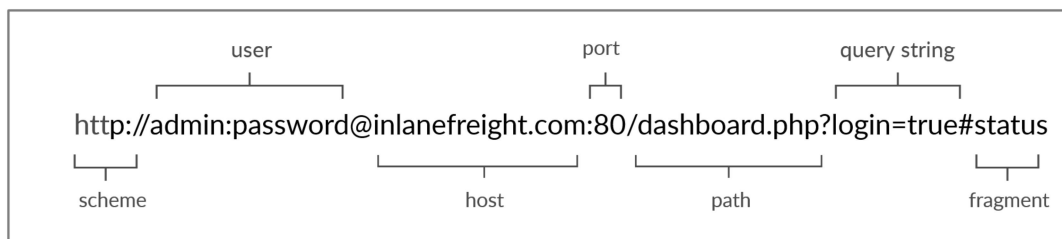


# Web request

## HyperTextTransferProtocol (HTTP) -

### Uniform Resource Locator (URL)-



Component	Example	Description
Scheme	<code>http://</code> <code>https://</code>	This is used to identify the protocol being accessed by the client, and ends with a colon and a double slash ( <code>://</code> )
User Info	<code>admin:password@</code>	This is an optional component that contains the credentials (separated by a colon <code>:</code> ) used to authenticate to the host, and is separated from the host with an at sign ( <code>@</code> )
Host	<code>inlanefreight.com</code>	The host signifies the resource location. This can be a hostname or an IP address
Port	<code>:80</code>	The <b>Port</b> is separated from the <b>Host</b> by a colon ( <code>:</code> ). If no port is specified, <code>http</code> schemes default to port <b>80</b> and <code>https</code> default to port <b>443</b>
Path	<code>/dashboard.php</code>	This points to the resource being accessed, which can be a file or a folder. If there is no path specified, the server returns the default index (e.g. <code>index.html</code> ).

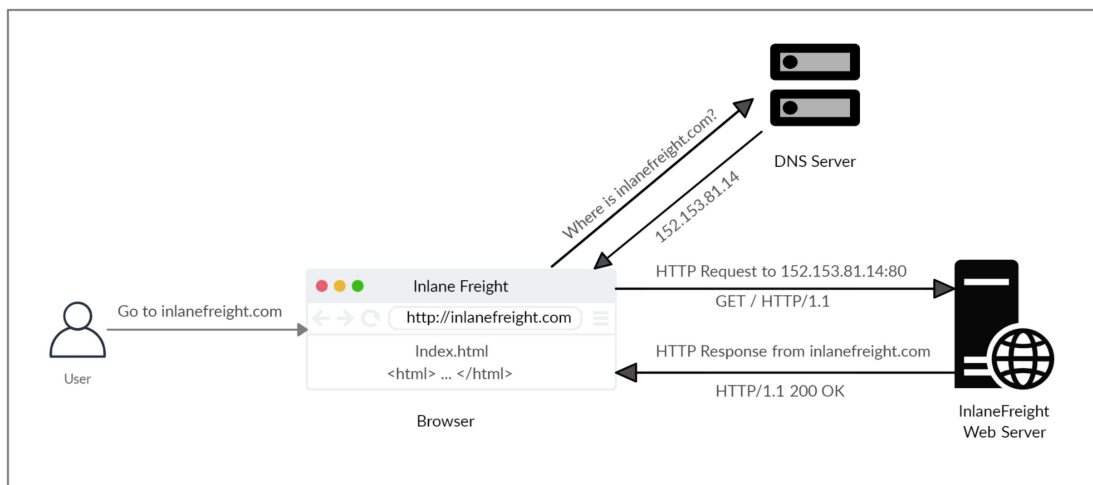
Query String `?login=true`

The query string starts with a question mark (?), and consists of a parameter (e.g. `login`) and a value (e.g. `true`). Multiple parameters can be separated by an ampersand (&).

Fragments `#status`

Fragments are processed by the browsers on the client-side to locate sections within the primary resource (e.g. a header or section on the page).

