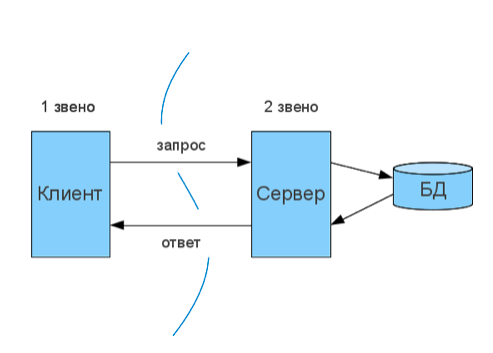
***1) Прочиать про клиент-серверную архитектуру***

**Веб-приложение**– это клиент-серверное приложение, в котором *клиентом* выступает браузер, а *сервером* – веб-сервер (в широком смысле).

Основная часть приложения, как правило, находится на стороне веб-сервера, который обрабатывает полученные запросы в соответствии с бизнес-логикой продукта и формирует ответ, отправляемый пользователю. На этом этапе в работу включается браузер, именно он преобразовывает полученный ответ от сервера в графический интерфейс, понятный пользователю.

**Архитектура «клиент-сервер»**определяет общие принципы организации взаимодействия в сети, где имеются серверы, узлы-поставщики некоторых специфичных функций (сервисов) и клиенты (потребители этих функций).  
Практические реализации такой архитектуры называются **клиент-серверными технологиями.**

**Двухзвенная архитектура**- распределение трех базовых компонентов между двумя узлами (клиентом и сервером). Двухзвенная архитектура используется в клиент-серверных системах, где сервер отвечает на клиентские запросы напрямую и в полном объеме.



Расположение компонентов на стороне клиента или сервера определяет следующие основные модели их взаимодействия в рамках двухзвенной архитектуры:

* **Сервер терминалов**— распределенное представление данных.
* **Файл-сервер**— доступ к удаленной базе данных и файловым ресурсам.
* **Сервер БД**— удаленное представление данных.
* **Сервер приложений**— удаленное приложение.

**Клиент** – это браузер, но встречаются и исключения (в тех случаях, когда один веб-сервер (ВС1) выполняет запрос к другому (ВС2), роль клиента играет веб-сервер ВС1). В классической ситуации (когда роль клиента выполняет браузер) для того, чтобы пользователь увидел графический интерфейс приложения в окне браузера, последний должен обработать полученный ответ веб-сервера, в котором будет содержаться информация, реализованная с применением HTML, CSS, JS (самые используемые технологии). Именно эти технологии «дают понять» браузеру, как именно необходимо «отрисовать» все, что он получил в ответе.

**Веб-сервер**– это сервер, принимающий HTTP-запросы от клиентов и выдающий им HTTP-ответы. Веб-сервером называют как программное обеспечение, выполняющее функции веб-сервера, так и непосредственно компьютер, на котором это программное обеспечение работает. Наиболее распространенными видами ПО веб-серверов являются Apache, IIS и NGINX. На веб-сервере функционирует тестируемое приложение, которое может быть реализовано с применением самых разнообразных языков программирования: PHP, Python, Ruby, Java, Perl и пр.

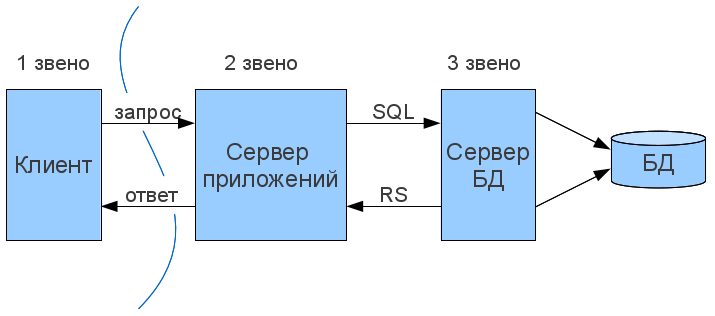
**База данных**фактически не является частью веб-сервера, но большинство приложений просто не могут выполнять все возложенные на них функции без нее, так как именно в базе данных хранится вся динамическая информация приложения (учетные, пользовательские данные и пр).

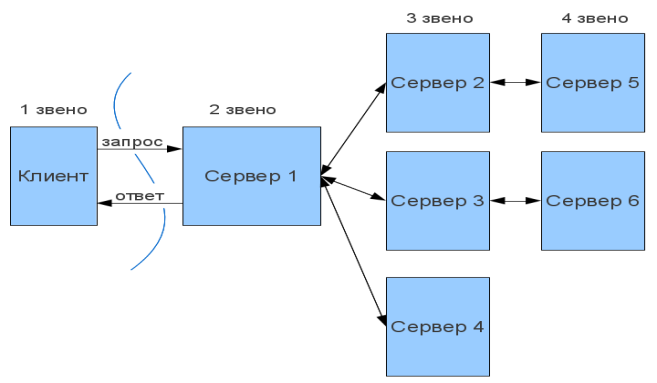
**База данных**- это информационная модель, позволяющая упорядоченно хранить данные об объекте или группе объектов, обладающих набором свойств, которые можно категоризировать. Базы данных функционируют под управлением так называемых систем управления базами данных (далее – СУБД). Самыми популярными СУБД являются MySQL, MS SQL Server, PostgreSQL, Oracle (все – клиент-серверные).

**Трехзвенная архитектура -**сетевое приложение разделено на две и более частей, каждая из которых может выполняться на отдельном компьютере. Выделенные части приложения взаимодействуют друг с другом, обмениваясь сообщениями в заранее согласованном формате.

Третьим звеном в трехзвенной архитектуре становится *сервер приложений*, т.е. компоненты распределяются следующим образом:

1. Представление данных — на стороне клиента.
2. Прикладной компонент — на выделенном сервере приложений (как вариант, выполняющем функции промежуточного ПО).
3. Управление ресурсами — на сервере БД, который и представляет запрашиваемые данные.

  
Трехзвенная архитектура может быть расширена до **многозвенной (N-tier, Multi-tier)**путем выделения дополнительных серверов, каждый из которых будет представлять собственные сервисы и пользоваться услугами прочих серверов разного уровня.



Двухзвенная архитектура проще, так как все запросы обслуживаются одним сервером, но именно из-за этого она менее надежна и предъявляет повышенные требования к производительности сервера.

Трехзвенная архитектура сложнее, но, благодаря тому, что функции распределены между серверами второго и третьего уровня, эта архитектура предоставляет:

1. Высокую степень гибкости и масштабируемости.
2. Высокую безопасность (т.к. защиту можно определить для каждого сервиса или уровня).
3. Высокую производительность (т.к. задачи распределены между серверами).

## ****Клиент-серверные технологии****

Архитектура клиент-сервер применяется в большом числе сетевых технологий, используемых для доступа к различным сетевым сервисам.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Типы сервисов** | | | | | | | |
| * **Web-серверы** | **Серверы приложений** | * **Серверы баз данных** | * **Файл-серверы** | **Прокси-сервер** | **Файрволы**  (брандмауэры) | * **Почтовые серверы** | **Серверы удаленного доступа (RAS)** |
| Изначально предоставляли доступ к гипертекстовым документам по протоколу HTTP (Hyper Text Transfer Protocol). Сейчас поддерживают расширенные возможности, в частности, работу с бинарными файлами (изображения, мультимедиа и т.п.). | Предназначены для централизованного решения прикладных задач в некоторой предметной области. Для этого пользователи имеют право запускать серверные программы на исполнение. Использование серверов приложений позволяет снизить требования к конфигурации клиентов и упрощает общее управление сетью. | Серверы баз данных используются для обработки пользовательских запросов на языке SQL. При этом, СУБД находится на сервере, к которому и подключаются клиентские приложения. | Файл-сервер хранит информацию в виде файлов и предоставляет пользователям доступ к ней. Как правило, файл-сервер обеспечивает и определенный уровень защиты от несанкционированного доступа | Во-первых, действует как посредник, помогая пользователям получить информацию из Интернета и, при этом, обеспечивая защиту сети.  Во-вторых, сохраняет часто запрашиваемую информацию в кэш-памяти на локальном диске, быстро доставляя ее пользователям, без повторного обращения к Интернету. | Межсетевые экраны, анализирующие и фильтрующие проходящий сетевой трафик, с целью обеспечения безопасности сети. | Предоставляют услуги по отправке и получению электронных почтовых сообщений. | Эти системы обеспечивают связь с сетью по коммутируемым линиям. Удаленный сотрудник может использовать ресурсы корпоративной ЛВС, подключившись к ней с помощью обычного модема. |

Для доступа к тем или иным сетевым сервисам используются клиенты, возможности которых характеризуются **понятием «толщины».** Оно определяет конфигурацию оборудования и программное обеспечение, имеющиеся у клиента. Рассмотрим возможные граничные значения:

|  |  |  |
| --- | --- | --- |
| **«Тонкий» клиент** | **«Толстый» клиент** | **«Rich»-client** |
| Этот термин определяет клиента, вычислительных ресурсов которого достаточно лишь для запуска необходимого сетевого приложения через web-интерфейс. Пользовательский интерфейс такого приложения формируется средствами статического HTML (выполнение JavaScript не предусматривается), вся прикладная логика выполняется на сервере. Для работы тонкого клиента достаточно лишь обеспечить возможность запуска web-браузера, в окне которого и осуществляются все действия. По этой причине web-браузер часто называют "универсальным клиентом". | Таковым является рабочая станция или персональный компьютер, работающие под управлением собственной дисковой операционной системы и имеющие необходимый набор программного обеспечения. К сетевым серверам «толстые» клиенты обращаются, в основном, за дополнительными услугами (например, доступ к web-серверу или корпоративной базе данных).  Так же под «толстым» клиентом подразумевается и клиентское сетевое приложение, запущенное под управлением локальной ОС. Такое приложение совмещает компонент представления данных (графический пользовательский интерфейс ОС) и прикладной компонент (вычислительные мощности клиентского компьютера). | своего рода, компромисс между «толстым» и «тонким» клиентом. Как и «тонкий» клиент, «rich»-клиент также представляет графический интерфейс, описываемый уже средствами XML и включающий некоторую функциональность толстых клиентов (например, интерфейс drag-and-drop, вкладки, множественные окна, выпадающие меню и т.п.)  Прикладная логика «rich»-клиента также реализована на сервере. Данные отправляются в стандартном формате обмена, на основе того же XML (протоколы SOAP, XML-RPC) и интерпретируются клиентом. |

Некоторые основные протоколы «rich»-клиентов на базе XML приведены ниже:

* XAML (eXtensible Application Markup Language) — разработан Microsoft и используется в приложениях на платформе .NET.
* XUL (XML User Interface Language) — стандарт, разработанный в рамках проекта Mozilla, используется, например, в почтовом клиенте Mozilla Thunderbird или браузере Mozilla Firefox.
* Flex — мультимедийная технология на основе XML, разработанная Macromedia/Adobe.

**Протокол передачи данных** — набор соглашений интерфейса логического уровня, которые определяют обмен данными между различными программами. Эти соглашения задают единообразный способ передачи сообщений и обработки ошибок при взаимодействии ПО.

**Сетевой протокол**— набор правил и действий (очерёдности действий), позволяющий осуществлять соединение и обмен данными между двумя и более включёнными в сеть устройствами.

***2) Что ткое HTTP и HTTPS***

HTTP (от англ. *HyperText Transfer Protocol* — протокол передачи гипертекста) — это прикладной протокол передачи данных в сети. На текущий момент используется для получения информации с веб-сайтов. Протокол HTTP основан на использовании технологии «клиент-сервер»: клиент, отправляющий запрос, является инициатором соединения; сервер, получающий запрос, выполняет его и отправляет клиенту результат.

HTTPS (от англ. *HyperText Transfer Protocol Secure* — безопасный протокол передачи гипертекста) — это расширение протокола HTTP, поддерживающее шифрование посредством криптографических протоколов SSL и TLS.

**Чем отличаются HTTP от HTTPS**

* HTTPS не является отдельным протоколом передачи данных, а представляет собой расширение протокола HTTP с надстройкой шифрования;
* передаваемые по протоколу HTTP данные не защищены, HTTPS обеспечивает конфиденциальность информации путем ее шифрования;
* HTTP использует порт 80, HTTPS — порт 443.

***3) HTTP методы***

**Краткий обзор HTTP методов**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **GET** | **HEAD** | **POST** | **PUT** | **DELETE** | **CONNECT** | **OPTIONS** | **TRACE** |
| Метода GET в HTTP используется для получения информации от сервера по заданному URI ([URI в HTTP](https://zametkinapolyah.ru/servera-i-protokoly/uri-v-http-protokole.html)). Запросы клиентов, использующие метод GET должны получать только данные и не должны никак влиять на эти данные.( URI. Запросы с использованием GET должны только извлекать данные и не должны иметь никакого другого воздействия на данные.) | работает точно так же, как GET, но в ответ сервер посылает только заголовки и статусную строку без тела [HTTP сообщения](https://zametkinapolyah.ru/servera-i-protokoly/tema-3-http-soobshhenie-tipy-http-soobshhenij-zagolovki-http-soobshhenij-telo-soobshheniya-i-primery-http-soobshhenij.html).( То же, что и GET, но передает только строку состояния и раздел заголовка.) | используется для отправки данных на сервер, например, из HTML форм, которые заполняет посетитель сайта.  (POST-запрос используется для отправки данных на сервер, например, информации о клиенте, загрузки файла и т. д. с использованием HTML-форм.) | используется для загрузки содержимого запроса на указанный в этом же запросе URI.  (Заменяет все текущие представления целевого ресурса загруженным содержимым.) | удаляет указанный в URI ресурс.  (Удаляет все текущие представления целевого ресурса, заданного URI.) | преобразует существующее соединение в тоннель.  (Устанавливает туннель к серверу, идентифицированному данным URI.) | используется для получения параметров текущего HTTP соединения.  (Описывает параметры связи для целевого ресурса.) | создает петлю, благодаря которой клиент может увидеть, что происходит с сообщением на всех узлах передачи.  (Выполняет циклическую проверку сообщения по пути к целевому ресурсу.) |

**HTTP метод GET**

 позволяет получать информацию с [HTTP сервера](https://zametkinapolyah.ru/servera-i-protokoly/http-server-ili-veb-server-naznachenie-funkcii-i-rol-servera-v-http.html).

Информация, получаемая от сервера может быть любой, главное, чтобы она была в форме [HTTP объекта](https://zametkinapolyah.ru/servera-i-protokoly/tema-6-http-obekty-http-entity.html).

Доступ к информации при использовании метода GET осуществляется через URI.

Часто бывает так, что HTTP  метод GET обращается к какому-то коду, а не к конкретной страницы (все CMS генерируют контент налету), поэтому метод GET работает так, что мы получаем не исходный код, который генерирует текст, а сам текст.

**HTTP метод GET** бывает двух видов:

**условный** метод GET и **частичный** метод GET.

Давайте сперва посмотрим на **условный метод** GET. Когда используется условный HTTP метод GET, то к HTTP сообщению добавляются следующие [поля заголовков](https://zametkinapolyah.ru/servera-i-protokoly/tema-10-spravochnik-polej-http-zagolovkov-spisok-polej-http-zagolovka-zagolovki-http-soobshhenij-zaprosov-i-otvetov.html): If-Modified-Since, If-Unmodified-Since, If-Match, If-None-Match, или If-Range. Значение таких полей является какое-либо условие и если это условие выполняется, то происходит передача объекта, который хранится по указанному URI, если же условие не выполняется, то и сервер не передает никаких данных. Условный HTTP метод GET предназначен для уменьшения нагрузки на сеть.

