**HL7 over HTTP Specification**

|  |  |
| --- | --- |
| Draft | Jan.17th, 2018 |
| URL: | <https://hapifhir.github.io/hapi-hl7v2/hapi-hl7overhttp/specification.html> |

This document is a DRAFT for comments. Please submit comments the [HAPI Mailing List](http://hl7api.sourceforge.net/mail-lists.html) or by contacting directly at [jamesagnew@users.sourceforge.net](mailto:jamesagnew@users.sourceforge.net)

HL7 over HTTP is an initiative to provide a standardized transport mechanism to send HL7 v2 messages over a network using the HTTP protocol.

Requirement levels (MUST, SHOULD, etc.) are used according to RFC 2119 (<http://www.ietf.org/rfc/rfc2119.txt>).

**Contents**

Inhalt

[1.0 - Introduction 2](#_Toc503942844)

[1.1 Intent 2](#_Toc503942845)

[1.2 Scope 2](#_Toc503942846)

[2.0 - Specification 3](#_Toc503942847)

[2.1 Use of HTTP 3](#_Toc503942848)

[2.2 Transport Messages 3](#_Toc503942849)

[2.3 Addressing 4](#_Toc503942850)

[2.4 Content Type and Character Set 4](#_Toc503942851)

[2.5 Date 5](#_Toc503942852)

[2.6 HTTP Response Codes 5](#_Toc503942853)

[2.7 HL7 Query/Response Mode 7](#_Toc503942854)

[3.0 - Security Profiles 7](#_Toc503942855)

[3.1 Intent 7](#_Toc503942856)

[3.2 Profile Matrix 8](#_Toc503942857)

[3.2 Security Profile Level 0 8](#_Toc503942858)

[3.3 Security Profile Level 1 8](#_Toc503942859)

[3.4 Security Profile Level 2 9](#_Toc503942860)

[3.5 Security Profile Level 3 9](#_Toc503942861)

[4.0 - Implementation Considerations 9](#_Toc503942862)

[4.1 Sequencing and Threading 10](#_Toc503942863)

# 1.0 - Introduction

HL7 over HTTP is a transport mechanism which uses the Hypertext Transfer Protocol (as defined in [RFC 2616](http://tools.ietf.org/html/rfc2616)) to transmit HL7 artifacts (for example messages, documents, resources).

## 1.1 Intent

HL7 over HTTP is intended as an alternative to the traditional Minimal Lower Layer Protocol (MLLP), and provides a number of key improvements:

* HTTP is widely used for a variety of purposes, and is well understood by application developers, network engineers, etc.
* HTTP allows for authentication (username/password) and character set encoding to be specified in a standardized way.
* Tool and hardware support for HTTP is widespread, with many specialized software and hardware devices providing enhanced support for common protocols such as HTTP. In addition, HTTP is widely supported on many platforms, and may be viewed as a "commodity-level" feature. Many languages and toolsets have built in support for HTTP.

## 1.2 Scope

HL7 over HTTP is intended as a **transport level protocol** for transactional messaging between systems which support HL7 protocols.

HL7 over HTTP is **not** intended to provide a different messaging paradigm for HL7 v2 messages (e.g. REST-ful services).

# 2.0 - Specification

HL7 over HTTP uses the standard HTTP/1.1 protocol ([RFC 2616](http://www.ietf.org/rfc/rfc2616.txt)) as a mechanism to transfer a raw HL7 message using standard HL7 encoding (i.e. "vertical bar" or XML) encoding to a destination, and then to receive a response to that message.

## 2.1 Use of HTTP

HL7 over HTTP is a **constraint** of the HTTP/1.1 specification. This means that it should be possible to configure any conformant HTTP implementation to comply with this specification, but it does not mean that any conformant HTTP implementation is necessarily compliant with HL7 over HTTP.

HL7 over HTTP does not extend HTTP/1.1 in any way. There are no protocol features in the specification which are not a part of the source specification.

All recommended and required constraints on the HTTP specification are defined in the following sections.

