored HTTP cookies associated with the server (i.e. previously sent by the server with the Set-Cookie header or set in Javascript using Document.cookie).
* **Authorization** - The HTTP Authorization request header can be used to provide credentials that authenticate a user agent with a server, allowing access to a protected resource.

**Referrer**

* **Referrer** - HTTP request header contains an absolute or partial address of the page that makes the request. The Referer header allows a server to identify a page where people are visiting it from. This data can be used for analytics, logging, optimized caching, and more.

**Client**

* **User-agent** - The User-Agent request header is a characteristic string that lets servers and network peers identify the application, operating system, vendor, and/or version of the requesting user agent.
* **Accept** - The Accept request HTTP header indicates which content types, expressed as MIME types, the client is able to understand
* **Accept-Language** - The Accept-Language request HTTP header indicates the natural language and locale that the client prefers

**Response Headers -** включаются только в ответы сервера.

**Cache**

* **Expires** - указывает до какого времени нужно хранить ресурс в кеше
* **Last-modified** - The Last-Modified response HTTP header contains a date and time when the origin server believes the resource was last modified. It is used as a validator to determine if the resource is the same as the previously stored one
* **ETag** - lets caches be more efficient and save bandwidth, as a web server does not need to resend a full response if the content was not changed. If the resource at a given URL changes, a new Etag value must be generated. A comparison of them can determine whether two representations of a resource are the same
* **Cache-control** - указывает до какого времени нужно хранить ресурс в кеше

**Cookie**

* **Set-Cookie** - The Set-Cookie HTTP response header is used to send a cookie from the server to the user agent, so that the user agent can send it back to the server later. To send multiple cookies, multiple Set-Cookie headers should be sent in the same response.

**Redirect**

* **Location** - The Location response header indicates the URL to redirect a page to. It only provides a meaning when served with a 3xx (redirection) or 201 (created) status response. 303, 307, 308, 301, 302

**Other**

* **Server** - The Server header describes the software used by the origin server that handled the request — that is, the server that generated the response.

**Request and Response Headers** – Включаются и в ответы и запросы

**Referrer**

* **Referrer Policy -** The Referrer-Policy HTTP header controls how much referrer information (sent with the Referer header) should be included with requests

**Other**

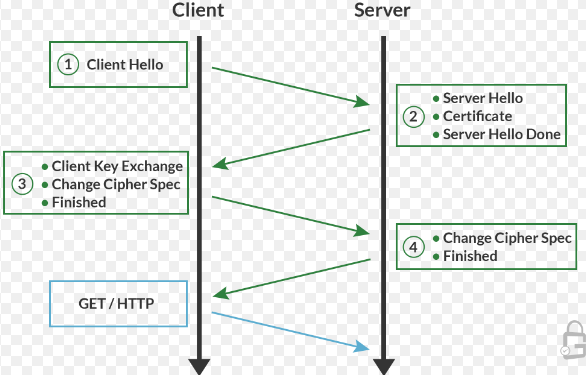
* **Content-Length** – определяет длину тела запроса в октетах. Это значение не имеет особой важности, но некоторые HTTP-методы (например PUT) требуют этот заголовок
* **Content-Type** – Сообщает какой будет тип передаваемого контента. Сервер сообщает клиенту, или клиент серверу, во время POST, PUT запросов
* **Accept-Encoding** - The Accept-Encoding request HTTP header indicates the content encoding (usually a compression algorithm) that the client can understand. The server uses content negotiation to select one of the proposal and informs the client of that choice with the Content-Encoding response header.
* **Connection** - The Connection general header controls whether the network connection stays open after the current transaction finishes. If the value sent is keep-alive, the connection is persistent and not closed, allowing for subsequent requests to the same server to be done.
* **Date** - The Date general HTTP header contains the date and time at which the message was originated.
* **Upgrade** - The HTTP 1.1 (only) Upgrade header can be used to upgrade an already established client/server connection to a different protocol (over the same transport protocol). For example, it can be used by a client to upgrade a connection from HTTP 1.1 to HTTP 2.0, or an HTTP or HTTPS connection into a WebSocket.