## HTTP FLOW -



## cURL

Command	Description
<code>curl -h</code>	cURL help menu
<code>curl inlane freight.com</code>	Basic GET request
<code>curl -s -O inlane freight.com/index.html</code>	Download file

No.	Time	Source	Destination	Protocol	Length	Info
	10.1.444226935	216.58.197.36	192.168.0.108	TLSv1.2	1486	Application Data
	11.444242725	192.168.0.108	216.58.197.36	TCP	68	35854 → 443 [ACK] Seq=163 Ack=1704 Win=1673 Len=0 TSva
	12.444662791	216.58.197.36	192.168.0.108	TLSv1.2	2904	Application Data, Application Data
	13.444671948	192.168.0.108	216.58.197.36	TCP	68	35854 → 443 [ACK] Seq=163 Ack=4540 Win=1717 Len=0 TSva
	14.444790442	216.58.197.36	192.168.0.108	TLSv1.2	2416	Application Data, Application Data
	15.444801724	192.168.0.108	216.58.197.36	TCP	68	35854 → 443 [ACK] Seq=163 Ack=6888 Win=1754 Len=0 TSva
▶ Frame 10: 1486 bytes on wire (11888 bits), 1486 bytes captured (11888 bits) on interface 0						
▶ Linux cooked capture						
▶ Internet Protocol Version 4, Src: 216.58.197.36, Dst: 192.168.0.108						
▶ Transmission Control Protocol, Src Port: 443, Dst Port: 35854, Seq: 286, Ack: 163, Len: 1418						
▼ Transport Layer Security						
▼ TLSv1.2 Record Layer: Application Data Protocol: http-over-tls						
Content Type: Application Data (23)						
Version: TLS 1.2 (0x0303)						
Length: 1413						
Encrypted Application Data: bfbb1a63857cc8fb4f78e3650ab13767a56f92ee89df919...						

<https://www.google.com>

Site Information for www.google.com

Connection secure

Permissions

You have not granted this site any special permissions.

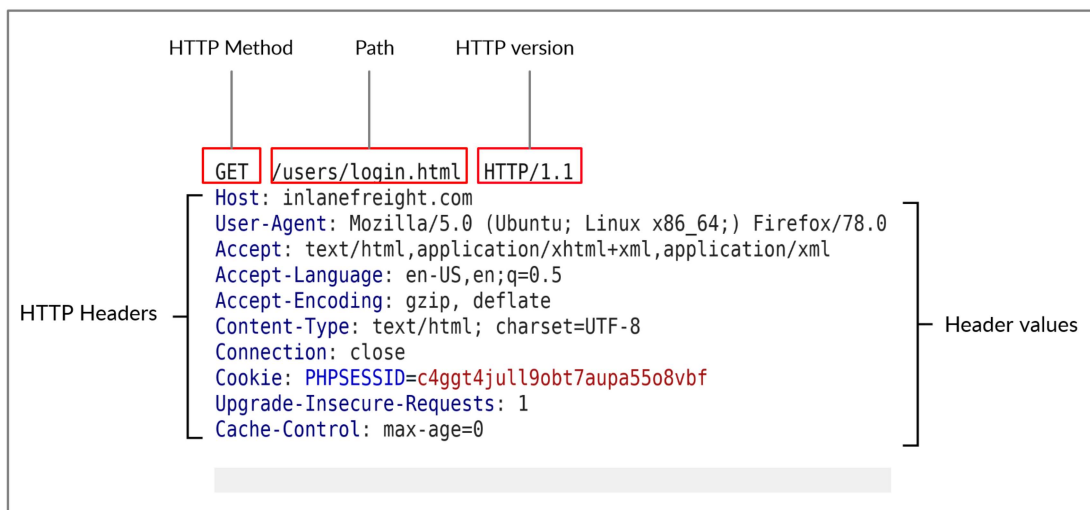
Clear Cookies and Site Data...