Давайте теперь посмотрим на особенности работы **частичного HTTP метода GET**. Особенность частичного метода GET заключается в том, что в его заголовке присутствует поле Range. Когда используется частичные метод GET полезная информация, предназначенная для человека передается кусками, после чего она из этих кусков собирается. Не напоминает ли это вам скачивание файлов по HTTP протоколу, когда мы можем остановить загрузку, отключить браузер, потом опять включить браузер и закачка будет происходить ровно с того места, где она была приостановлена. Не стоит забывать, что поля заголовков — это [параметры HTTP протокола](https://zametkinapolyah.ru/servera-i-protokoly/tema-2-parametry-http-protokola.html), которые определяют, как будут работать [клиент и сервер](https://zametkinapolyah.ru/veb-programmirovanie/zametki-o-php/preimushhestva-dinamicheskix-stranic-dlya-chego-ispolzuyut-php-vzaimodejstvie-klient-server-s-ispolzovaniem-php.html).

[Сервер может кэшировать ответы на запросы](https://zametkinapolyah.ru/servera-i-protokoly/tema-13-keshirovanie-v-http-mexanizmy-klientskogo-i-servernogo-kesha-v-http.html) с **HTTP методом GET**, но при соблюдение определенных требований, о которых мы поговорим чуть позже. Давайте лучше самостоятельно напишем HTTP запрос с методом GET и посмотрим, какой ответ мы можем получить от сервера:

**Описание HTTP метода POST.**

**HTTP метод POST** является вторым по использованию в Интернете и нужен для того, чтобы отправлять данные на сервер. HTTP метод POST позволяет отправлять данные на сервер. Разработчики ввели метод POST в HTTP  стандарт, чтобы клиенты могли:

* оставлять сообщения на различных Интернет-ресурсах;
* передавать информацию о себе, заполняя HTML формы;

То, как будет работать метод POST определяется исключительно на стороне сервера и обычно зависит от запрашиваемого URI. Если сравнить URI, которому обращается клиент и сообщение, которое он хочет отправить с файловой системой, то URI – это папка, а сообщение клиента – это файл, который лежит в папке.

В результате выполнения **HTTP метода POST** сервер не обязательно в качестве ресурса выдает URI, [код состояния сервера](https://zametkinapolyah.ru/servera-i-protokoly/spisok-kodov-sostoyaniya-http-servera-spravochnik-kodov-sostoyaniya-http-servera.html) при использовании HTTP метода POST может быть [200](https://zametkinapolyah.ru/servera-i-protokoly/uspeshnye-kody-sostoyaniya-http-servera-200-201-202-203-204-205-206.html)(в этом случае вы получите какой-либо ресурс), либо 204 (в этом случае вы не получите никакого содержимого). Ответы сервера на метод POST не кэшируются, но это можно сделать принудительно, если использовать поле Cache-Control или Expires в заголовке.

Мы рассмотрели **HTTP метод POST**, давайте теперь посмотрим на HTTP метод HEAD, который очень похож на метод GET.

**Описание HTTP метода HEAD. Пример использования HTTP метода HEAD**

**HTTP метод HEAD** работает точно так же, как и метод GET, с той лишь разницей, что сервер в ответ не посылает тело HTTP сообщения. Все заголовки ответа при запросе клиента с использованием метода HEAD идентичны тем заголовкам, которые бы были, если бы использовался метод GET. Обычно **HTTP метод HEAD** используется для получения метаинформации об объекте без пересылки тела HTTP сообщения. Метод HEAD часто используется для тестирования HTTP соединений и достижимости узлов и ресурсов, так как нет необходимости гонять по сети содержимое, тестирование HTTP методом HEAD производится гораздо быстрее. Сервер может кэшировать свои ответы на запросы с методом HEAD. Еще одно применение метода HEAD заключается в [обсуждение HTTP содержимого](https://zametkinapolyah.ru/servera-i-protokoly/tema-12-process-obsuzhdenie-soderzhimogo-v-http-pri-peredachi-dannyx.html).

Давайте лучше самостоятельно напишем HTTP запрос с методом HEAD и посмотрим, какой ответ мы можем получить от сервера:

Вы можете посмотреть на пример использования метода GET и сравнить в чем **разница между двумя HTTP методами: HEAD и GET**. Давайте перейдем к рассмотрению HTTP метода OPTIONS.

**Описание HTTP метода OPTIONS. Пример использования HTTP метода OPTIONS**

**HTTP метод OPTIONS** используется для получения параметров HTTP соединения и другой служебной информации. Обратите внимание на то, что метод OPTIONS дает возможность запросить параметры для конкретного ресурса, указанного в URI.  Особенность **HTTP метода OPTIONS** заключается в том, что он не производит никаких действий с самим ресурсом (если браузер будет использовать метод OPTIONS, то он даже не станет загружать страницу).

Сервер отвечает на запрос с методом OPTIONS только опциями соединения, например он посылает поля заголовков Allow, но не пошлет Content-Type, ответы сервера на запросы с методом OPTIONS не кэшируются. Если в качестве URI указана звездочка «\*», то параметры соединения передаются для сервера в целом, а не для какого-то конкретного URL. Этот метод [не самый безопасный для HTTP сервера](https://zametkinapolyah.ru/servera-i-protokoly/tema-14-bezopasnost-v-http-protokole-bezopasnost-peredachi-dannyx-po-http-protokolu.html), поэтому зачастую клиенты его не могут применять из-за настроек безопасности.

Давайте посмотрим **пример запроса с HTTP методом OPTIONS**:

**Описание HTTP метода PUT. Пример использования HTTP метода PUT**

**HTTP метод PUT** используется для загрузки содержимого запроса на указанный в этом же запросе URI. То есть HTTP запрос с методом PUT уже заранее содержат в теле сообщения какой-то объект, который должен быть сохранен на сервере по адресу, который указан в URI. Но если по данному URI уже есть какие-либо данные, то данные, поступающие из запроса с методом PUT, считаются модификацией. Если запрос с **HTTP методом PUT** обращается к не существующему URI, то сервер создает новый URI и сообщает об этом клиенту. Если ресурс успешно создан по средствам метода PUT, то сервер возвращает ответ с [кодом состояния](https://zametkinapolyah.ru/servera-i-protokoly/tema-9-http-kody-sostoyaniya-klassy-kodov-sostoyaniya-http-servera.html) 201, если ресурс успешно модифицирован, то сервер вернет код 200, либо 204. Если по каким-либо причинам серверу не удается создать ресурс, то в ответ клиенту он высылает описание проблемы, возможно, с [кодом ошибки клиента](https://zametkinapolyah.ru/servera-i-protokoly/http-kody-oshibok-klienta.html) или [кодом ошибки сервера](https://zametkinapolyah.ru/servera-i-protokoly/http-kody-oshibok-servera-500-501-502-503-504-505.html).

Ответы сервера на **HTTP метод PUT** не кэшируются. Стоит обратить внимание, что метод POST и метод PUT выполняют совершенно разные операции. Метод POST обращается к ресурсу (странице или коду), которая содержит механизмы обработки сообщения метода POST, а вот метод PUT создает какой-то объект по URI, указанному в сообщение с HTTP методом PUT.

Давайте теперь посмотрим **пример работы HTTP метода PUT**:

Сервер в этом случае сохранит файл hello.htm, он будет доступен по указанному URI, в самом файле будет находиться HTML код, который указан в теле сообщения, а в ответ сервер отправит примерно следующее:

Мы рассмотрели всё, что качается **метода PUT**. Давайте теперь перейдем к рассмотрению HTTP метода DELETE

**Описание HTTP метода DELETE. Пример использования HTTP метода DELETE**

**HTTP метод DELETE** используется для удаления ресурса, указанного в URI. Действие метода DELETE может быть отменено вмешательством администратора HTTP сервера или программным кодом. Даже в том случае, когда сервер отправит вам код 200 после обработки метода DELETE, это не будет означать, что ресурс удален, это всего лишь означает, что сервер вас понял и обработал ваш запрос. Ответы сервера на **HTTP метод DELETE** не кэшируются.

Давайте теперь рассмотрим пример HTTP запроса, который использует метод DELETE:

Мы разобрали **HTTP метод DELETE**, давайте теперь рассмотрим метод TRACE.

**Описание HTTP метода TRACE. Пример использования HTTP метода TRACE**

**HTTP метод TRACE** используется для получения информации о том, что происходит с сообщением на промежуточных узлах. У сообщений с HTTP методом TRACE есть конечный получатель, конечный получатель определяется значением поля заголовка Max-Forwards: первый HTTP сервер, прокси-сервер или шлюз, получивший данное сообщение с значением Max-Forwards 0 является конечным получателем. Запросы с **HTTP методом TRACE не должны содержать объектов**.

**HTTP метод TRACE применяется для диагностики**, он позволяет видеть клиенту, что происходит в каждом звене цепочки между компьютером клиента и конечным получателем, для этого существует специальное поле Via. Ответы сервера на метод TRACE не кэшируются.

Давайте теперь посмотрим **пример HTTP метода TRACE**:

Мы рассмотрели **HTTP метод TRACE**, давайте рассмотрим последний метод HTTP протокола – метод CONNECT.

**Описание HTTP метода CONNECT. Пример использования HTTP метода CONNECT**

**HTTP метод CONNECT** используется для преобразования HTTP соединения в прозрачный TCP/IP туннель.  Пожалуй, это всё, что можно сказать про HTTP метод CONNECT в контексте рассматриваемого протокола, разве что стоит добавить, что данный метод используется в основном для шифрования соединения (не путайте с [кодировкой сообщений](https://zametkinapolyah.ru/servera-i-protokoly/kodirovanie-v-http-kodovye-tablicy-v-http.html)).

Давайте посмотрим **пример использования HTTP метода CONNECT**:

4) HTTP статус коды сервера

* [**1xx: информационный**](https://restfulapi.net/http-status-codes/#1xx) — передает информацию на уровне протокола передачи.
* [**2xx: Успех**](https://restfulapi.net/http-status-codes/#2xx) — указывает, что запрос клиента был успешно принят.
* [**3xx: перенаправление**](https://restfulapi.net/http-status-codes/#3xx) — указывает, что клиент должен предпринять дополнительные действия, чтобы выполнить свой запрос.
* [**4xx: Ошибка клиента**](https://restfulapi.net/http-status-codes/#4xx) — эта категория кодов состояния ошибки указывает на клиентов.
* [**5xx: Ошибка**](https://restfulapi.net/http-status-codes/#5xx) сервера. Сервер берет на себя ответственность за эти коды состояния ошибки.

## Коды состояния 1xx [информация]

|  |  |
| --- | --- |
| **Код состояния** | **Описание** |
| **100 Продолжить** | Промежуточный ответ. Указывает клиенту, что начальная часть запроса получена и еще не отклонена сервером. Клиент ДОЛЖЕН продолжить отправку оставшейся части запроса или, если запрос уже был выполнен, проигнорировать этот ответ. Сервер ДОЛЖЕН отправить окончательный ответ после завершения запроса. |
| **101 Протокол переключения** | Отправляется в ответ на заголовок запроса на [обновление](https://developer.mozilla.org/en-US/docs/Web/HTTP/Protocol_upgrade_mechanism) от клиента и указывает протокол, на который переключается сервер. |
| **102 Обработка (WebDAV)** | Указывает, что сервер получил и обрабатывает запрос, но ответа пока нет. |
| **103 Ранние подсказки** | В первую очередь предназначен для использования с Linkзаголовком. Он предлагает агенту пользователя начать предварительную загрузку ресурсов, пока сервер готовит окончательный ответ. |

## 2xx коды состояния [Успех]

|  |  |
| --- | --- |
| **Код состояния** | **Описание** |
| **200 ОК** | Указывает, что запрос выполнен успешно. |
| **201 Создано** | Указывает, что запрос выполнен успешно и в результате был создан новый ресурс. |
| **202 Принято** | Указывает, что запрос получен, но еще не выполнен. Обычно он используется в запросах на ведение журнала и пакетной обработке. |
| **203 Неофициальная информация** | Указывает, что возвращенная метаинформация в заголовке объекта не является окончательным набором, доступным на исходном сервере, а собрана из локальной или сторонней копии. Представленный набор МОЖЕТ быть подмножеством или надмножеством исходной версии. |
| **204 Нет содержания** | Сервер выполнил запрос, но ему не нужно возвращать тело ответа. Сервер может вернуть обновленную метаинформацию. |
| **205 Сбросить содержимое** | Указывает клиенту сбросить документ, отправивший этот запрос. |
| **206 Частичное содержание** | Он используется, когда Rangeзаголовок отправляется от клиента для запроса только части ресурса. |
| **207 Мультистатус (WebDAV)** | Индикатор для клиента о том, что произошло несколько операций и что статус каждой операции можно найти в тексте ответа. |
| **208 Уже сообщалось (WebDAV)** | Позволяет клиенту сообщить серверу, что тот же ресурс (с той же привязкой) упоминался ранее. Он никогда не отображается как истинный код ответа HTTP в строке состояния и появляется только в теле сообщения. |
| **226 ИМ Используется** | Сервер выполнил запрос GET для ресурса, и ответ представляет собой представление результата одной или нескольких манипуляций с экземпляром, примененных к текущему экземпляру. |

## 3xx коды состояния [перенаправление]

|  |  |
| --- | --- |
| **Код состояния** | **Описание** |
| **300 множественных вариантов** | Запрос имеет более одного возможного ответа. Пользовательский агент или пользователь должен выбрать один из них. |
| **301 Перемещено навсегда** | URL запрошенного ресурса был изменен навсегда. Новый URL задается Locationполем заголовка в ответе. Этот ответ можно кэшировать, если не указано иное. |
| **302 найдено** | URL запрошенного ресурса был временно изменен. Новый URL задается Locationполем в ответе. Этот ответ можно кэшировать только в том случае, если он указан в поле заголовка Cache-Controlили .Expires |
| **303 См. Другое** | Ответ можно найти по другому URI, и его СЛЕДУЕТ извлекать с помощью метода GET для этого ресурса. |
| **304 Не изменено** | Указывает клиенту, что ответ не был изменен, поэтому клиент может продолжать использовать ту же кэшированную версию ответа. |
| **305 Использовать прокси (устарело)** | Указывает, что запрошенный ответ должен быть доступен через прокси. |
| **306 (не используется)** | Это зарезервированный код состояния, который больше не используется. |
| **307 Временное перенаправление** | Указывает, что клиент должен получить запрошенный ресурс по другому URI тем же методом, который использовался в предыдущем запросе. Это похоже на 302 Foundто, за одним исключением, что будет использоваться тот же метод HTTP, который использовался в предыдущем запросе. |
| **308 Постоянное перенаправление (экспериментальное)** | Указывает, что ресурс теперь постоянно находится по другому URI, указанному в Locationзаголовке. Это похоже на 301 Moved Permanentlyто, за одним исключением, что будет использоваться тот же метод HTTP, который использовался в предыдущем запросе. |