**Keep-alive**

Время ответа на запрос складывается из 2 вещей

1. Сетевые задержки во время запроса от клиента к серверу и обратно
2. Обработка запроса на сервере (выполнение кода – бизнес логики, запросы к БД)

Во время запроса от клиента к серверу для HTTPs протокола, при каждом запросе выполняется TLS handshake для обмена ключами и проверки сертификатов (открывается соединение). После того как запрос завершился, соединение закрывается. Из за этого время выполнения каждого запроса увеличивается на время этого handshake



Что бы решить эту проблему есть заколовок **connection: keep-alive**. При нем мы 1 раз делаем handshake, а при следующих запросах мы уже handshake не делаем – соединение остается открытым.

**Методы (get, post, ...)**

**GET**

* предназначен для получения данных с сервера;
* тело запроса пустое;
* обрабатываются на стороне сервера быстрее и с меньшим потреблением ресурсов сервера за счет пустого тела запроса;
* передача переменных происходит в адресной строке (так видит пользователь, технически данные передаются в строке запроса) и поэтому видна информация о переменных и их значениях (данные не защищены);
* способен передать небольшое количество данных на сервер: есть ограничения на длину URL, которое зависит от браузера, например, IE6 = 2Kb. На это число и рекомендуют ориентироваться разработчики Yahoo!;
* может передать только ASCII символы;
* такой запрос можно скопировать, сохранить (например, в закладках);
* запрос может кэшироваться (этим можно управлять);
* для дополнительного снижения нагрузки на канал и сервер доступны условные и частичные запросы;
* не разрывает HTTP соединение (при включенном на сервере режиме keepAlive).

**POST**

* предназначен для отправки данных на сервер;
* передача данных происходит в теле запроса;
* обработка на стороне сервера медленнее и «тяжелее», чем GET, потому что помимо заголовков нужно анализировать тело запроса;
* способен передать большие объемы данных;
* способен передать файлы;
* страницу, сгенерированную методом POST нельзя сохранить в закладки;
* разрывает HTTP соединение;
* для передачи даже очень малого объема информации большинством браузеров отправляет минимум два TCP пакета: заголовок, а потом тело запроса.

****

**PUT** - заменяет все текущие представления ресурса данными запроса.

**PATCH** - используется для частичного изменения ресурса.

**DELETE** - удаляет указанный ресурс.

**OPTIONS** – позволяет узнать какие методы поддерживаются для запрашиваемого ресурса

**HEAD** - запрашивает ресурс так же, как и метод GET, но без тела ответа.

**CONNECT** - устанавливает "туннель" к серверу, определённому по ресурсу.

**TRACE** – позволяет отслеживать кто вносит изменения на странице

Some calls to APIs can change things on the server, while others return data without changing anything. It is important to distinguish between these different types of calls when testing or designing an API.

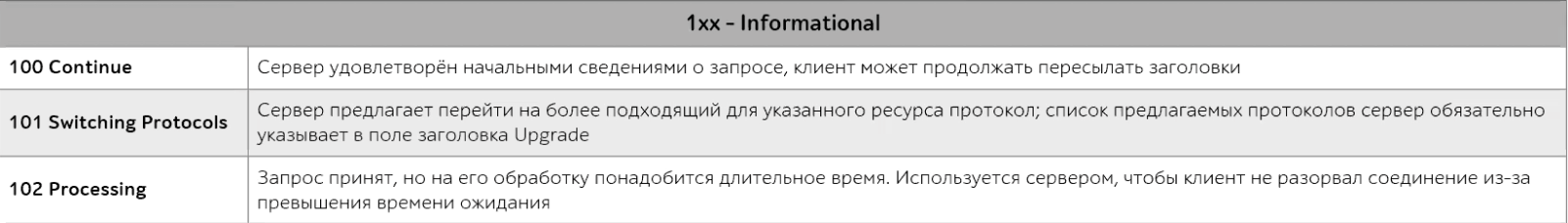
1. Метод HTTP является **безопасным** (**safe**), если он не меняет состояние сервера. Другими словами, безопасный метод проводит операции "только чтение" (read-only). **GET**
2. **Не безопасный** тот который меняет состояние на сервере. Например загрузка файла на сервер
3. **Идемпотентный** - если повторный идентичный запрос, сделанный один или несколько раз подряд, имеет один и тот же эффект, не изменяющий состояние сервера*.* Например после удаления файла с сервера, повторное удаление этого файла не поменяет его состояние, так как файл уже был удален. Если мы редактируем запись, и вводим те же данные что и в первый раз, то состояние сервера не поменяется.