74.573774918	192.168.0.108	192.168.0.108	TCP	7640386 → 80	[SYN] Seq=0 Win=65495 Len=0 MSS=65495 SACK_PERM=1
84.573794134	192.168.0.108	192.168.0.108	TCP	7680 → 40386	[SYN, ACK] Seq=0 Ack=1 Win=65483 Len=0 MSS=65495 S
94.573806187	192.168.0.108	192.168.0.108	TCP	6840386 → 80	[ACK] Seq=1 Ack=1 Win=65536 Len=0 TSval=280780439
104.573966701	192.168.0.108	192.168.0.108	HTTP	640 POST /login.php	HTTP/1.1 (application/x-www-form-urlencoded)
114.573985767	192.168.0.108	192.168.0.108	TCP	6880 → 40386	[ACK] Seq=1 Ack=573 Win=65024 Len=0 TSval=28078043

Frame 10: 640 bytes on wire (5120 bits), 640 bytes captured (5120 bits) on interface 0  
Linux cooked capture  
Internet Protocol Version 4, Src: 192.168.0.108, Dst: 192.168.0.108  
Transmission Control Protocol, Src Port: 40386, Dst Port: 80, Seq: 1, Ack: 1, Len: 572  
Hypertext Transfer Protocol  
HTML Form URL Encoded: application/x-www-form-urlencoded  

- Form item: "username" = "admin"
  - Key: username
  - Value: admin
- Form item: "password" = "password"
  - Key: password
  - Value: password

## HTTP Requests and Responses



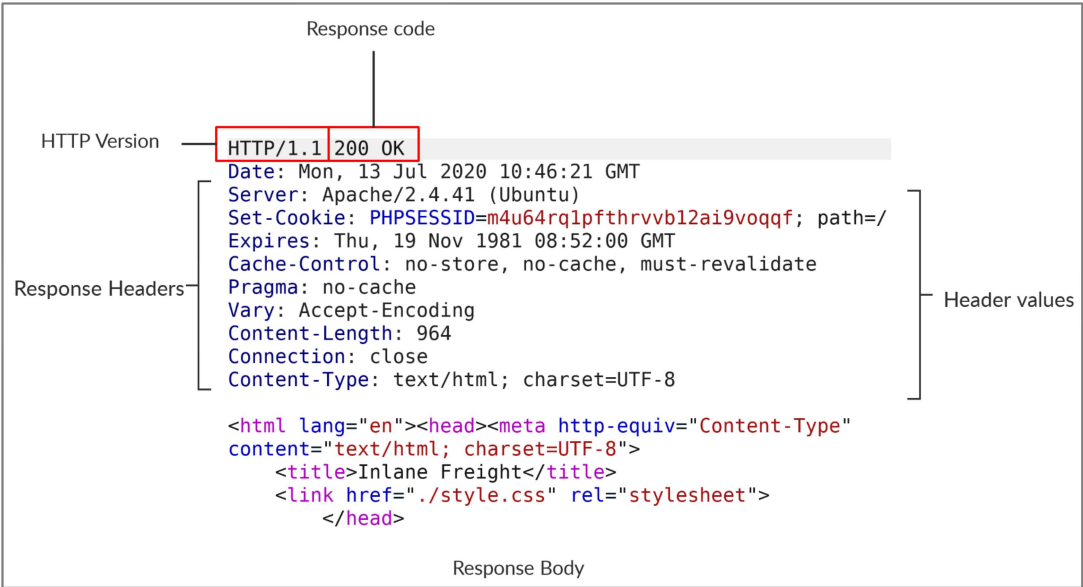
The image above shows an HTTP GET request to the URL:

- <http://inlanefreight.com/users/login.html>

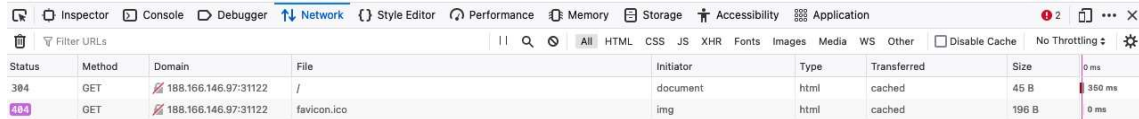
The first line of any HTTP request contains three main fields 'separated by spaces':

Field	Example	Description
Metho d	GET	The HTTP method or verb, which specifies the type of action to perform.
Path	/users/login. html	The path to the resource being accessed. This field can also be suffixed with a query string (e.g. <a href="#">?username=user</a> ).
Versio n	HTTP/1.1	The third and final field is used to denote the HTTP version.

