Article 04 – HTTP Protocol

**HTTP Versions**

1) HTTP/0.9

Introduced in 1991. It is the first version of HTML to be released. Only supported a simple GET request with no headers or status codes.

2) HTTP/1.0

Introduced in 1996. Added many of the features available in HTML today, such as the GET, POST, and HEAD methods. Supports headers and response codes such as 200 (OK) and 404 (Not Found).

3) HTTP/1.1

Introduced in 1997. Supports pipelining, which implies that the user can send multiple requests at a time without waiting for each response.

4) HTTP/2

Introduced in 2015. Supports multiplexing, which implies that multiple requests and response cycles can be completed in a single TCP session. Allows server push, which means that the server may proactively send resources (e.g., HTML/CSS/JS) before the client even requests it, thus reducing load times.

**HTTP v/s HTTPS**

HTTP: An application-layer protocol used for fetching resources (such as HTML, JSON, CSS) over the Internet. All requests and responses are sent as plain text.

HTTPS (HTTP Secure): Refers to an HTTP encapsulated in a TLS channel.

The key difference between the two is that HTTP is fast and simple, but leaves all the data exposed. Whereas, HTTPS adds a secure TLS layer that encrypts, authenticates, and protects the data.

**SSL/TLS Layer**

The Secure Sockets Layer (SSL) and the Transport Layer Security (TLS) are cryptographic protocols designed to secure communications over a network. The TLS is a successor (better) to the SSL.

The TLS sits between the application layer (HTTP) and the transport layer (TCP) to provide three fundamental security features:

1) Confidentiality (Encryption)

After the initial handshake, all data is encrypted with a symmetric cipher (such as AES). Thus, even if someone captures the packets, they won’t be able to decipher the data.

2) Integrity (Tamper-Detection)

Each encrypted packet consists of a Message Authentication Code (MAC) or an Authenticated Encryption with Associated Data (AEAD).

The MAC/AEAD ensures that any modification of the encrypted data (accidental or malicious) is detected. Altered packets are discarded.

3) Authentication (Trust Establishment)

During the TLS handshake, the server presents a certificate signed by a Certificate Authority (CA), which verifies that you really are talking to “bank.example.com” and not an impostor.

Optionally, the server can request a client certificate to authenticate the client to the server (mutual TLS).

**HTTP Methods**

An HTTP Method (also known as an HTTP verb) is the part of an HTTP request that tells the server what action the client wants to perform on the resource identified by the URL.

1) GET

Purpose: Retrieves a representation of a resource (e.g. a web page, or JSON data)

Does not modify the server’s state. E.g. viewing an article, requesting a user’s profile

2) POST

Purpose: Submit data to the server to create a subordinate resource or trigger processing.

Modifies the server’s state. E.g. form submissions, creating a blog post

3) PUT

Purpose: Update or replace the resource at this exact URI.

Modifies the server’s state. E.g. updating a user’s profile, uploading a file to a fixed path.

4) PATCH

Purpose: Apply partial modifications to a resource.

Modifies the server. E.g. changing just the email field on a user

5) DELETE

Purpose: Remove the resource identified by the URI.

Modifies the server’s state. E.g. deleting a comment posted by the user

6) HEAD

Purpose: Identical to GET but only retrieves response headers (no body).

Does not modify the server’s state. E.g. checking if a resource exists.

**HTTP Status Codes**

HTTP Status Codes are standardized codes that a server returns in response to an HTTP request. They can be grouped by their first digit in the following manner:  
  
1) 1xx – Informational

* 100 Continue: Client may continue with their request (e.g. after sending headers, waiting to send body)
* 101 Switching Protocols: Server is switching protocols

2) 2xx – Success

* 200 OK: Standard response for GET, POST, etc.
* 201 Created: Successful request that results in a new resource being created (e.g., after a POST).
* 202 Accepted: Request accepted for posting but not yet completed.
* 204 No Content: Success with no response body (common for DELETE).

3) 3xx – Redirection

* 301 Moved Permanently: Resource has a new permanent URL. The client must update their link.
* 302 Found: Temporary redirect when the user searches for an indirect URL.

4) 4xx – Client Error

* 400 Bad Request: Incorrect request syntax or invalid message request framing
* 401 Unauthorized: Authorization is required (or has failed).
* 403 Forbidden: Server understood the request but refuses to authorize it.
* 404 Not Found: The requested resource does not exist at the server.
* 405 Method Not Allowed: HTTP method is not supported for the requested resource.
* 408 Request Timeout: Client didn’t produce a request within the server’s time limit.
* 429 Too Many Requests: Client sent too many requests in the given amount of time (rate limiting).

5) 5xx – Server Error

* 500 Internal Server Error: Generic error when the server throws an unexpected exception.
* 501 Not Implemented: Server does not support the functionality required to fulfill the request.
* 502 Bad Gateway: Server, acting as a gateway, received an invalid response from the upstream server.
* 503 Service Unavailable: Server is currently unable to handle the request (overloaded or down for maintenance).
* 504 Gateway Timeout: The gateway or proxy did not receive a timely response from the upstream server.

A gateway/proxy is an intermediary that sits between a client and another server, handling requests and responses on the client’s behalf.