## Коды состояния 4xx (ошибка клиента)

|  |  |
| --- | --- |
| **Код состояния** | **Описание** |
| **ошибка 400, неверный запрос** | Запрос не может быть понят сервером из-за неправильного синтаксиса. Клиент НЕ ДОЛЖЕН повторять запрос без изменений. |
| **401 Неавторизованный** | Указывает, что для запроса требуется информация для аутентификации пользователя. Клиент МОЖЕТ повторить запрос с подходящим полем заголовка авторизации. |
| **402 Требуется оплата (экспериментальная версия)** | Зарезервировано для использования в будущем. Он предназначен для использования в цифровых платежных системах. |
| **403 Запрещено** | Несанкционированный запрос. У клиента нет прав доступа к содержимому. В отличие от 401, личность клиента известна серверу. |
| **404 Не Найдено** | Сервер не может найти запрошенный ресурс. |
| **405 Метод не разрешен** | HTTP-метод запроса известен серверу, но он отключен и не может использоваться для этого ресурса. |
| **406 Недопустимо** | Сервер не находит контента, соответствующего критериям, заданным пользовательским агентом в Acceptзаголовке, отправленном в запросе. |
| **407 Требуется аутентификация прокси** | Указывает, что клиент должен сначала аутентифицировать себя с помощью прокси. |
| **408 Время ожидания запроса** | Указывает, что сервер не получил полный запрос от клиента в течение выделенного сервером периода ожидания. |
| **409 Конфликт** | Запрос не может быть выполнен из-за конфликта с текущим состоянием ресурса. |
| **410 ушел** | Запрошенный ресурс больше не доступен на сервере. |
| **411 Требуемая длина** | Сервер отказывается принимать запрос без определенной Content-Length. Клиент МОЖЕТ повторить запрос, если он добавляет допустимое Content-Lengthполе заголовка. |
| **412 Предварительное условие не выполнено** | Клиент указал в своих заголовках предварительные условия, которым сервер не соответствует. |
| **413 Слишком большой объект запроса** | Объект запроса превышает ограничения, установленные сервером. |
| **414 Request-URI слишком длинный** | URI, запрошенный клиентом, длиннее, чем может интерпретировать сервер. |
| **415 Неподдерживаемый тип носителя** | Тип носителя Content-typeзапроса не поддерживается сервером. |
| **416 Запрошенный диапазон не удовлетворяется** | Диапазон, указанный Rangeполем заголовка в запросе, не может быть выполнен. |
| **417 Ожидание не оправдалось** | Ожидание, указанное в Expectполе заголовка запроса, не может быть выполнено сервером. |
| **418 Я чайник (RFC 2324)** | Это было определено как апрельская шутка, и ожидается, что оно не будет реализовано реальными HTTP-серверами. ( [RFC 2324](https://tools.ietf.org/html/rfc2324) ) |
| **420 Улучшите свое спокойствие (Twitter)** | Возвращается Twitter Search and Trends API, когда скорость клиента ограничена. |
| **422 Необрабатываемый объект (WebDAV)** | Сервер понимает тип содержимого и синтаксис объекта запроса, но по какой-то причине сервер не может обработать запрос. |
| **423 Заблокировано (WebDAV)** | Ресурс, к которому осуществляется доступ, заблокирован. |
| **424 Неудачная зависимость (WebDAV)** | Запрос не выполнен из-за отказа предыдущего запроса. |
| **425 Слишком рано (WebDAV)** | Указывает, что сервер не хочет рисковать обработкой запроса, который может быть воспроизведен. |
| **426 Требуется обновление** | Сервер отказывается выполнять запрос. Сервер обработает запрос после того, как клиент перейдет на другой протокол. |
| **428 Требуется предварительное условие** | Исходный сервер требует, чтобы запрос был условным. |
| **429 Слишком много запросов** | Пользователь отправил слишком много запросов за заданный промежуток времени («ограничение скорости»). |
| **431 Поля заголовка запроса слишком велики** | Сервер не желает обрабатывать запрос, поскольку его поля заголовка слишком велики. |
| **444 Нет ответа (Nginx)** | Сервер Nginx не возвращает клиенту никакой информации и закрывает соединение. |
| **449 Повторить попытку с (Майкрософт)** | Запрос должен быть повторен после выполнения соответствующего действия. |
| **450 Заблокировано родительским контролем Windows (Microsoft)** | Родительский контроль Windows включен и блокирует доступ к данной веб-странице. |
| **451 Недоступно по юридическим причинам** | Пользовательский агент запросил ресурс, который не может быть предоставлен по закону. |
| **499 Закрытый запрос клиента (Nginx)** | Соединение закрывается клиентом, пока HTTP-сервер обрабатывает свой запрос, из-за чего сервер не может отправить заголовок HTTP обратно. |

## Коды состояния 5xx (ошибка сервера)

|  |  |
| --- | --- |
| **Код состояния** | **Описание** |
| **внутренняя ошибка сервера 500** | Сервер столкнулся с непредвиденной ситуацией, из-за которой он не смог выполнить запрос. |
| **501 Не реализовано** | Метод HTTP не поддерживается сервером и не может быть обработан. |
| **502 Неверный шлюз** | Сервер получил недопустимый ответ, работая в качестве шлюза, чтобы получить ответ, необходимый для обработки запроса. |
| **503 Сервис недоступен** | Сервер не готов обработать запрос. |
| **Ошибка 504 Время ответа сервера истекло** | Сервер действует как шлюз и не может вовремя получить ответ на запрос. |
| **505 Версия HTTP не поддерживается (экспериментальная)** | Версия HTTP, используемая в запросе, не поддерживается сервером. |
| **506 Вариант также ведет переговоры (экспериментальный)** | Указывает, что на сервере произошла внутренняя ошибка конфигурации: выбранный ресурс варианта настроен на участие в прозрачном согласовании содержимого и, следовательно, не является надлежащей конечной точкой в ​​процессе согласования. |
| **507 Недостаточно памяти (WebDAV)** | Этот метод не может быть выполнен для ресурса, так как сервер не может сохранить представление, необходимое для успешного выполнения запроса. |
| **508 Обнаружена петля (WebDAV)** | Сервер обнаружил бесконечный цикл при обработке запроса. |
| **510 Не расширен** | Для выполнения сервером требуются дополнительные расширения запроса. |
| **511 Требуется сетевая аутентификация** | Указывает, что клиент должен пройти аутентификацию, чтобы получить доступ к сети. |

***5) Что такое ядро браузера***

Ядро браузера можно разделить на две части: движок рендеринга (инженер макета или движок рендеринга) и движок JS.  
Механизм рендеринга. Он отвечает за получение содержимого веб-страницы (HTML, XML, изображения и т. д.), организацию информации (например, добавление CSS и т. д.) и вычисление режима отображения веб-страницы, а затем вывод на монитор или принтер. Разница в ядре браузера будет по-разному интерпретировать синтаксис веб-страницы, поэтому эффект рендеринга также будет другим.  
Механизм JS анализирует язык Javascript и выполняет язык Javascript для достижения динамического эффекта веб-страницы.

Сначала не было четкого различия между движком рендеринга и движком JS, а позже движок JS становился все более и более независимым, и ядро ​​ссылалось только на движок рендеринга. Команда проекта веб-стандартов создала ACID для проверки совместимости и производительности движка.

***6) Какие браузеры какиие ядра используют***

Существует много типов ядер, таких как добавление некоммерческих бесплатных ядер, которые никто не использует, может быть более 10 видов, но общие ядра браузера можно разделить на четыре типа: Trident, Gecko, Blink, Webkit.

（1）**Trident(IE**Ядро)

Trident, который называется «Режим совместимости», является одним из ядер многих двухъядерных браузеров в Китае.

Представители: IE, Maxthon, Window of the World Browser, Avant, Tencent TT, Cheetah Secure Browser, 360 Speed ​​Browser, Baidu Browser и др.

После выпуска Window10 IE назвал свой встроенный браузер Edge, наиболее заметной особенностью Edge является новое ядро ​​EdgeHTML.

（2）**Gecko(firefox**)

Gecko (ядро Firefox): Mozilla FireFox (браузер Firefox) использует это ядро, для Gecko характерен полностью открытый код, поэтому его можно разрабатывать в высокой степени, и программисты по всему миру могут писать для него код и добавлять функции. Жаль, что эти годы упали, такие как медленная скорость открытия, частые обновления, свирепый напарник по команде, богоподобный хром противника.

（3） **webkit(Safari**)

Safari - это браузер, разработанный Apple, и название используемого ядра браузера - известный WebKit.

Сейчас многие люди по ошибке называют webkit ядром Chrome (даже если ядро ​​Chrome уже мигает), Apple чувствует, что у него украли жену, и они снова плачут и кружатся в туалете.

Типичные браузеры: Maxthon Browser 3, Apple Safari (Win / Mac / iPhone / iPad), мобильный браузер Symbian, браузер Android по умолчанию,

（4） **Chromium/Bink(chrome**)

В проекте Chromium механизм рендеринга Blink (то есть ядро ​​браузера) был разработан и встроен в браузер Chrome. Blink на самом деле является веткой WebKit.

В последней версии большинства отечественных браузеров используется ядро ​​Blink.

（5） **Presto(Opera**)

Presto является «предшественником» ядра норвежской браузерной оперы, поэтому его называют «предшественником», потому что последний браузер оперы давно отказался от него и поставил его в объятия Google.

Ядро браузера мобильного терминала в основном относится к ядру браузера, встроенного в систему.

В настоящее время в браузерах мобильных устройств чаще всего используются ядра Webkit, Blink, Trident, Gecko и т. Д. Среди них платформы Apple iOS, такие как iPhone и iPad, в основном WebKit, а ядро ​​браузера системы Android до Android 4.4 - это WebKit, переключение браузера системы Android4.4. В Chromium ядро ​​- Blink, ветвь Webkit, а браузерное ядро ​​системы Windows Phone 8 - Trident.

***7) Что такое API***

Аббревиатура API расшифровывается как «Application Programming Interface» (интерфейс программирования приложений, программный интерфейс приложения)

API (Application Programming Interface — программный интерфейс приложения, или интерфейс программирования приложений) — специальный протокол для взаимодействия компьютерных программ, который позволяет использовать функции одного приложения внутри другого.

## Использование Postman

Postman использует протокол HTTP для взаимодействия между серверами. Он доступен как в веб-версии, так и в виде настольного приложения с графическим интерфейсом.

возаожности:

· Создание и отправка запросов

· Изменение параметров запросов.

· Сохранение созданных запросы в коллекции и папках.

· Запуск автоматических тестов на основе коллекций при помощи Collection Runner.

· Изменение окружения для одинаковых запросов.

· Добавление контрольных точек к вызову API.

***8. Что такое ендпоинты***

Встречаются немного различные интерпретации этого понятия. Например такая. Каждый api-сервис имеет endpoint, к которому и нужно обращаться, например отправлять http-запрос. Обычно это url. Т.о. endpoint - это url. Или такая. url включает в себя маршрут (основная часть адреса сервиса) и endpoint (часть url, которая содержит например имя вызываемого метода). Существует один маршрут и у него несколько эндпоинтов. Т.о. здесь endpoint - это только часть url.

Endpoint - это конечное защищаемое устройство. Компьютер с корпоративным антивирусом или мобилка с ним же. Хотя тут зависит от контекста, который вы не прояснили.

Если говорить про веб, а чаще всего этот термин юзается в веб, то  
На mysite.com может висеть ваше приложение  
И у него могут быть ендпоинты  
mysite.com/healthcheck  
mysite.com/action.php

Как узнать Эндпоинт?

**Из командной строки**

1. Запустите командную строку: Пуск -> cmd -> Enter.
2. Перейдите в папку с установленным Kaspersky **Endpoint** Security 10 для Windows.
3. Выполните команду: avp.com statistics | find "**Endpoint**".

Проще говоря, конечная точка — это один конец канала связи. Когда API взаимодействует с другой системой, точки взаимодействия этого взаимодействия считаются конечными точками. Для API конечная точка может включать URL-адрес сервера или службы. Каждая конечная точка — это место, из которого API-интерфейсы могут получить доступ к ресурсам, необходимым им для выполнения своих функций.

API работают с использованием «запросов» и «ответов». Когда API запрашивает информацию у веб-приложения или веб-сервера, он получает ответ. Место, куда API отправляют запросы и где находится ресурс, называется конечной точкой.

***9) URL (URI, URL, URN)***

**URL** - Uniform Resource Locator (унифицированный **определитель местонахождения** ресурса)

**URN** - Unifrorm Resource Name (унифицированное **имя** ресурса)

**URI** - Uniform Resource Identifier (унифицированный **идентификатор** ресурса)

Внимание! Далее в мелочах кроется истина, и пока ничего не понятно, - какая-то каша, но, едем дальше.

### В чем различия

**URL**: Исторически возник самым первым из понятий и закрепился как синоним термина веб-адрес. URL определяет местонахождение ресурса в сети и способ его (ресурса) извлечения.  
Это позволяет нам полностью узнать: как, кому и где можно достать требуемый ресурс, вводя понятия схемы, данных авторизации и местонахождения.

**URN**: Неизменяемая последовательность символов определяющая только имя некоторого ресурса. Смысл URN в том, что им единоразово и уникально именуется какая-либо сущность в рамках конкретного пространства имен (контекста), либо без пространства имен, в общем (что не желательно). Таким образом, URN способен преодолеть недостаток URL связанный с возможным будущим изменением и перемещением ссылок, однако, теперь для того, чтобы знать местонахождение URN ресурса необходимо обращаться к системе разрешения имен URN, в которой он должен быть зарегистрирован.

**URI**: Это лишь обобщенное понятие (множество) идентификации ресурса, включающее в нашем случае как URL, так и URN, как по отдельности, так и совместно. Т.е. мы можем считать, что: URI = URL или URI = URN или URI = URL + URN

### Подведем итоги

URI - это абстракция концепции идентификации,  
а URL и URN - это конкретные реализации - полного адреса ресурса и уникального контекстного имени соответственно.

***10) Идемпотентные HTTP методы***

Метод HTTP является идемпотентным, если повторный идентичный запрос, сделанный один или несколько раз подряд, имеет один и тот же эффект, не изменяющий состояние сервера. Другими словами, идемпотентный метод не должен иметь никаких побочных эффектов (side-effects), кроме сбора статистики или подобных операций. Корректно реализованные методы [GET](https://developer.mozilla.org/ru/docs/Web/HTTP/Methods/GET), [HEAD](https://developer.mozilla.org/ru/docs/Web/HTTP/Methods/HEAD), [PUT](https://developer.mozilla.org/ru/docs/Web/HTTP/Methods/PUT) и [DELETE](https://developer.mozilla.org/ru/docs/Web/HTTP/Methods/DELETE) **идемпотентны**, но не метод [POST](https://developer.mozilla.org/ru/docs/Web/HTTP/Methods/POST). Также все [безопасные](https://developer.mozilla.org/ru/docs/Glossary/safe) методы являются идемпотентными.

Метод считается «идемпотентным», если эффект на сервер от одного запроса такой же как от нескольких идентичных запросов такого типа. Из методов запроса, определенных в этой спецификации, PUT, DELETE и безопасные методы (не изменяют состояние сервера) запросов являются идемпотентными.