# HTTP Response



# Browser DevTools



Status	Method	Domain	File	Initiator	Type	Transferred	Size	
304	GET	188.166.146.97:31122	/	document	html	cached	45 B	0 ms
404	GET	188.166.146.97:31122	favicon.ico	img	html	cached	196 B	360 ms

## HTTP Headers

We have seen examples of HTTP requests and response headers in the previous section.

Such HTTP headers pass information between the client and the server. Some headers are only used with either requests or responses, while some other general headers are common to both.

Headers can have one or multiple values, appended after the header name and separated by a colon. We can divide headers into the following categories:

1. General Headers
2. Entity Headers
3. Request Headers
4. Response Headers
5. Security Headers

## General Headers

**General headers** are used in both HTTP requests and responses. They are contextual and are used to **describe the message rather than its contents**.

Header	Example	Description
Date	Date: Wed, 16 Feb 2022 10:38:44 GMT	Holds the date and time at which the message originated. It's preferred to convert the time to the standard <b>UTC</b> time zone.
Connection	Connection: close	Dictates if the current network connection should stay alive after the request finishes. Two commonly used values for this header are <b>close</b> and <b>keep-alive</b> . The <b>close</b> value from either the client or server means that they would like to terminate the connection, while the <b>keep-alive</b> header indicates that the connection should remain open to receive more data and input.

## Entity Headers

Similar to general headers, **Entity Headers** can be **common to both the request and response**. These headers are used to **describe the content** (entity) transferred by a message. They are usually found in responses and POST or PUT requests.

Header	Example	Description
Content-Type	Content-Type: text/html	Used to describe the type of resource being transferred. The value is automatically added by the browsers on the client-side and returned in the server response. The <b>charset</b> field denotes the encoding standard, such as <b>UTF-8</b> .
Media-Type	Media-Type: application/pdf	The <b>media-type</b> is similar to <b>Content-Type</b> , and describes the data being transferred. This header can play a crucial role in making the server interpret our input. The <b>charset</b> field may also be used with this header.
Boundary	boundary="b4e4fbd93540"	Acts as a marker to separate content when there is more than one in the same message. For example, within a form data, this boundary gets used as <b>--b4e4fbd93540</b> to separate different parts of the form.

Content-Length	Content-Length: 385	Holds the size of the entity being passed. This header is necessary as the server uses it to read data from the message body, and is automatically generated by the browser and tools like cURL.
Content-Encoding	Content-Encoding: gzip	Data can undergo multiple transformations before being passed. For example, large amounts of data can be compressed to reduce the message size. The type of encoding being used should be specified using the <b>Content-Encoding</b> header.

## Request Headers

The client sends **Request Headers** in an HTTP transaction. These headers are **used in an HTTP request and do not relate to the content** of the message. The following headers are commonly seen in HTTP requests.

Header	Example	Description
Host	Host: <a href="http://www.inlanefreight.com">www.inlanefreight.com</a>	Used to specify the host being queried for the resource. This can be a domain name or an IP address. HTTP servers can be configured to host different websites, which are revealed based on the hostname. This makes the host header an important enumeration target, as it can indicate the existence of other hosts on the target server.
User-Agent	User-Agent: curl/7.77.0	The <b>User-Agent</b> header is used to describe the client requesting resources. This header can reveal a lot about the client, such as the browser, its version, and the operating system.
Referer	Referer: <a href="http://www.inlanefreight.com/">http://www.inlanefreight.com/</a>	Denotes where the current request is coming from. For example, clicking a link from Google search results would make <a href="https://google.com">https://google.com</a> the referer. Trusting this header can be dangerous as it can be easily manipulated, leading to unintended consequences.
Accept	Accept: */*	The <b>Accept</b> header describes which media types the client can understand. It can contain multiple media types separated by commas. The <b>*/*</b> value signifies that all media types are accepted.