## 2.2 Transport Messages

Every interaction consists of a **request message** and a **response message**. In a normal message exchange, the receiving system opens a server socket. The sending system then connects to that socket and uses the HTTP POST action to transmit a message. The receiving system then replies with an HTTP 2xx success, and transmits the response message (typically an HL7 Acknowledgement/ACK message) in the body of the response.

The following figure shows a typical message exchange:



## 2.3 Addressing

The **request message** SHALL include a universal resource identifier (URI), per the HTTP specification. This URI SHOULD indicate the appropriate "interface" for the message. This is analogous to the port number in a standard *MLLP* configuration.

For example, a receiving laboratory information system application might open an HTTP server on port 1234 which is capable of receiving several types of data. The sending system could then address ADT data to the receiving system's ADT queue using the following POST request

POST /Lab\_Info\_System/ADT HTTP/1.1

**Implementation Consideration**

An implementing receiving system MAY also choose to use a separate port for each interface, as is the norm for MLLP interfaces. This is acceptable and will work, but is not desirable as it is not common practise in HTTP applications.

## 2.4 Content Type and Character Set

The **request message** and the **response message** SHALL include a Content-Type header, which indicates the type of data being transferred. The following MIME types apply:

* If the message is a "vertical bar" encoded HL7 v2.x message, the content type SHALL be:  
      x-application/hl7-v2+er7
* If the message is an XML encoded HL7 v2.x message, the content type SHALL be:  
      x-application/hl7-v2+xml
* If the message is an XML encoded HL7 v3 message, the content type SHALL be:  
      x-application/hl7-v3+xml
* If the message is an XML encoded FHIR message, the content type SHALL be:  
      x-application/fhir+xml
* If the message is a JSON encoded FHIR message, the content type SHALL be:  
      x-application/fhir+json
* If the message is a CDA document, the content type SHALL be:  
      x-application/xml+cda
* If the **response message** content is not an HL7 message (for instance because the server is malfunctioning and returning a 500 status or the URI being addressed is unknown and the server is returning a 404 status), the content SHALL be another type, such as  
      text/html or text/plain

**Non-error response messages** being transmitted by a server MUST use the same content type as the corresponding **request message**. In other words, if a request message uses XML encoding, the corresponding response message MUST also use XML encoding. See section [2.6 Response Codes](https://hapifhir.github.io/hapi-hl7v2/hapi-hl7overhttp/specification.html#a2_6_RESP_CODES) for more information about response codes for error handling.

**Character Set**

HL7 over HTTP message payloads SHALL use the UTF-8 character encoding scheme.

In addition, both the **request message** and the **response message** SHALL include a charset definition which indicates which character encoding is being used for the message body.

The HL7 v2.x specification allows the character encoding to be specified in MSH-18. When a message is sent using HL7 over HTTP, this value shall be either UNICODE UTF-8 (according to the designation in table 211), or shall not be present (indicating ASCII, a proper subset of UTF-8). When a message is received using HL7 over HTTP, this value SHALL be ignored, and the character encoding specified in the HTTP "Content-Type" header MUST be observed instead.

**Implementation Consideration**

When a message is sent using HL7 over HTTP, the sending application MAY choose to use the value in MSH-18 to inform the choice of charset, although it is not obligated to do so.

An example follows which shows the Content-Type header used to indicate that a "vertical bar" encoded message is being transmitted.

Content-Type: application/hl7-v2+er7; charset=utf-8

## 2.5 Date

**Date**

**Request messages** and **Response messages** SHALL provide a date header which informs the other party of the time that the message transmission was started.

**Implementation Consideration**

Note that the HTTP Date header is not required to agree with the HL7 message timestamp found in the HL7 v2.x message within MSH-7.

The recommended use of these two dates is as follows:

* The HTTP Date header is an indication of the transmission time associated with the message.
* The MSH-7 timestamp is an indication of the message creation time. This means that if the message is created but not delivered immediately (as in the case of a guaranteed delivery queue), the MSH-7 timestamp would contain an date which is earlier than the HTTP Date header.