Из спецификации HTTP:

| **Метода** | **Безопасный** | **Идемпотентный** | **Ссылка** |
| --- | --- | --- | --- |
| CONNECT | нет | нет | [Section 4.3.6](https://tools.ietf.org/html/rfc7231#section-4.3.6) |
| DELETE | нет | да | [Section 4.3.5](https://tools.ietf.org/html/rfc7231#section-4.3.5) |
| GET | да | да | [Section 4.3.1](https://tools.ietf.org/html/rfc7231#section-4.3.1) |
| HEAD | да | да | [Section 4.3.2](https://tools.ietf.org/html/rfc7231#section-4.3.2) |
| OPTIONS | да | да | [Section 4.3.7](https://tools.ietf.org/html/rfc7231#section-4.3.7) |
| POST | нет | нет | [Section 4.3.3](https://tools.ietf.org/html/rfc7231#section-4.3.3) |
| PUT | нет | да | [Section 4.3.4](https://tools.ietf.org/html/rfc7231#section-4.3.4) |
| TRACE | да | да | [Section 4.3.8](https://tools.ietf.org/html/rfc7231#section-4.3.8) |

Согласно стандарту HTTP, запросы типа GET считаются идемпотентными.

***11) Безопасные HTTP методы***

Метод HTTP является **безопасным**, если он не меняет состояние сервера. Другими словами, безопасный метод проводит операции "только чтение" (read-only). Несколько следующих методов HTTP безопасные: [GET](https://developer.mozilla.org/ru/docs/Web/HTTP/Methods/GET), [HEAD](https://developer.mozilla.org/ru/docs/Web/HTTP/Methods/HEAD) или [OPTIONS](https://developer.mozilla.org/ru/docs/Web/HTTP/Methods/OPTIONS). Все безопасные методы являются также [идемпотентными](https://developer.mozilla.org/ru/docs/Glossary/Idempotent), как и некоторые другие, но при этом небезопасные, такие как [PUT](https://developer.mozilla.org/ru/docs/Web/HTTP/Methods/PUT) или [DELETE](https://developer.mozilla.org/ru/docs/Web/HTTP/Methods/DELETE).

Даже если безопасные методы являются по существу "только для чтения", сервер всё равно может сменить своё состояние: например, он может сохранять статистику. Что существенно, так то, когда клиент вызывает безопасный метод, то он не запрашивает никаких изменений на сервере, и поэтому не создаёт дополнительную нагрузку на сервер. Браузеры могут вызывать безопасные методы, не опасаясь причинить вред серверу: это позволяет им выполнять некоторые действия, например, предварительная загрузка без риска. Поисковые роботы также полагаются на вызовы безопасных методов.

Безопасные методы не обязательно должны обрабатывать только статичные файлы; сервер может генерировать ответ "на-лету", пока скрипт, генерирующий ответ, гарантирует безопасность: он не должен вызывать внешних эффектов, таких как формирование заказов, отправка писем и др..

Правильная реализация безопасного метода - это ответственность **серверного приложения**, потому что сам веб-сервер, будь то Apache, nginx, IIS это соблюсти не сможет. В частности, приложение не должно разрешать изменение состояния сервера запросами [GET](https://developer.mozilla.org/ru/docs/Web/HTTP/Methods/GET).

***12) Иденфикация, Аутентификация, Авторизация***

* Идентификация — процедура, в результате выполнения которой для субъекта идентификации выявляется его идентификатор, однозначно определяющий этого субъекта в информационной системе.
* Аутентификация — процедура проверки подлинности, например проверка подлинности пользователя путем сравнения введенного им пароля с паролем, сохраненным в базе данных.
* Авторизация — предоставление определенному лицу или группе лиц прав на выполнение определенных действий.

Скажем, пользователь хочет войти в свой аккаунт Google. Google подходит лучше всего, потому что там процедура входа явным образом разбита на несколько простейших этапов. Вот что при этом происходит:

* Для начала система запрашивает логин, пользователь его указывает, система распознает его как существующий — это **идентификация**.
* После этого Google просит ввести пароль, пользователь его вводит, и система соглашается, что пользователь, похоже, действительно настоящий, раз пароль совпал, — это [**аутентификация**](https://encyclopedia.kaspersky.ru/glossary/authentication/).
* Скорее всего, Google дополнительно спросит еще и одноразовый код из SMS или приложения. Если пользователь и его правильно введет, то система окончательно согласится с тем, что он настоящий владелец аккаунта, — это [**двухфакторная аутентификация**](https://encyclopedia.kaspersky.ru/glossary/two-factor-authentication/).
* После этого система предоставит пользователю право читать письма в его почтовом ящике и все в таком духе — это [**авторизация**](https://encyclopedia.kaspersky.ru/glossary/authorization/).

Аутентификация без предварительной идентификации лишена смысла — пока система не поймет, подлинность чего же надо проверять, совершенно бессмысленно начинать проверку. Для начала надо представиться.

Идентификация без аутентификации — это просто глупо. Потому что мало ли кто ввел существующий в системе логин! Системе обязательно надо удостовериться, что этот кто-то знает еще и пароль. Но пароль могли подсмотреть или подобрать, поэтому лучше подстраховаться и спросить что-то дополнительное, что может быть известно только данному пользователю: например, одноразовый код для подтверждения входа.

А вот авторизация без идентификации и тем более аутентификации очень даже возможна. Например, в Google Документах можно публиковать документы так, чтобы они были доступны вообще кому угодно. В этом случае вы как владелец файла увидите сверху надпись, гласящую, что его читает неопознанный енот. Несмотря на то, что енот совершенно неопознанный, система его все же авторизовала — то есть выдала право прочитать этот документ.

***13) Что такое IP***

IP-адрес – это уникальный адрес, идентифицирующий устройство в интернете или локальной сети. IP означает «Интернет-протокол» – набор правил, регулирующих формат данных, отправляемых через интернет или локальную сеть.

По сути, IP-адрес – это идентификатор, позволяющий передавать информацию между устройствами в сети: он содержит информацию о местоположении устройства и обеспечивает его доступность для связи. IP-адреса позволяют различать компьютеры, маршрутизаторы и веб-сайты в интернете и являются важным компонентом работы интернета.

IP-адрес – это строка чисел, разделенных точками. IP-адреса представляют собой набор из четырех чисел, например, 192.158.1.38. Каждое число в этом наборе принадлежит интервалу от 0 до 255. Таким образом, полный диапазон IP-адресации – это адреса от 0.0.0.0 до 255.255.255.255.

IP-адреса не случайны. Они рассчитываются математически и распределяются [Администрацией адресного пространства Интернета](https://www.iana.org/) (Internet Assigned Numbers Authority, IANA), подразделением [Корпорации по присвоению имен и номеров в Интернете](https://www.icann.org/) (Internet Corporation for Assigned Names and Numbers, ICANN). ICANN – это некоммерческая организация, основанная в США в 1998 году с целью поддержки безопасности интернета и обеспечения его доступности для всех пользователей. Каждый раз, когда кто-либо регистрирует домен в интернете, он пользуется услугами регистратора доменных имен, который платит ICANN небольшой сбор за регистрацию домена.

Использование IP-адресов обычно происходит незаметно. Процесс работает следующим образом:

1. Устройство подключается к интернету не напрямую: сначала оно подключается к сети, подключенной к интернету, а сеть, в свою очередь, предоставляет устройству доступ к интернету.
2. Если вы находитесь дома, скорее всего, этой сетью является сеть вашего интернет-провайдера. В офисе это будет сеть вашей компании.
3. IP-адрес назначается устройству вашим интернет-провайдером.
4. Ваша интернет-активность проходит через интернет-провайдера, а он перенаправляет вам ответы на запросы, используя ваш IP-адрес. Поскольку провайдер предоставляет доступ в Интернет, его роль заключается в назначении IP-адрес вашему устройству.
5. Однако ваш IP-адрес может измениться, например, при включение или выключение модема или маршрутизатора. Можно также связаться с интернет-провайдером, чтобы он изменил IP-адрес.
6. Если вы находитесь вне дома, например, путешествуете, и берете с собой устройство, домашний IP-адрес не закрепляется за устройством. Это связано с тем, что устройство будет использовать другую сеть (Wi-Fi в отеле, аэропорту, кафе) для доступа в интернет и другой временный IP-адрес, назначенный интернет-провайдером в отеле, аэропорту или кафе.

Типы IP-адресов

Существуют разные категории IP-адресов, и в каждой категории имеются разные типы.

Клиентские IP-адреса

У каждого человека или компании с тарифным планом на получение интернет-услуг есть два типа IP-адресов: частный и общедоступный. Термины частный и общедоступный относятся к сетевому расположению: частный IP-адрес используется внутри сети, а общедоступный – за пределами сети.

Частные IP-адреса

Каждое устройство, которое подключается к вашей интернет-сети, имеет частный IP-адрес. Это могут быть компьютеры, смартфоны, планшеты, а также любые устройства с поддержкой Bluetooth, такие как динамики, принтеры, смарт-телевизоры. С развитием интернета вещей растет и количество частных IP-адресов в домашней сети. Маршрутизатору необходимо идентифицировать каждое из этих устройств, а многие устройства также должны идентифицировать друг друга. Поэтому маршрутизатор генерирует частные IP-адреса, которые являются уникальными идентификаторами каждого устройства и позволяют различать их в сети.

Общедоступные IP-адреса

Общедоступный IP-адрес – это основной адрес, связанный со всей сетью. Каждое подключенное устройство имеет собственный IP-адрес, но они также включены в состав основного IP-адреса сети. Как было описано выше, общедоступный IP-адрес предоставляется маршрутизатору интернет-провайдером. Обычно у интернет-провайдеров есть большой пул IP-адресов, которые они присваивают клиентам. Общедоступный IP-адрес – это адрес, который устройства за пределами интернет-сети будут использовать для распознавания этой сети.

Общедоступные IP-адреса

Общедоступные IP-адреса бывают двух видов: динамические и статические.

Динамические IP-адреса

Динамические IP-адреса меняются автоматически и регулярно. Интернет-провайдеры покупают большой пул IP-адресов и автоматически присваивают их своим клиентам. Периодически они меняют присвоенные IP-адреса и помещают старые IP-адреса обратно в пул для использования другими клиентами. Обоснованием этого подхода служит экономия средств провайдера. Автоматизация регулярного изменения IP-адресов позволяет им не выполнять никаких действий для восстановления IP-адреса клиента, например, если он переезжает. Также имеются преимущества с точки зрения безопасности, поскольку изменение IP-адреса затрудняет взлом сетевого интерфейса злоумышленниками.

Статические IP-адреса

В отличие от динамических IP-адресов, статические IP- адреса остаются неизменными. После того, как сеть назначает IP-адрес, он остается неизменным. Большинству частных лиц и организаций не нужны статические IP-адреса, но для организаций, планирующих размещать собственные серверы, наличие статического IP-адреса крайне важно. Это связано с тем, что статический IP-адрес гарантирует, что привязанные к нему веб-сайты и адреса электронной почты будут иметь постоянные IP-адреса. Это очень важно, если требуется, чтобы другие устройства могли находить их в интернете.

В результате возникла классификация по типам IP-адресов веб-сайтов.

Два типа IP-адресов веб-сайтов

Для владельцев веб-сайтов, использующих пакет веб-хостинга (что характерно для большинства веб-сайтов), а не собственный сервер, существует два типа IP-адресов веб-сайтов: общие и выделенные.

Общие IP-адреса

Веб-сайты, использующие общие хостинговые планы от провайдеров веб-хостинга, обычно являются одним из многих веб-сайтов, размещенных на одном сервере. Это, как правило, веб-сайты физических лиц или компаний малого и среднего бизнеса, с ограниченным объемом трафика, количеством страниц и т. д. Такие веб-сайты имеют общие IP-адреса.

Выделенные IP-адреса

В некоторых тарифных планах веб-хостинга есть возможность приобрести выделенный IP-адрес (или адреса). Это может упростить получение SSL-сертификата и позволяет использовать собственный FTP-сервер (сервер протокола передачи файлов). Кроме того, это упрощает организацию общего доступа и передачу файлов в рамках организации и позволяет использовать анонимный FTP-доступ. Выделенный IP-адрес также позволяет получить доступ к веб-сайту, используя только IP-адрес, а не доменное имя. Это полезно, если требуется создать и протестировать его перед регистрацией домена.

***14) Что такое октаты в DNS***

***15) Что такое порт, сколько портов у Linux сервера***

Прежде всего, порт представляет собой логический объект, который является конечной точкой связи. Он непосредственно связан с процессом или службой в операционной системе. При этом, любой порт способен находиться в одном из четырех возможных состояний: **открытом**, **закрытом**, **отфильтрованном** (filtered) или **нефильтрованном**(unfiltered)**.** В том случае, если порт **открыт** — программное обеспечение целевого оборудования занимается тем, что прослушивает его соединения и пакеты. Открытый порт сам по себе не представляет опасности, но может стать угрозой безопасности системы, как только его начнет использовать программа. Для определения того, какая именно программа слушает конкретный порт в данный момент, существуют специальные утилиты — **Netstat**, **ss**, **Nmap**, **lsof**.

Прежде всего, данный материал расскажет о **четырех инструментах** для выявления **открытых портов**. И расскажет о том, как обнаружить программное обеспечение, прослушивающее конкретные порты в операционной системе [**Linux**](https://blog.sedicomm.com/2019/07/03/4-sposoba-najti-ip-adres-servera-v-terminale-linux/).

**Порт** ([англ.](https://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) *port*) — целое неотрицательное число, записываемое в заголовках [протоколов](https://ru.wikipedia.org/wiki/%D0%9F%D1%80%D0%BE%D1%82%D0%BE%D0%BA%D0%BE%D0%BB_%D0%BF%D0%B5%D1%80%D0%B5%D0%B4%D0%B0%D1%87%D0%B8_%D0%B4%D0%B0%D0%BD%D0%BD%D1%8B%D1%85) [транспортного уровня](https://ru.wikipedia.org/wiki/%D0%A2%D1%80%D0%B0%D0%BD%D1%81%D0%BF%D0%BE%D1%80%D1%82%D0%BD%D1%8B%D0%B9_%D1%83%D1%80%D0%BE%D0%B2%D0%B5%D0%BD%D1%8C) сетевой [модели OSI](https://ru.wikipedia.org/wiki/%D0%A1%D0%B5%D1%82%D0%B5%D0%B2%D0%B0%D1%8F_%D0%BC%D0%BE%D0%B4%D0%B5%D0%BB%D1%8C_OSI) ([TCP](https://ru.wikipedia.org/wiki/TCP), [UDP](https://ru.wikipedia.org/wiki/UDP), [SCTP](https://ru.wikipedia.org/wiki/SCTP), [DCCP](https://ru.wikipedia.org/wiki/DCCP)).