Cookie	Cookie: PHPSESSID=b4e4fbd93540	Contains cookie-value pairs in the format <b>name=value</b> . A <b>cookie</b> is a piece of data stored on the client-side and on the server, which acts as an identifier. These are passed to the server per request, thus maintaining the client's access. Cookies can also serve other purposes, such as saving user preferences or session tracking. There can be multiple cookies in a single header separated by a semi-colon.
Authorization	Authorization: BASIC cGFzc3dvcmQK	Another method for the server to identify clients. After successful authentication, the server returns a token unique to the client. Unlike cookies, tokens are stored only on the client-side and retrieved by the server per request. There are multiple types of authentication types based on the webserver and application type used.

A complete list of request headers and their usage can be found [here](#).

## Response Headers

**Response Headers** can be used in an HTTP response and do not relate to the content. Certain response headers such as **Age**, **Location**, and **Server** are used to provide more context about the response. The following headers are commonly seen in HTTP responses.

Header	Example	Description
Server	Server: Apache/2.2.14 (Win32)	Contains information about the HTTP server, which processed the request. It can be used to gain information about the server, such as its version, and enumerate it further.
Set-Cookie	Set-Cookie: PHPSESSID=b4e4fbd93540	Contains the cookies needed for client identification. Browsers parse the cookies and store them for future requests. This header follows the same format as the <b>Cookie</b> request header.
WWW-Authenticate	WWW-Authenticate: BASIC realm="localhost"	Notifies the client about the type of authentication required to access the requested resource.

## Security Headers

Finally, we have [Security Headers](#). With the increase in the variety of browsers and web-based attacks, defining certain headers that enhanced security was necessary. HTTP Security headers are [a class of response headers used to specify certain rules and policies](#) to be followed by the browser while accessing the website.

Header	Example	Description
<a href="#">Content-Security-Policy</a>	<a href="#">Content-Security-Policy: script-src 'self'</a>	Dictates the website's policy towards externally injected resources. This could be JavaScript code as well as script resources. This header instructs the browser to accept resources only from certain trusted domains, hence preventing attacks such as <a href="#">Cross-site scripting (XSS)</a> .
<a href="#">Strict-Transport-Security</a>	<a href="#">Strict-Transport-Security: max-age=31536000</a>	Prevents the browser from accessing the website over the plaintext HTTP protocol, and forces all communication to be carried over the secure HTTPS protocol. This prevents attackers from sniffing web traffic and accessing protected information such as passwords or other sensitive data.
<a href="#">Referrer-Policy</a>	<a href="#">Referrer-Policy: origin</a>	Dictates whether the browser should include the value specified via the <a href="#">Referer</a> header or not. It can help in avoiding disclosing sensitive URLs and information while browsing the website.

## General Headers

General headers are used in both HTTP requests and responses. They are contextual and are used to describe the message rather than its contents.

Header	Example	Description
<a href="#">Date</a>	<a href="#">Date: Wed, 16 Feb 2022 10:38:44 GMT</a>	Holds the date and time at which the message originated. It's preferred to convert the time to the standard UTC time zone.

Connection	Connection: close	Dictates if the current network connection should stay alive after the request finishes. Two commonly used values for this header are close and keep-alive. The close value from either the client or server means that they would like to terminate the connection, while the keep-alive header indicates that the connection should remain open to receive more data and input.
------------	-------------------	---

## Entity Headers

Similar to general headers, Entity Headers can be common to both the request and response. These headers are used to describe the content (entity) transferred by a message. They are usually found in responses and POST or PUT requests.