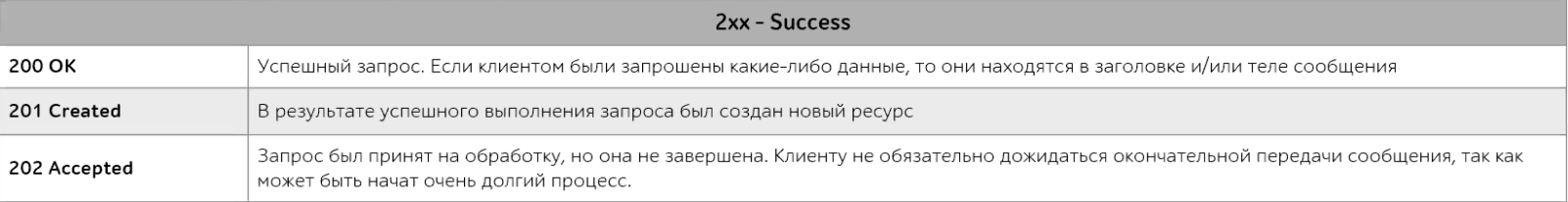
Другими словами, идемпотентный метод не должен иметь никаких побочных эффектов (side-effects), кроме сбора статистики или подобных операций. Корректно реализованные методы GET, PATCH, PUT и DELETE идемпотентны, но не метод POST (например при создании 1 и той же сущности у нее могут быть разные id).