Обмен данными по сети ведётся между двумя процессами по определённому [протоколу](https://ru.wikipedia.org/wiki/%D0%9F%D1%80%D0%BE%D1%82%D0%BE%D0%BA%D0%BE%D0%BB_%D0%BF%D0%B5%D1%80%D0%B5%D0%B4%D0%B0%D1%87%D0%B8_%D0%B4%D0%B0%D0%BD%D0%BD%D1%8B%D1%85). Для установки соединения необходимы:

* номер [протокола](https://ru.wikipedia.org/wiki/%D0%9F%D1%80%D0%BE%D1%82%D0%BE%D0%BA%D0%BE%D0%BB_%D0%BF%D0%B5%D1%80%D0%B5%D0%B4%D0%B0%D1%87%D0%B8_%D0%B4%D0%B0%D0%BD%D0%BD%D1%8B%D1%85);
* два [IP-адреса](https://ru.wikipedia.org/wiki/IP-%D0%B0%D0%B4%D1%80%D0%B5%D1%81) (адрес хоста-отправителя и адрес хоста-получателя для [построения маршрута](https://ru.wikipedia.org/wiki/%D0%9C%D0%B0%D1%80%D1%88%D1%80%D1%83%D1%82%D0%B8%D0%B7%D0%B0%D1%86%D0%B8%D1%8F) между ними);
* два номера порта (порт процесса-отправителя и порт получателя).

Порт процесса-отправителя (источника) может быть постоянным (статическим) или назначаться [динамически](https://ru.wikipedia.org/wiki/%D0%94%D0%B8%D0%BD%D0%B0%D0%BC%D0%B8%D1%87%D0%B5%D1%81%D0%BA%D0%B8%D0%B9_%D0%BF%D0%BE%D1%80%D1%82) для каждого нового сеанса связи.

При соединении по протоколу [TCP](https://ru.wikipedia.org/wiki/TCP) порт процесса-отправителя используется:

* операционной системой хоста-получателя для отправки пакета-подтверждения о получении данных;
* процессом-получателем для отправки пакета-ответа.

При соединении по протоколу [UDP](https://ru.wikipedia.org/wiki/UDP) допустимо вместо порта процесса-отправителя указывать число ноль, означающее «порт не указан».

При соединении по протоколу [SCTP](https://ru.wikipedia.org/wiki/SCTP) в рамках ассоциации может использоваться:

* несколько портов процесса-отправителя (источника)
* несколько портов процесса-получателя.

Так как IP-адрес хоста-отправителя и номер порта процесса-отправителя являются аналогом обратного адреса, записываемого на почтовых [конвертах](https://ru.wikipedia.org/wiki/%D0%9A%D0%BE%D0%BD%D0%B2%D0%B5%D1%80%D1%82) (позволяют получателю отправить ответ отправителю), номер порта процесса-отправителя иногда называют «обратным» портом.

Если на хосте какой‑либо процесс постоянно использует один номер порта (например, процесс программы, реализующей [web-сервер](https://ru.wikipedia.org/wiki/%D0%92%D0%B5%D0%B1-%D1%81%D0%B5%D1%80%D0%B2%D0%B5%D1%80), может использовать порт 80 для приёма и передачи данных), говорят, что порт является «открытым».

Термины «открытый порт» и «закрытый порт» (заблокированный) также используются, когда речь идёт о фильтрации [сетевого трафика](https://ru.wikipedia.org/wiki/%D0%A1%D0%B5%D1%82%D0%B5%D0%B2%D0%BE%D0%B9_%D1%82%D1%80%D0%B0%D1%84%D0%B8%D0%BA).

Если процесс получил номер порта у ОС («открыл порт») и «держит его открытым» для приёма и передачи данных, говорят, что процесс *«прослушивает»* ([разг.](https://ru.wikipedia.org/wiki/%D0%A0%D0%B0%D0%B7%D0%B3%D0%BE%D0%B2%D0%BE%D1%80%D0%BD%D1%8B%D0%B9_%D1%81%D1%82%D0%B8%D0%BB%D1%8C) *слушает*, от [англ.](https://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) *listen*) порт.

Обычно прослушиванием порта занимается процесс программы, реализующей [сервер](https://ru.wikipedia.org/wiki/%D0%A1%D0%B5%D1%80%D0%B2%D0%B5%D1%80_(%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%BD%D0%BE%D0%B5_%D0%BE%D0%B1%D0%B5%D1%81%D0%BF%D0%B5%D1%87%D0%B5%D0%BD%D0%B8%D0%B5)) для какого-либо протокола. Процесс программы, реализующей [клиент](https://ru.wikipedia.org/wiki/%D0%9A%D0%BB%D0%B8%D0%B5%D0%BD%D1%82_(%D0%B8%D0%BD%D1%84%D0%BE%D1%80%D0%BC%D0%B0%D1%82%D0%B8%D0%BA%D0%B0)) для того же протокола, часто позволяет ОС указать номер порта для подключения к серверу.

Если [хост](https://ru.wikipedia.org/wiki/%D0%A5%D0%BE%D1%81%D1%82) получит пакет, порт процесса-отправителя называется «удалённым» ([англ.](https://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) *remote*) портом или "открытым на другом [хосте](https://ru.wikipedia.org/wiki/%D0%A5%D0%BE%D1%81%D1%82)", а порт процесса получателя — «локальным» портом, то есть открытым на текущем хосте. Если хост отправил пакет, порт процесса-отправителя называется «локальным» портом (открытым на текущем [хосте](https://ru.wikipedia.org/wiki/%D0%A5%D0%BE%D1%81%D1%82)), а порт процесса-получателя — «удалённым» портом (открытым на другом [хосте](https://ru.wikipedia.org/wiki/%D0%A5%D0%BE%D1%81%D1%82)).

Номера портов для [протоколов](https://ru.wikipedia.org/wiki/%D0%9F%D1%80%D0%BE%D1%82%D0%BE%D0%BA%D0%BE%D0%BB_%D0%BF%D0%B5%D1%80%D0%B5%D0%B4%D0%B0%D1%87%D0%B8_%D0%B4%D0%B0%D0%BD%D0%BD%D1%8B%D1%85) прикладного уровня модели [TCP/IP](https://ru.wikipedia.org/wiki/TCP/IP) ([HTTP](https://ru.wikipedia.org/wiki/HTTP), [SSH](https://ru.wikipedia.org/wiki/SSH) и др.) обычно назначаются организацией [IANA](https://ru.wikipedia.org/wiki/IANA) ([англ.](https://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) ***i****nternet****a****ssigned****n****umbers****a****uthority*). Однако на практике в целях безопасности номера портов могут выбираться произвольно.

Термин «порт» чаще всего применяется по отношению к [протоколам](https://ru.wikipedia.org/wiki/%D0%9F%D1%80%D0%BE%D1%82%D0%BE%D0%BA%D0%BE%D0%BB_%D0%BF%D0%B5%D1%80%D0%B5%D0%B4%D0%B0%D1%87%D0%B8_%D0%B4%D0%B0%D0%BD%D0%BD%D1%8B%D1%85) [TCP](https://ru.wikipedia.org/wiki/TCP) и [UDP](https://ru.wikipedia.org/wiki/UDP) ввиду популярности этих протоколов. В протоколах [SCTP](https://ru.wikipedia.org/wiki/SCTP) и [DCCP](https://ru.wikipedia.org/wiki/DCCP) используются номера, соответствующие понятию «номер порта» для протоколов TCP и UDP.

***16) Уровни OSI***

**Сетевая модель OSI** ( The **Open Systems Interconnection** model **)** — сетевая модель стека (магазина) сетевых протоколов OSI/ISO. Посредством данной модели различные сетевые устройства могут взаимодействовать друг с другом. Модель определяет различные уровни взаимодействия систем. Каждый уровень выполняет определённые функции при таком взаимодействии.

### Уровни модели OSI[[править](https://ru.wikipedia.org/w/index.php?title=%D0%A1%D0%B5%D1%82%D0%B5%D0%B2%D0%B0%D1%8F_%D0%BC%D0%BE%D0%B4%D0%B5%D0%BB%D1%8C_OSI&veaction=edit&section=5) | [править код](https://ru.wikipedia.org/w/index.php?title=%D0%A1%D0%B5%D1%82%D0%B5%D0%B2%D0%B0%D1%8F_%D0%BC%D0%BE%D0%B4%D0%B5%D0%BB%D1%8C_OSI&action=edit&section=5)]

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Модель** | | | | | |
| **Уровень (layer)** | | **Тип данных (PDU**[[15]](https://ru.wikipedia.org/wiki/%D0%A1%D0%B5%D1%82%D0%B5%D0%B2%D0%B0%D1%8F_%D0%BC%D0%BE%D0%B4%D0%B5%D0%BB%D1%8C_OSI#cite_note-15)**)** | **Функции** | **Примеры** | **Оборудование** |
| **Host layers** | 7. [Прикладной (application)](https://ru.wikipedia.org/wiki/%D0%9F%D1%80%D0%B8%D0%BA%D0%BB%D0%B0%D0%B4%D0%BD%D0%BE%D0%B9_%D1%83%D1%80%D0%BE%D0%B2%D0%B5%D0%BD%D1%8C) | Данные | Доступ к сетевым службам | [HTTP](https://ru.wikipedia.org/wiki/HTTP), [FTP](https://ru.wikipedia.org/wiki/FTP), [POP3](https://ru.wikipedia.org/wiki/POP3), [WebSocket](https://ru.wikipedia.org/wiki/WebSocket) | [Хосты](https://ru.wikipedia.org/wiki/%D0%A5%D0%BE%D1%81%D1%82) (клиенты сети),  [Межсетевой экран](https://ru.wikipedia.org/wiki/%D0%9C%D0%B5%D0%B6%D1%81%D0%B5%D1%82%D0%B5%D0%B2%D0%BE%D0%B9_%D1%8D%D0%BA%D1%80%D0%B0%D0%BD) |
| 6. [Представления (presentation)](https://ru.wikipedia.org/wiki/%D0%9F%D1%80%D0%B5%D0%B4%D1%81%D1%82%D0%B0%D0%B2%D0%B8%D1%82%D0%B5%D0%BB%D1%8C%D1%81%D0%BA%D0%B8%D0%B9_%D1%83%D1%80%D0%BE%D0%B2%D0%B5%D0%BD%D1%8C) | Представление и [шифрование](https://ru.wikipedia.org/wiki/%D0%A8%D0%B8%D1%84%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5) данных | [ASCII](https://ru.wikipedia.org/wiki/ASCII), [EBCDIC](https://ru.wikipedia.org/wiki/EBCDIC), [JPEG](https://ru.wikipedia.org/wiki/JPEG), [MIDI](https://ru.wikipedia.org/wiki/MIDI) |
| 5. [Сеансовый (session)](https://ru.wikipedia.org/wiki/%D0%A1%D0%B5%D0%B0%D0%BD%D1%81%D0%BE%D0%B2%D1%8B%D0%B9_%D1%83%D1%80%D0%BE%D0%B2%D0%B5%D0%BD%D1%8C) | Управление сеансом связи | [RPC](https://ru.wikipedia.org/wiki/Remote_Procedure_Call), [PAP](https://ru.wikipedia.org/wiki/Password_Authentication_Protocol), [L2TP](https://ru.wikipedia.org/wiki/L2TP), [gRPC](https://ru.wikipedia.org/wiki/GRPC) |
| 4. [Транспортный (transport)](https://ru.wikipedia.org/wiki/%D0%A2%D1%80%D0%B0%D0%BD%D1%81%D0%BF%D0%BE%D1%80%D1%82%D0%BD%D1%8B%D0%B9_%D1%83%D1%80%D0%BE%D0%B2%D0%B5%D0%BD%D1%8C) | [Сегменты](https://ru.wikipedia.org/wiki/%D0%A1%D0%B5%D0%B3%D0%BC%D0%B5%D0%BD%D1%82%D0%B0%D1%86%D0%B8%D1%8F_%D0%BF%D0%B0%D0%BA%D0%B5%D1%82%D0%BE%D0%B2)  (segment) /[Датаграммы](https://ru.wikipedia.org/wiki/%D0%94%D0%B0%D1%82%D0%B0%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B0) (datagram) | Прямая связь между конечными пунктами и надёжность | [TCP](https://ru.wikipedia.org/wiki/Transmission_Control_Protocol), [UDP](https://ru.wikipedia.org/wiki/User_Datagram_Protocol), [SCTP](https://ru.wikipedia.org/wiki/SCTP), [Порты](https://ru.wikipedia.org/wiki/%D0%9F%D0%BE%D1%80%D1%82_(%D0%BA%D0%BE%D0%BC%D0%BF%D1%8C%D1%8E%D1%82%D0%B5%D1%80%D0%BD%D1%8B%D0%B5_%D1%81%D0%B5%D1%82%D0%B8)) |
| **Media**[[16]](https://ru.wikipedia.org/wiki/%D0%A1%D0%B5%D1%82%D0%B5%D0%B2%D0%B0%D1%8F_%D0%BC%D0%BE%D0%B4%D0%B5%D0%BB%D1%8C_OSI#cite_note-16) **layers** | 3. [Сетевой (network)](https://ru.wikipedia.org/wiki/%D0%A1%D0%B5%D1%82%D0%B5%D0%B2%D0%BE%D0%B9_%D1%83%D1%80%D0%BE%D0%B2%D0%B5%D0%BD%D1%8C) | [Пакеты](https://ru.wikipedia.org/wiki/%D0%9F%D0%B0%D0%BA%D0%B5%D1%82_(%D1%81%D0%B5%D1%82%D0%B5%D0%B2%D1%8B%D0%B5_%D1%82%D0%B5%D1%85%D0%BD%D0%BE%D0%BB%D0%BE%D0%B3%D0%B8%D0%B8)) (packet) | Определение маршрута и логическая адресация | [IPv4](https://ru.wikipedia.org/wiki/IPv4), [IPv6](https://ru.wikipedia.org/wiki/IPv6), [IPsec](https://ru.wikipedia.org/wiki/IPsec), [AppleTalk](https://ru.wikipedia.org/wiki/AppleTalk), [ICMP](https://ru.wikipedia.org/wiki/ICMP) | [Маршрутизатор](https://ru.wikipedia.org/wiki/%D0%9C%D0%B0%D1%80%D1%88%D1%80%D1%83%D1%82%D0%B8%D0%B7%D0%B0%D1%82%D0%BE%D1%80), [Сетевой шлюз](https://ru.wikipedia.org/wiki/%D0%A1%D0%B5%D1%82%D0%B5%D0%B2%D0%BE%D0%B9_%D1%88%D0%BB%D1%8E%D0%B7),  [Межсетевой экран](https://ru.wikipedia.org/wiki/%D0%9C%D0%B5%D0%B6%D1%81%D0%B5%D1%82%D0%B5%D0%B2%D0%BE%D0%B9_%D1%8D%D0%BA%D1%80%D0%B0%D0%BD) |
| 2. [Канальный (data link)](https://ru.wikipedia.org/wiki/%D0%9A%D0%B0%D0%BD%D0%B0%D0%BB%D1%8C%D0%BD%D1%8B%D0%B9_%D1%83%D1%80%D0%BE%D0%B2%D0%B5%D0%BD%D1%8C) | Биты (bit)/ [Кадры](https://ru.wikipedia.org/wiki/%D0%9A%D0%B0%D0%B4%D1%80_(%D1%82%D0%B5%D0%BB%D0%B5%D0%BA%D0%BE%D0%BC%D0%BC%D1%83%D0%BD%D0%B8%D0%BA%D0%B0%D1%86%D0%B8%D0%B8)) (frame) | Физическая адресация | [PPP](https://ru.wikipedia.org/wiki/Point-to-Point_Protocol), [IEEE 802.22](https://ru.wikipedia.org/wiki/IEEE_802.22), [Ethernet](https://ru.wikipedia.org/wiki/Ethernet), [DSL](https://ru.wikipedia.org/wiki/XDSL), [ARP](https://ru.wikipedia.org/wiki/ARP), [сетевая карта](https://ru.wikipedia.org/wiki/%D0%A1%D0%B5%D1%82%D0%B5%D0%B2%D0%B0%D1%8F_%D0%BA%D0%B0%D1%80%D1%82%D0%B0). | [Сетевой мост](https://ru.wikipedia.org/wiki/%D0%A1%D0%B5%D1%82%D0%B5%D0%B2%D0%BE%D0%B9_%D0%BC%D0%BE%D1%81%D1%82), [Коммутатор](https://ru.wikipedia.org/wiki/%D0%9A%D0%BE%D0%BC%D0%BC%D1%83%D1%82%D0%B0%D1%82%D0%BE%D1%80_Ethernet),  [точка доступа](https://ru.wikipedia.org/wiki/%D0%91%D0%B5%D1%81%D0%BF%D1%80%D0%BE%D0%B2%D0%BE%D0%B4%D0%BD%D0%B0%D1%8F_%D1%82%D0%BE%D1%87%D0%BA%D0%B0_%D0%B4%D0%BE%D1%81%D1%82%D1%83%D0%BF%D0%B0) |
| 1. [Физический (physical)](https://ru.wikipedia.org/wiki/%D0%A4%D0%B8%D0%B7%D0%B8%D1%87%D0%B5%D1%81%D0%BA%D0%B8%D0%B9_%D1%81%D0%BB%D0%BE%D0%B9) | [Биты](https://ru.wikipedia.org/wiki/%D0%91%D0%B8%D1%82) (bit) | Работа со средой передачи, сигналами и двоичными данными | [USB](https://ru.wikipedia.org/wiki/USB), [RJ](https://ru.wikipedia.org/wiki/Registered_Jack) («витая пара», коаксиальный, оптоволоконный), радиоканал | [Концентратор](https://ru.wikipedia.org/wiki/%D0%A1%D0%B5%D1%82%D0%B5%D0%B2%D0%BE%D0%B9_%D0%BA%D0%BE%D0%BD%D1%86%D0%B5%D0%BD%D1%82%D1%80%D0%B0%D1%82%D0%BE%D1%80),  [Повторитель (сетевое оборудование)](https://ru.wikipedia.org/wiki/%D0%9F%D0%BE%D0%B2%D1%82%D0%BE%D1%80%D0%B8%D1%82%D0%B5%D0%BB%D1%8C_(%D1%81%D0%B5%D1%82%D0%B5%D0%B2%D0%BE%D0%B5_%D0%BE%D0%B1%D0%BE%D1%80%D1%83%D0%B4%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5)) |