Header	Example	Description
Content-Type	Content-Type: text/html	Used to describe the type of resource being transferred. The value is automatically added by the browsers on the client-side and returned in the server response. The charset field denotes the encoding standard, such as UTF-8.
Media-Type	Media-Type: application/pdf	The media-type is similar to Content-Type, and describes the data being transferred. This header can play a crucial role in making the server interpret our input. The charset field may also be used with this header.
Boundary	boundary="b4e4fb d93540"	Acts as a marker to separate content when there is more than one in the same message. For example, within a form data, this

---

		boundary gets used as --b4e4fbd93540 to separate different parts of the form.
--	--	---

Content-Length	Content-Length: 385	Holds the size of the entity being passed. This header is necessary as the server uses it to read data from the message body, and is automatically generated by the browser and tools like cURL.
----------------	---------------------	--

Content-Encoding	Content-Encoding: gzip	Data can undergo multiple transformations before being passed. For example, large amounts of data can be compressed to reduce the message size. The type of encoding being used should be specified using the Content-Encoding header.
------------------	------------------------	--

---

## Request Headers

The client sends Request Headers in an HTTP transaction. These headers are used in an HTTP request and do not relate to the content of the message. The following headers are commonly seen in HTTP requests.

---

Header	Example	Description
Host	Host: www.inlanefreight.com	Used to specify the host being queried for the resource. This can be a domain name or an IP address. HTTP servers can be configured to host different websites, which are revealed based on the hostname. This makes the host header an important enumeration target, as it can indicate the existence of other hosts on the target server.

---

User-Agent	User-Agent: curl/7.77.0	The User-Agent header is used to describe the client requesting resources. This header can reveal a lot about the client, such as the browser, its version, and the operating system.
Referer	Referer: http://www.inlanefreight.com/	Denotes where the current request is coming from. For example, clicking a link from Google search results would make https://google.com the referer. Trusting this header can be dangerous as it can be easily manipulated, leading to unintended consequences.
Accept	Accept: */*	The Accept header describes which media types the client can understand. It can contain multiple media types separated by commas. The */* value signifies that all media types are accepted.
Cookie	Cookie: PHPSESSID=b4e4fbd93540	Contains cookie-value pairs in the format name=value. A cookie is a piece of data stored on the client-side and on the server, which acts as an identifier. These are passed to the server per request, thus maintaining the client's access. Cookies can also serve other purposes, such as saving user preferences or session tracking. There can be multiple cookies in a single header separated by a semi-colon.
Authorization	Authorization: Basic cGFZc3dvcmQK	Another method for the server to identify clients. After successful authentication, the server returns a token unique to the client. Unlike cookies, tokens are stored only on the client-side and retrieved by the server per request. There are multiple types of authentication types based on the webserver and application type used.

A complete list of request headers and their usage can be found [here](#).

## Response Headers

Response Headers can be used in an HTTP response and do not relate to the content. Certain response headers such as Age, Location, and Server are used to provide more context about the response. The following headers are commonly seen in HTTP responses.

Header	Example	Description
Server	Server: Apache/2.2.14 (Win32)	Contains information about the HTTP server, which processed the request. It can be used to gain information about the server, such as its version, and enumerate it further.
Set-Cookie	Set-Cookie: PHPSESSID=b4e4fbd93540	Contains the cookies needed for client identification. Browsers parse the cookies and store them for future requests. This header follows the same format as the Cookie request header.
WWW-Authenticate	WWW-Authenticate: BASIC realm="localhost"	Notifies the client about the type of authentication required to access the requested resource.

## Security Headers

Finally, we have Security Headers. With the increase in the variety of browsers and web-based attacks, defining certain headers that enhanced security was necessary.

HTTP Security headers are a class of response headers used to specify certain rules and policies to be followed by the browser while accessing the website.

Header	Example	Description
Content-Security-Policy	Content-Security-Policy: script-src 'self'	Dictates the website's policy towards externally injected resources. This could be JavaScript code as well as script resources. This header instructs the browser to accept resources only from certain trusted domains, hence preventing attacks such as Cross-site scripting (XSS).
Strict-Transport-Security	Strict-Transport-Security: max-age=31536000	Prevents the browser from accessing the website over the plaintext HTTP protocol, and forces all communication to be carried over the secure HTTPS protocol. This prevents attackers from sniffing web traffic and accessing protected information such as passwords or other sensitive data.
Referrer-Policy	Referrer-Policy: origin	Dictates whether the browser should include the value specified via the Referer header or not. It can help in avoiding disclosing sensitive URLs and information while browsing the website.