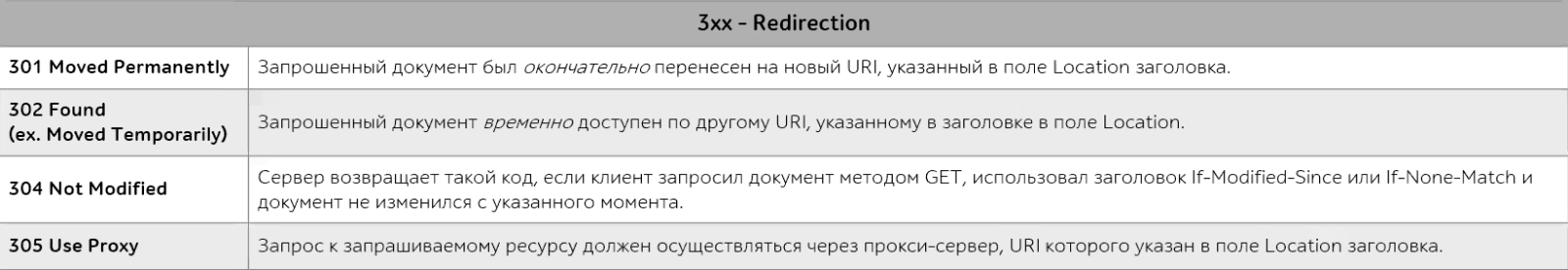


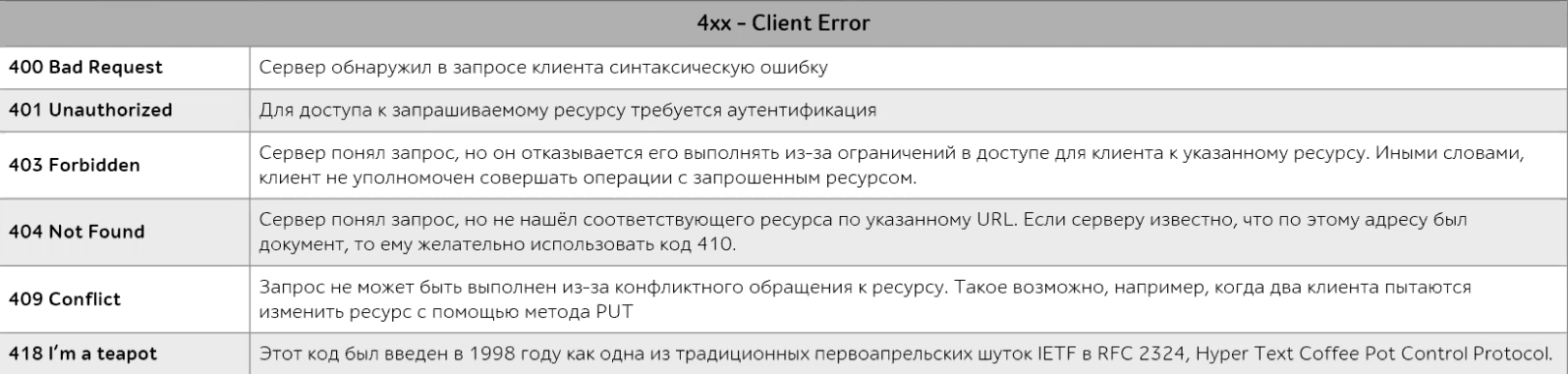
**СТАТУС КОДЫ**

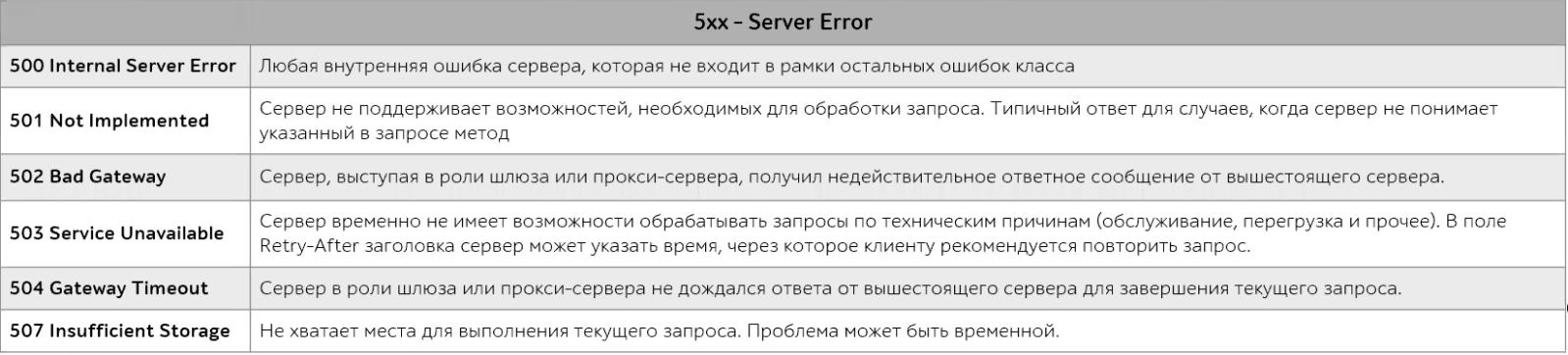
****

****

****

****

****

****

**(cancelled)** - That's server push cancelling the long polling request to either refresh it (because it might be timed out by intermediate proxies) or to signal a client-side event.

There's a rule of thumb that you should have a new request after about 50s because a proxy might silently drop your long polling connection.