В литературе наиболее часто принято начинать описание уровней модели OSI с 7-го уровня, называемого прикладным, на котором пользовательские приложения обращаются к сети. Модель OSI заканчивается 1-м уровнем — физическим, на котором определены стандарты, предъявляемые независимыми производителями к средам передачи данных:

* тип передающей среды (медный кабель, оптоволокно, радиоэфир и др.),
* тип модуляции сигнала,
* сигнальные уровни логических дискретных состояний (нули и единицы).

Любой протокол модели OSI должен взаимодействовать либо с протоколами своего уровня, либо с протоколами на единицу выше и/или ниже своего уровня. Взаимодействия с протоколами своего уровня называются горизонтальными, а с уровнями на единицу выше или ниже — вертикальными. Любой протокол модели OSI может выполнять только функции своего уровня и не может выполнять функций другого уровня, что не выполняется в протоколах альтернативных моделей.

Каждому уровню с некоторой долей условности соответствует свой операнд — логически неделимый элемент данных, которым на отдельном уровне можно оперировать в рамках модели и используемых протоколов: на физическом уровне мельчайшая единица — бит, на канальном уровне информация объединена в кадры, на сетевом — в пакеты , на транспортном — в сегменты, датаграммы. Любой фрагмент данных, логически объединённых для передачи — кадр, пакет, датаграмма — считается сообщением. Именно сообщения в общем виде являются операндами сеансового, представления и прикладного уровней.

К базовым сетевым технологиям относятся физический и канальный уровни.

#### Прикладной уровень[[править](https://ru.wikipedia.org/w/index.php?title=%D0%A1%D0%B5%D1%82%D0%B5%D0%B2%D0%B0%D1%8F_%D0%BC%D0%BE%D0%B4%D0%B5%D0%BB%D1%8C_OSI&veaction=edit&section=6) | [править код](https://ru.wikipedia.org/w/index.php?title=%D0%A1%D0%B5%D1%82%D0%B5%D0%B2%D0%B0%D1%8F_%D0%BC%D0%BE%D0%B4%D0%B5%D0%BB%D1%8C_OSI&action=edit&section=6)]

*Основная статья:*[***Прикладной уровень***](https://ru.wikipedia.org/wiki/%D0%9F%D1%80%D0%B8%D0%BA%D0%BB%D0%B0%D0%B4%D0%BD%D0%BE%D0%B9_%D1%83%D1%80%D0%BE%D0%B2%D0%B5%D0%BD%D1%8C)

Прикладной уровень (уровень приложений; [англ.](https://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) *application layer*) — верхний уровень модели, обеспечивающий взаимодействие пользовательских приложений с сетью:

* позволяет приложениям использовать сетевые службы:
  + удалённый доступ к файлам и базам данных,
  + пересылка электронной почты;
* отвечает за передачу служебной информации;
* предоставляет приложениям информацию об ошибках;
* формирует запросы к уровню представления.

Протоколы прикладного уровня: [RDP](https://ru.wikipedia.org/wiki/Remote_Desktop_Protocol), [HTTP](https://ru.wikipedia.org/wiki/HTTP), [SMTP](https://ru.wikipedia.org/wiki/SMTP), [SNMP](https://ru.wikipedia.org/wiki/SNMP), [POP3](https://ru.wikipedia.org/wiki/Post_Office_Protocol), [FTP](https://ru.wikipedia.org/wiki/FTP), [XMPP](https://ru.wikipedia.org/wiki/XMPP), [OSCAR](https://ru.wikipedia.org/wiki/OSCAR), [Modbus](https://ru.wikipedia.org/wiki/Modbus), [SIP](https://ru.wikipedia.org/wiki/SIP), [TELNET](https://ru.wikipedia.org/wiki/TELNET) и другие.

Определения протокола прикладного уровня и уровня представления очень размыты, и принадлежность протокола к тому или иному уровню, например протокола HTTPS зависит от конечного сервиса который предоставляет приложение.

В том случае если протокол, например HTTPS, используется для просмотра некоей простой интернет страницы через браузер — его можно рассматривать как протокол прикладного уровня. В том же случае если протокол HTTPS используется как низкоуровневый протокол для передачи финансовой информации например по протоколу ISO 8583, то протокол HTTPS будет являться протоколом уровня представления, а протокол ISO 8583 — будет протоколом уровня приложения. То же касается иных протоколов прикладного уровня[[*источник не указан 385 дней*](https://ru.wikipedia.org/wiki/%D0%92%D0%B8%D0%BA%D0%B8%D0%BF%D0%B5%D0%B4%D0%B8%D1%8F:%D0%A1%D1%81%D1%8B%D0%BB%D0%BA%D0%B8_%D0%BD%D0%B0_%D0%B8%D1%81%D1%82%D0%BE%D1%87%D0%BD%D0%B8%D0%BA%D0%B8)].

#### Уровень представления[[править](https://ru.wikipedia.org/w/index.php?title=%D0%A1%D0%B5%D1%82%D0%B5%D0%B2%D0%B0%D1%8F_%D0%BC%D0%BE%D0%B4%D0%B5%D0%BB%D1%8C_OSI&veaction=edit&section=7) | [править код](https://ru.wikipedia.org/w/index.php?title=%D0%A1%D0%B5%D1%82%D0%B5%D0%B2%D0%B0%D1%8F_%D0%BC%D0%BE%D0%B4%D0%B5%D0%BB%D1%8C_OSI&action=edit&section=7)]

*Основная статья:*[***Уровень представления***](https://ru.wikipedia.org/wiki/%D0%A3%D1%80%D0%BE%D0%B2%D0%B5%D0%BD%D1%8C_%D0%BF%D1%80%D0%B5%D0%B4%D1%81%D1%82%D0%B0%D0%B2%D0%BB%D0%B5%D0%BD%D0%B8%D1%8F)

Уровень представления ([англ.](https://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) *presentation layer*) обеспечивает преобразование протоколов и кодирование/декодирование данных. Запросы приложений, полученные с прикладного уровня, на уровне представления преобразуются в формат для передачи по сети, а полученные из сети данные преобразуются в формат приложений. На этом уровне может осуществляться сжатие/распаковка или шифрование/дешифрование, а также перенаправление запросов другому сетевому ресурсу, если они не могут быть обработаны локально.

Уровень представлений обычно представляет собой промежуточный протокол для преобразования информации из соседних уровней. Это позволяет осуществлять обмен между приложениями на разнородных компьютерных системах прозрачным для приложений образом. Уровень представлений обеспечивает форматирование и преобразование кода. Форматирование кода используется для того, чтобы гарантировать приложению поступление информации для обработки, которая имела бы для него смысл. При необходимости этот уровень может выполнять перевод из одного формата данных в другой.

Уровень представлений имеет дело не только с форматами и представлением данных, он также занимается структурами данных, которые используются программами. Таким образом, уровень 6 обеспечивает организацию данных при их пересылке.

Чтобы понять, как это работает, представим, что имеются две системы. Одна использует для представления данных расширенный двоичный код обмена информацией [EBCDIC](https://ru.wikipedia.org/wiki/EBCDIC), например, это может быть [мейнфрейм](https://ru.wikipedia.org/wiki/%D0%9C%D0%B5%D0%B9%D0%BD%D1%84%D1%80%D0%B5%D0%B9%D0%BC) компании [IBM](https://ru.wikipedia.org/wiki/IBM), а другая — американский стандартный код обмена информацией [ASCII](https://ru.wikipedia.org/wiki/ASCII) (его использует большинство других производителей компьютеров). Если этим двум системам необходимо обменяться информацией, то нужен уровень представлений, который выполнит преобразование и осуществит перевод между двумя различными форматами.

Другой функцией, выполняемой на уровне представлений, является шифрование данных, которое применяется в тех случаях, когда необходимо защитить передаваемую информацию от доступа несанкционированными получателями. Чтобы решить эту задачу, процессы и коды, находящиеся на уровне представлений, должны выполнить преобразование данных. На этом уровне существуют и другие подпрограммы, которые сжимают тексты и преобразовывают графические изображения в битовые потоки, так, что они могут передаваться по сети.

Стандарты уровня представлений также определяют способы представления графических изображений. Для этих целей может использоваться формат [PICT](https://ru.wikipedia.org/w/index.php?title=PICT&action=edit&redlink=1) — формат изображений, применяемый для передачи графики QuickDraw между программами.

Другим форматом представлений является тэгированный формат файлов изображений [TIFF](https://ru.wikipedia.org/wiki/TIFF), который обычно используется для растровых изображений с высоким [разрешением](https://ru.wikipedia.org/wiki/%D0%A0%D0%B0%D0%B7%D1%80%D0%B5%D1%88%D0%B5%D0%BD%D0%B8%D0%B5_(%D0%BA%D0%BE%D0%BC%D0%BF%D1%8C%D1%8E%D1%82%D0%B5%D1%80%D0%BD%D0%B0%D1%8F_%D0%B3%D1%80%D0%B0%D1%84%D0%B8%D0%BA%D0%B0)). Следующим стандартом уровня представлений, который может использоваться для графических изображений, является стандарт, разработанный Объединённой экспертной группой по фотографии (Joint Photographic Expert Group); в повседневном пользовании этот стандарт называют просто [JPEG](https://ru.wikipedia.org/wiki/JPEG).

Существует другая группа стандартов уровня представлений, которая определяет представление звука и кинофрагментов. Сюда входят интерфейс электронных музыкальных инструментов ([англ.](https://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) *Musical Instrument Digital Interface*, [MIDI](https://ru.wikipedia.org/wiki/MIDI)) для цифрового представления музыки, разработанный Экспертной группой по кинематографии стандарт [MPEG](https://ru.wikipedia.org/wiki/MPEG), используемый для сжатия и кодирования видеороликов на компакт-дисках, хранения в оцифрованном виде и передачи со скоростями до 1,5 Мбит/с, и [QuickTime](https://ru.wikipedia.org/wiki/QuickTime) — стандарт, описывающий звуковые и видео элементы для программ, выполняемых на компьютерах Macintosh и PowerPC.

Протоколы уровня представления: AFP — [Apple Filing Protocol](https://ru.wikipedia.org/wiki/Apple_Filing_Protocol), ICA — [Independent Computing Architecture](https://ru.wikipedia.org/wiki/Independent_Computing_Architecture), LPP — Lightweight Presentation Protocol, NCP — [NetWare Core Protocol](https://ru.wikipedia.org/wiki/NetWare_Core_Protocol), NDR — [Network Data Representation](https://ru.wikipedia.org/w/index.php?title=Network_Data_Representation&action=edit&redlink=1), XDR — [eXternal Data Representation](https://ru.wikipedia.org/wiki/External_Data_Representation), X.25 PAD — [Packet Assembler/Disassembler Protocol](https://ru.wikipedia.org/wiki/X.25).

#### Сеансовый уровень[[править](https://ru.wikipedia.org/w/index.php?title=%D0%A1%D0%B5%D1%82%D0%B5%D0%B2%D0%B0%D1%8F_%D0%BC%D0%BE%D0%B4%D0%B5%D0%BB%D1%8C_OSI&veaction=edit&section=8) | [править код](https://ru.wikipedia.org/w/index.php?title=%D0%A1%D0%B5%D1%82%D0%B5%D0%B2%D0%B0%D1%8F_%D0%BC%D0%BE%D0%B4%D0%B5%D0%BB%D1%8C_OSI&action=edit&section=8)]

*Основная статья:*[***Сеансовый уровень***](https://ru.wikipedia.org/wiki/%D0%A1%D0%B5%D0%B0%D0%BD%D1%81%D0%BE%D0%B2%D1%8B%D0%B9_%D1%83%D1%80%D0%BE%D0%B2%D0%B5%D0%BD%D1%8C)

Сеансовый уровень ([англ.](https://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) *session layer*) модели обеспечивает поддержание сеанса связи, позволяя приложениям взаимодействовать между собой длительное время. Уровень управляет созданием/завершением сеанса, обменом информацией, синхронизацией задач, определением права на передачу данных и поддержанием сеанса в периоды неактивности приложений.

Протоколы сеансового уровня: H.245 ([Call Control Protocol for Multimedia Communication](https://ru.wikipedia.org/wiki/H.245)), ISO-SP (OSI Session Layer Protocol (X.225, ISO 8327)), iSNS ([Internet Storage Name Service](https://ru.wikipedia.org/wiki/Internet_Storage_Name_Service)), L2F ([Layer 2 Forwarding Protocol](https://ru.wikipedia.org/wiki/Layer_2_Forwarding_Protocol)), L2TP ([Layer 2 Tunneling Protocol](https://ru.wikipedia.org/wiki/L2TP)), NetBIOS ([Network Basic Input Output System](https://ru.wikipedia.org/wiki/NetBIOS)), PAP ([Password Authentication Protocol](https://ru.wikipedia.org/wiki/Password_Authentication_Protocol)), PPTP ([Point-to-Point Tunneling Protocol](https://ru.wikipedia.org/wiki/PPTP)), RPC ([Remote Procedure Call Protocol](https://ru.wikipedia.org/wiki/Remote_procedure_call)), RTCP ([Real-time Transport Control Protocol](https://ru.wikipedia.org/wiki/RTCP)), SMPP ([Short Message Peer-to-Peer](https://ru.wikipedia.org/wiki/SMPP)), SCP ([Session Control Protocol](https://ru.wikipedia.org/w/index.php?title=Session_Control_Protocol&action=edit&redlink=1)), ZIP ([Zone Information Protocol](https://ru.wikipedia.org/wiki/Zone_Information_Protocol)), SDP ([Sockets Direct Protocol](https://ru.wikipedia.org/wiki/Sockets_Direct_Protocol))…

#### Транспортный уровень[[править](https://ru.wikipedia.org/w/index.php?title=%D0%A1%D0%B5%D1%82%D0%B5%D0%B2%D0%B0%D1%8F_%D0%BC%D0%BE%D0%B4%D0%B5%D0%BB%D1%8C_OSI&veaction=edit&section=9) | [править код](https://ru.wikipedia.org/w/index.php?title=%D0%A1%D0%B5%D1%82%D0%B5%D0%B2%D0%B0%D1%8F_%D0%BC%D0%BE%D0%B4%D0%B5%D0%BB%D1%8C_OSI&action=edit&section=9)]

*Основная статья:*[***Транспортный уровень***](https://ru.wikipedia.org/wiki/%D0%A2%D1%80%D0%B0%D0%BD%D1%81%D0%BF%D0%BE%D1%80%D1%82%D0%BD%D1%8B%D0%B9_%D1%83%D1%80%D0%BE%D0%B2%D0%B5%D0%BD%D1%8C)

Транспортный уровень ([англ.](https://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) *transport layer*) модели предназначен для обеспечения надёжной передачи данных от отправителя к получателю. При этом уровень надёжности может варьироваться в широких пределах. Существует множество классов протоколов транспортного уровня, начиная от протоколов, предоставляющих только основные транспортные функции (например, функции передачи данных без подтверждения приёма), и заканчивая протоколами, которые гарантируют доставку в пункт назначения нескольких пакетов данных в надлежащей последовательности, мультиплексируют несколько потоков данных, обеспечивают механизм управления потоками данных и гарантируют достоверность принятых данных. Например, [UDP](https://ru.wikipedia.org/wiki/UDP) ограничивается контролем целостности данных в рамках одной датаграммы и не исключает возможности потери пакета целиком или дублирования пакетов, нарушение порядка получения пакетов данных; [TCP](https://ru.wikipedia.org/wiki/TCP) обеспечивает надёжную непрерывную передачу данных, исключающую потерю данных или нарушение порядка их поступления или дублирования, может перераспределять данные, разбивая большие порции данных на фрагменты и наоборот, склеивая фрагменты в один пакет.

Протоколы транспортного уровня: ATP ([AppleTalk Transaction Protocol](https://ru.wikipedia.org/wiki/AppleTalk)), CUDP ([Cyclic UDP](https://ru.wikipedia.org/w/index.php?title=Cyclic_UDP&action=edit&redlink=1)), DCCP ([Datagram Congestion Control Protocol](https://ru.wikipedia.org/wiki/Datagram_Congestion_Control_Protocol)), FCP ([Fibre Channel Protocol](https://ru.wikipedia.org/wiki/Fibre_Channel)), IL ([IL Protocol](https://ru.wikipedia.org/wiki/IL_Protocol)), NBF ([NetBIOS Frames protocol](https://ru.wikipedia.org/wiki/NetBIOS_Frame_Protocol)), NCP ([NetWare Core Protocol](https://ru.wikipedia.org/wiki/NetWare_Core_Protocol)), SCTP ([Stream Control Transmission Protocol](https://ru.wikipedia.org/wiki/Stream_Control_Transmission_Protocol)), SPX ([Sequenced Packet Exchange](https://ru.wikipedia.org/wiki/IPX/SPX)), SST ([Structured Stream Transport](https://ru.wikipedia.org/wiki/Structured_Stream_Transport)), TCP ([Transmission Control Protocol](https://ru.wikipedia.org/wiki/Transmission_Control_Protocol)), UDP ([User Datagram Protocol](https://ru.wikipedia.org/wiki/User_Datagram_Protocol)).

#### Сетевой уровень[[править](https://ru.wikipedia.org/w/index.php?title=%D0%A1%D0%B5%D1%82%D0%B5%D0%B2%D0%B0%D1%8F_%D0%BC%D0%BE%D0%B4%D0%B5%D0%BB%D1%8C_OSI&veaction=edit&section=10) | [править код](https://ru.wikipedia.org/w/index.php?title=%D0%A1%D0%B5%D1%82%D0%B5%D0%B2%D0%B0%D1%8F_%D0%BC%D0%BE%D0%B4%D0%B5%D0%BB%D1%8C_OSI&action=edit&section=10)]

*Основная статья:*[***Сетевой уровень***](https://ru.wikipedia.org/wiki/%D0%A1%D0%B5%D1%82%D0%B5%D0%B2%D0%BE%D0%B9_%D1%83%D1%80%D0%BE%D0%B2%D0%B5%D0%BD%D1%8C)

Сетевой уровень ([англ.](https://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) *network layer*) модели предназначен для определения пути передачи данных. Отвечает за трансляцию логических адресов и имён в физические, определение кратчайших маршрутов, коммутацию и маршрутизацию, отслеживание неполадок и «заторов» в сети.

Протоколы сетевого уровня маршрутизируют данные от источника к получателю. Работающие на этом уровне устройства ([маршрутизаторы](https://ru.wikipedia.org/wiki/%D0%9C%D0%B0%D1%80%D1%88%D1%80%D1%83%D1%82%D0%B8%D0%B7%D0%B0%D1%82%D0%BE%D1%80)) условно называют устройствами третьего уровня (по номеру уровня в модели OSI).

Протоколы сетевого уровня: IP/IPv4/IPv6 ([Internet Protocol](https://ru.wikipedia.org/wiki/Internet_Protocol)), IPX ([Internetwork Packet Exchange](https://ru.wikipedia.org/wiki/Internetwork_Packet_Exchange), протокол межсетевого обмена), X.25 (частично этот протокол реализован на уровне 2), CLNP (сетевой протокол без организации соединений), IPsec ([Internet Protocol Security](https://ru.wikipedia.org/wiki/Internet_Protocol_Security)).

Протоколы маршрутизации — RIP ([Routing Information Protocol](https://ru.wikipedia.org/wiki/Routing_Information_Protocol)), OSPF ([Open Shortest Path First](https://ru.wikipedia.org/wiki/Open_Shortest_Path_First)).

#### Канальный уровень[[править](https://ru.wikipedia.org/w/index.php?title=%D0%A1%D0%B5%D1%82%D0%B5%D0%B2%D0%B0%D1%8F_%D0%BC%D0%BE%D0%B4%D0%B5%D0%BB%D1%8C_OSI&veaction=edit&section=11) | [править код](https://ru.wikipedia.org/w/index.php?title=%D0%A1%D0%B5%D1%82%D0%B5%D0%B2%D0%B0%D1%8F_%D0%BC%D0%BE%D0%B4%D0%B5%D0%BB%D1%8C_OSI&action=edit&section=11)]

*Основная статья:*[***Канальный уровень***](https://ru.wikipedia.org/wiki/%D0%9A%D0%B0%D0%BD%D0%B0%D0%BB%D1%8C%D0%BD%D1%8B%D0%B9_%D1%83%D1%80%D0%BE%D0%B2%D0%B5%D0%BD%D1%8C)

Канальный уровень ([англ.](https://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) *data link layer*) предназначен для обеспечения взаимодействия сетей на физическом уровне и контроля ошибок, которые могут возникнуть. Полученные с физического уровня данные, представленные в битах, он упаковывает в [кадры](https://ru.wikipedia.org/wiki/%D0%9A%D0%B0%D0%B4%D1%80_(%D1%82%D0%B5%D0%BB%D0%B5%D0%BA%D0%BE%D0%BC%D0%BC%D1%83%D0%BD%D0%B8%D0%BA%D0%B0%D1%86%D0%B8%D0%B8)), проверяет их на целостность и, если нужно, исправляет ошибки (либо формирует повторный запрос повреждённого кадра) и отправляет на сетевой уровень. Канальный уровень может взаимодействовать с одним или несколькими физическими уровнями, контролируя и управляя этим взаимодействием.

Спецификация [IEEE 802](https://ru.wikipedia.org/wiki/IEEE_802) разделяет этот уровень на два подуровня: [MAC](https://ru.wikipedia.org/wiki/Media_Access_Control) ([англ.](https://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) *media access control*) регулирует доступ к разделяемой физической среде, [LLC](https://ru.wikipedia.org/wiki/Logical_link_control) ([англ.](https://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) *logical link control*) обеспечивает обслуживание сетевого уровня.

На этом уровне работают [коммутаторы](https://ru.wikipedia.org/wiki/%D0%A1%D0%B5%D1%82%D0%B5%D0%B2%D0%BE%D0%B9_%D0%BA%D0%BE%D0%BC%D0%BC%D1%83%D1%82%D0%B0%D1%82%D0%BE%D1%80), [мосты](https://ru.wikipedia.org/wiki/%D0%A1%D0%B5%D1%82%D0%B5%D0%B2%D0%BE%D0%B9_%D0%BC%D0%BE%D1%81%D1%82) и другие устройства. Эти устройства используют адресацию второго уровня (по номеру уровня в модели OSI).

Протоколы канального уровня: [ARCnet](https://ru.wikipedia.org/wiki/ARCnet), [ATM](https://ru.wikipedia.org/wiki/Asynchronous_Transfer_Mode), [Controller Area Network](https://ru.wikipedia.org/wiki/Controller_Area_Network) (CAN), [Econet](https://ru.wikipedia.org/wiki/Econet), [IEEE 802.3](https://ru.wikipedia.org/wiki/IEEE_802.3) ([Ethernet](https://ru.wikipedia.org/wiki/Ethernet)), [Ethernet Automatic Protection Switching](https://ru.wikipedia.org/wiki/Ethernet_Automatic_Protection_Switching) (EAPS), [Fiber Distributed Data Interface](https://ru.wikipedia.org/wiki/Fiber_Distributed_Data_Interface) (FDDI), [Frame Relay](https://ru.wikipedia.org/wiki/Frame_Relay), [High-Level Data Link Control](https://ru.wikipedia.org/wiki/High-Level_Data_Link_Control) (HDLC), [IEEE 802.2](https://ru.wikipedia.org/wiki/IEEE_802.2) (предоставляет функции LLC для подуровня IEEE 802 MAC), [Link Access Procedures, D channel](https://ru.wikipedia.org/w/index.php?title=Link_Access_Procedures,_D_channel&action=edit&redlink=1) (LAPD), [IEEE 802.11](https://ru.wikipedia.org/wiki/IEEE_802.11) [wireless LAN](https://ru.wikipedia.org/wiki/Wireless_LAN), [LocalTalk](https://ru.wikipedia.org/wiki/LocalTalk), [Multiprotocol Label Switching](https://ru.wikipedia.org/wiki/Multiprotocol_Label_Switching) (MPLS), [Point-to-Point Protocol](https://ru.wikipedia.org/wiki/Point-to-Point_Protocol) (PPP), [Point-to-Point Protocol over Ethernet](https://ru.wikipedia.org/wiki/PPPoE) (PPPoE), [Serial Line Internet Protocol](https://ru.wikipedia.org/wiki/SLIP) (SLIP, устарел), [StarLan](https://ru.wikipedia.org/wiki/StarLan), [Token ring](https://ru.wikipedia.org/wiki/Token_ring), [Unidirectional Link Detection](https://ru.wikipedia.org/w/index.php?title=Unidirectional_Link_Detection&action=edit&redlink=1)[[en]](https://en.wikipedia.org/wiki/Unidirectional_Link_Detection) (UDLD), [x.25](https://ru.wikipedia.org/wiki/X.25), [ARP](https://ru.wikipedia.org/wiki/ARP).

При разработке стеков протоколов на этом уровне решаются задачи помехоустойчивого кодирования. К таким способам кодирования относится [код Хемминга](https://ru.wikipedia.org/wiki/%D0%9A%D0%BE%D0%B4_%D0%A5%D1%8D%D0%BC%D0%BC%D0%B8%D0%BD%D0%B3%D0%B0), блочное кодирование, [код Рида — Соломона](https://ru.wikipedia.org/wiki/%D0%9A%D0%BE%D0%B4_%D0%A0%D0%B8%D0%B4%D0%B0_%E2%80%94_%D0%A1%D0%BE%D0%BB%D0%BE%D0%BC%D0%BE%D0%BD%D0%B0).

В программировании этот уровень представляет [драйвер](https://ru.wikipedia.org/wiki/%D0%94%D1%80%D0%B0%D0%B9%D0%B2%D0%B5%D1%80) сетевой платы, в [операционных системах](https://ru.wikipedia.org/wiki/%D0%9E%D0%BF%D0%B5%D1%80%D0%B0%D1%86%D0%B8%D0%BE%D0%BD%D0%BD%D0%B0%D1%8F_%D1%81%D0%B8%D1%81%D1%82%D0%B5%D0%BC%D0%B0) имеется программный интерфейс взаимодействия канального и сетевого уровней между собой. Это не новый уровень, а просто реализация модели для конкретной ОС. Примеры таких интерфейсов: [ODI](https://ru.wikipedia.org/w/index.php?title=ODI_(%D0%B8%D0%BD%D1%82%D0%B5%D1%80%D1%84%D0%B5%D0%B9%D1%81)&action=edit&redlink=1) ([англ.](https://en.wikipedia.org/wiki/Open_Data-Link_Interface)), [NDIS](https://ru.wikipedia.org/wiki/NDIS" \o "NDIS), [UDI](https://ru.wikipedia.org/w/index.php?title=UDI&action=edit&redlink=1).

#### Физический уровень[[править](https://ru.wikipedia.org/w/index.php?title=%D0%A1%D0%B5%D1%82%D0%B5%D0%B2%D0%B0%D1%8F_%D0%BC%D0%BE%D0%B4%D0%B5%D0%BB%D1%8C_OSI&veaction=edit&section=12) | [править код](https://ru.wikipedia.org/w/index.php?title=%D0%A1%D0%B5%D1%82%D0%B5%D0%B2%D0%B0%D1%8F_%D0%BC%D0%BE%D0%B4%D0%B5%D0%BB%D1%8C_OSI&action=edit&section=12)]

*Основная статья:*[***Физический уровень***](https://ru.wikipedia.org/wiki/%D0%A4%D0%B8%D0%B7%D0%B8%D1%87%D0%B5%D1%81%D0%BA%D0%B8%D0%B9_%D1%83%D1%80%D0%BE%D0%B2%D0%B5%D0%BD%D1%8C)

Физический уровень ([англ.](https://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) *physical layer*) — нижний уровень модели, который определяет метод передачи данных, представленных в двоичном виде, от одного устройства (компьютера) к другому. Составлением таких методов занимаются разные организации, в том числе: [Институт инженеров по электротехнике и электронике](https://ru.wikipedia.org/wiki/%D0%98%D0%BD%D1%81%D1%82%D0%B8%D1%82%D1%83%D1%82_%D0%B8%D0%BD%D0%B6%D0%B5%D0%BD%D0%B5%D1%80%D0%BE%D0%B2_%D1%8D%D0%BB%D0%B5%D0%BA%D1%82%D1%80%D0%BE%D1%82%D0%B5%D1%85%D0%BD%D0%B8%D0%BA%D0%B8_%D0%B8_%D1%8D%D0%BB%D0%B5%D0%BA%D1%82%D1%80%D0%BE%D0%BD%D0%B8%D0%BA%D0%B8), [Альянс электронной промышленности](https://ru.wikipedia.org/wiki/Electronic_Industries_Alliance), [Европейский институт телекоммуникационных стандартов](https://ru.wikipedia.org/wiki/%D0%95%D0%B2%D1%80%D0%BE%D0%BF%D0%B5%D0%B9%D1%81%D0%BA%D0%B8%D0%B9_%D0%B8%D0%BD%D1%81%D1%82%D0%B8%D1%82%D1%83%D1%82_%D1%82%D0%B5%D0%BB%D0%B5%D0%BA%D0%BE%D0%BC%D0%BC%D1%83%D0%BD%D0%B8%D0%BA%D0%B0%D1%86%D0%B8%D0%BE%D0%BD%D0%BD%D1%8B%D1%85_%D1%81%D1%82%D0%B0%D0%BD%D0%B4%D0%B0%D1%80%D1%82%D0%BE%D0%B2) и другие. Осуществляют передачу электрических или оптических сигналов в кабель или в радиоэфир и, соответственно, их приём и преобразование в биты данных в соответствии с [методами кодирования цифровых сигналов](https://ru.wikipedia.org/wiki/%D0%A4%D0%B8%D0%B7%D0%B8%D1%87%D0%B5%D1%81%D0%BA%D0%BE%D0%B5_%D0%BA%D0%BE%D0%B4%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5).

На этом уровне также работают [концентраторы](https://ru.wikipedia.org/wiki/%D0%A1%D0%B5%D1%82%D0%B5%D0%B2%D0%BE%D0%B9_%D0%BA%D0%BE%D0%BD%D1%86%D0%B5%D0%BD%D1%82%D1%80%D0%B0%D1%82%D0%BE%D1%80), [повторители](https://ru.wikipedia.org/wiki/%D0%9F%D0%BE%D0%B2%D1%82%D0%BE%D1%80%D0%B8%D1%82%D0%B5%D0%BB%D1%8C_(%D1%81%D0%B5%D1%82%D0%B5%D0%B2%D0%BE%D0%B5_%D0%BE%D0%B1%D0%BE%D1%80%D1%83%D0%B4%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5)) сигнала и [медиаконвертеры](https://ru.wikipedia.org/wiki/%D0%9C%D0%B5%D0%B4%D0%B8%D0%B0%D0%BA%D0%BE%D0%BD%D0%B2%D0%B5%D1%80%D1%82%D0%B5%D1%80).

Функции физического уровня реализуются на всех устройствах, подключенных к сети. Со стороны компьютера функции физического уровня выполняются сетевым адаптером или последовательным портом. К физическому уровню относятся физические, электрические и механические интерфейсы между двумя системами. Физический уровень определяет такие виды сред передачи данных как [оптоволокно](https://ru.wikipedia.org/wiki/%D0%9E%D0%BF%D1%82%D0%B8%D1%87%D0%B5%D1%81%D0%BA%D0%BE%D0%B5_%D0%B2%D0%BE%D0%BB%D0%BE%D0%BA%D0%BD%D0%BE), [витая пара](https://ru.wikipedia.org/wiki/%D0%92%D0%B8%D1%82%D0%B0%D1%8F_%D0%BF%D0%B0%D1%80%D0%B0), [коаксиальный кабель](https://ru.wikipedia.org/wiki/%D0%9A%D0%BE%D0%B0%D0%BA%D1%81%D0%B8%D0%B0%D0%BB%D1%8C%D0%BD%D1%8B%D0%B9_%D0%BA%D0%B0%D0%B1%D0%B5%D0%BB%D1%8C), спутниковый канал передач данных и т. п. Стандартными типами сетевых интерфейсов, относящимися к физическому уровню, являются: [V.35](https://ru.wikipedia.org/wiki/V.35), [RS-232](https://ru.wikipedia.org/wiki/RS-232), [RS-485](https://ru.wikipedia.org/wiki/RS-485), [RJ-11](https://ru.wikipedia.org/wiki/RJ-45), [RJ-45](https://ru.wikipedia.org/wiki/RJ-45), разъёмы [AUI](https://ru.wikipedia.org/wiki/AUI) и [BNC](https://ru.wikipedia.org/wiki/BNC-%D0%BA%D0%BE%D0%BD%D0%BD%D0%B5%D0%BA%D1%82%D0%BE%D1%80).

При разработке стеков протоколов на этом уровне решаются задачи синхронизации и линейного кодирования. К таким способам кодирования относится [код NRZ](https://ru.wikipedia.org/wiki/NRZ_(%D0%BF%D1%80%D1%8F%D0%BC%D0%BE%D0%B9)), [код RZ](https://ru.wikipedia.org/wiki/RZ), [MLT-3](https://ru.wikipedia.org/wiki/MLT-3), [PAM5](https://ru.wikipedia.org/wiki/%D0%9F%D0%BE%D1%82%D0%B5%D0%BD%D1%86%D0%B8%D0%B0%D0%BB%D1%8C%D0%BD%D1%8B%D0%B9_%D0%BA%D0%BE%D0%B4_2B1Q), [Манчестер II](https://ru.wikipedia.org/wiki/%D0%9C%D0%B0%D0%BD%D1%87%D0%B5%D1%81%D1%82%D0%B5%D1%80%D1%81%D0%BA%D0%B8%D0%B9_%D0%BA%D0%BE%D0%B4).

Протоколы физического уровня: [IEEE 802.15 (Bluetooth)](https://ru.wikipedia.org/wiki/Bluetooth), [IRDA](https://ru.wikipedia.org/wiki/Infrared_Data_Association), [EIA](https://ru.wikipedia.org/wiki/Electronic_Industries_Alliance) [RS-232](https://ru.wikipedia.org/wiki/RS-232), [EIA-422](https://ru.wikipedia.org/wiki/EIA-422), [EIA-423](https://ru.wikipedia.org/wiki/RS-423), [RS-449](https://ru.wikipedia.org/wiki/RS-449), [RS-485](https://ru.wikipedia.org/wiki/RS-485), [DSL](https://ru.wikipedia.org/wiki/Digital_subscriber_line), [ISDN](https://ru.wikipedia.org/wiki/Integrated_Services_Digital_Network), [SONET/SDH](https://ru.wikipedia.org/wiki/Synchronous_optical_network), [802.11](https://ru.wikipedia.org/wiki/802.11) [Wi-Fi](https://ru.wikipedia.org/wiki/Wi-Fi), [Etherloop](https://ru.wikipedia.org/wiki/Etherloop), [GSM](https://ru.wikipedia.org/wiki/GSM) [Um radio interface](https://ru.wikipedia.org/w/index.php?title=Um_Interface&action=edit&redlink=1), [ITU](https://ru.wikipedia.org/wiki/International_Telecommunication_Union) и [ITU-T](https://ru.wikipedia.org/wiki/ITU-T), [TransferJet](https://ru.wikipedia.org/w/index.php?title=TransferJet&action=edit&redlink=1)[[en]](https://en.wikipedia.org/wiki/TransferJet), [ARINC 818](https://ru.wikipedia.org/w/index.php?title=ARINC_818&action=edit&redlink=1), [G.hn](https://ru.wikipedia.org/w/index.php?title=G.hn&action=edit&redlink=1)/[G.9960](https://ru.wikipedia.org/w/index.php?title=G.9960&action=edit&redlink=1), [Modbus Plus](https://ru.wikipedia.org/w/index.php?title=Modbus_Plus&action=edit&redlink=1).

***17) Хедеры http запросив***

**Заголовки HTTP**позволяют клиенту и серверу отправлять дополнительную информацию с HTTP запросом или ответом. В HTTP-заголовке содержится не чувствительное к регистру название, а затем после (:) непосредственно значение. [Пробелы](https://developer.mozilla.org/en-US/docs/Glossary/Whitespace) перед значением игнорируются.

Пользовательские собственные заголовки исторически использовались с префиксом X, но это соглашение было объявлено устаревшим в июне 2012 года из-за неудобств, вызванных тем, что нестандартные поля стали стандартом в [RFC 6648](https://tools.ietf.org/html/rfc6648); другие перечислены в реестре [IANA](https://www.iana.org/assignments/message-headers/perm-headers.html), исходное содержимое которого было определено в [RFC 4229](https://tools.ietf.org/html/rfc4229). IANA также поддерживает [реестр предлагаемых новых заголовков HTTP](https://www.iana.org/assignments/message-headers/message-headers.xhtml).

HTTP-заголовки сопровождают обмен данными по протоколу HTTP. Они могут содержать описание данных и информацию, необходимую для взаимодействия между клиентом и сервером. Заголовки и их статусы перечислены в [реестре IANA](https://www.iana.org/assignments/message-headers/perm-headers.html), который постоянно обновляется.

Заголовки могут быть сгруппированы по следующим контекстам:

* [Основные заголовки](https://developer.mozilla.org/ru/docs/Glossary/General_header) применяется как к запросам, так и к ответам, но не имеет отношения к данным, передаваемым в теле.
* Заголовки запроса содержит больше информации о ресурсе, который нужно получить, или о клиенте, запрашивающем ресурс.
* [Заголовки ответа (en-US)](https://developer.mozilla.org/en-US/docs/Glossary/Response_header) содержат дополнительную информацию об ответе, например его местонахождение, или о сервере, предоставившем его.
* [Заголовки сущности](https://developer.mozilla.org/ru/docs/Glossary/Entity_header) содержат информацию о теле ресурса, например его [длину содержимого](https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/Content-Length)или тип [MIME](https://developer.mozilla.org/en-US/docs/Web/SVG/Content_type).

Заголовки также могут быть сгруппированы согласно тому, как [прокси (proxies)](https://developer.mozilla.org/en-US/docs/Glossary/Proxy_server) обрабатывают их:

* [Connection](https://developer.mozilla.org/ru/docs/Web/HTTP/Headers/Connection)
* [Keep-Alive (en-US)](https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/Keep-Alive)
* [Proxy-Authenticate (en-US)](https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/Proxy-Authenticate)
* [Proxy-Authorization (en-US)](https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/Proxy-Authorization)
* [TE (en-US)](https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/TE)
* [Trailer (en-US)](https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/Trailer)
* [Transfer-Encoding (en-US)](https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/Transfer-Encoding)
* [Upgrade (en-US)](https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/Upgrade).

**Сквозные заголовки**  
     Эти заголовки должны быть переданы конечному получателю сообщения: серверу для запроса или клиенту для ответа. Промежуточные прокси-серверы должны повторно передавать эти заголовки без изменений, а кеши должны их хранить.

**Хоп-хоп заголовки (Хоп-хоп заголовки)**  
     Эти заголовки имеют смысл только для одного соединения транспортного уровня и не должны повторно передаваться прокси или кешироваться. Обратите внимание, что с помощью общего заголовка [Connection](https://developer.mozilla.org/ru/docs/Web/HTTP/Headers/Connection) могут быть установлены только заголовки переходов.

Все заголовки разделяются на четыре основных группы:

1. [**General Headers**](https://ru.wikipedia.org/wiki/%D0%A1%D0%BF%D0%B8%D1%81%D0%BE%D0%BA_%D0%B7%D0%B0%D0%B3%D0%BE%D0%BB%D0%BE%D0%B2%D0%BA%D0%BE%D0%B2_HTTP#%D0%9E%D1%81%D0%BD%D0%BE%D0%B2%D0%BD%D1%8B%D0%B5_%D0%B7%D0%B0%D0%B3%D0%BE%D0%BB%D0%BE%D0%B2%D0%BA%D0%B8) ([рус.](https://ru.wikipedia.org/wiki/%D0%A0%D1%83%D1%81%D1%81%D0%BA%D0%B8%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA) *Основные заголовки*) — должны включаться в любое сообщение клиента и сервера.
2. [**Request Headers**](https://ru.wikipedia.org/wiki/%D0%A1%D0%BF%D0%B8%D1%81%D0%BE%D0%BA_%D0%B7%D0%B0%D0%B3%D0%BE%D0%BB%D0%BE%D0%B2%D0%BA%D0%BE%D0%B2_HTTP#%D0%97%D0%B0%D0%B3%D0%BE%D0%BB%D0%BE%D0%B2%D0%BA%D0%B8_%D0%B7%D0%B0%D0%BF%D1%80%D0%BE%D1%81%D0%B0) ([рус.](https://ru.wikipedia.org/wiki/%D0%A0%D1%83%D1%81%D1%81%D0%BA%D0%B8%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA) *Заголовки запроса*) — используются только в запросах клиента.
3. [**Response Headers**](https://ru.wikipedia.org/wiki/%D0%A1%D0%BF%D0%B8%D1%81%D0%BE%D0%BA_%D0%B7%D0%B0%D0%B3%D0%BE%D0%BB%D0%BE%D0%B2%D0%BA%D0%BE%D0%B2_HTTP#%D0%97%D0%B0%D0%B3%D0%BE%D0%BB%D0%BE%D0%B2%D0%BA%D0%B8_%D0%BE%D1%82%D0%B2%D0%B5%D1%82%D0%B0) ([рус.](https://ru.wikipedia.org/wiki/%D0%A0%D1%83%D1%81%D1%81%D0%BA%D0%B8%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA) *Заголовки ответа*) — только для ответов от сервера.
4. [**Entity Headers**](https://ru.wikipedia.org/wiki/%D0%A1%D0%BF%D0%B8%D1%81%D0%BE%D0%BA_%D0%B7%D0%B0%D0%B3%D0%BE%D0%BB%D0%BE%D0%B2%D0%BA%D0%BE%D0%B2_HTTP#%D0%97%D0%B0%D0%B3%D0%BE%D0%BB%D0%BE%D0%B2%D0%BA%D0%B8_%D1%81%D1%83%D1%89%D0%BD%D0%BE%D1%81%D1%82%D0%B8) ([рус.](https://ru.wikipedia.org/wiki/%D0%A0%D1%83%D1%81%D1%81%D0%BA%D0%B8%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA) *Заголовки сущности*) — сопровождают каждую [сущность](https://ru.wikipedia.org/wiki/%D0%A1%D1%83%D1%89%D0%BD%D0%BE%D1%81%D1%82%D1%8C_(%D0%B8%D0%BD%D1%84%D0%BE%D1%80%D0%BC%D0%B0%D1%82%D0%B8%D0%BA%D0%B0)) сообщения.