An example follows which shows a date header:

Date: Tue, 15 Nov 1994 08:12:31 GMT

## 2.6 HTTP Response Codes

If a **receiving application** is able to respond to a request by producing an HL7 response payload, the HTTP status code returned MUST be a code of HTTP/1.1 2xx. This response payload is typically an HL7 ACK message, but other message types may apply to certain exchanges (e.g. RSP\_xxx)

**Application Error Conditions**

The HL7 protocol defines four non successful acknowledgement codes:

* AR (Application Reject)
* AE (Application Error)
* CR (Commit Reject)
* CE (Commit Error)

When a **receiving application** produces an HL7 response payload containing one of these status codes, this is considered a successful interaction at the transport layer, and response code of "HTTP/1.1 2xx" MUST be used.

**Transport Error Conditions**

Non 2xx status codes are reserved for use in cases where the message was not understood, could not be delivered to the receiving application layer, or the receiving application layer failed to process the message due to an unexpected error condition.

**Content Type for Error Conditions**

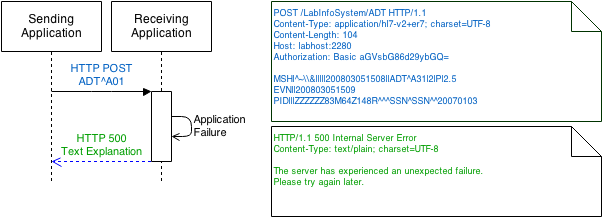
If the response status is not HTTP 2xx, the content type MUST NOT be an HL7 payload type. Instead, a textual representation of the transport level error must be provided using an appropriate content type.

**Implementation Consideration**

Many HTTP implementations will return a content type of "text/html" to provide a nicely formatted error message for internal server errors and invalid addressing. Because HL7 messaging is typically processed between servers, with no HTML browser present, implementers may consider using a content type of "text/plain" in order to increase readability.

**Example**

An example follows which illustrates a transport level problem.



If no HL7 response message is generated and the **receiving application** wishes to indicate an error, the **receiving application** MAY use any content type to encode the response message. Note that using a content type of text/plain may increase readability within system logs.

## 2.7 HL7 Query/Response Mode

HL7 defines a number of messages which are designed to be sent to a receiving system in a query/response mode, where the **receiving system** receives the query message and responds with a message containing data which fulfills the query.

When a **receiving system** responds to a query, a normal HTTP 2xx response code MUST be used in all cases where the application response is an HL7 message.



# 3.0 - Security Profiles

This specification defines three security profiles which offer various levels of security. Each may be appropriate depending on the needs of the application.

## 3.1 Intent

HTTP is a widely used protocol across a vast number of different network topologies and secure environments. As such there are many widely used techniques for securing HTTP traffic. Each mechanism has its own tradeoffs in terms of the type of security it provides (encryption, non-repudiation, tracking, etc.) and in terms of the ease of implementation.

Because HL7 transactions often contain data which is considered highly sensitive, it is important to consider the implications of the security mechanisms in place between systems which exchange HL7 data.

While implementors MAY choose to implement any security techniques which are considered appropriate for the specific implementation, the HL7 over HTTP security profiles take advantage of commonly used techniques for securing the HTTP protocol, and provide a testable and commonly understood definition for security.

These profiles are designed to have strict requirements, such that they may be tested in a conformance environment to ensure compatibility.

## 3.2 Profile Matrix

It is important to select a security model which are appropriate to the specific application being used and the network it is deployed to. The following table shows the security characteristics of the security profiles described in this specification.

* In most cases, it is desirable to use encryption in order to prevent sensitive data from being intercepted.
* Server authentication allows the system receiving data to ensure that the data being received is coming from a legitimate source.
* Client authentication allows the system sending data to ensure that the data being transmitted is not disclosed to an unintended recipient.

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Encryption (Prevents Eavesdropping)** | **Server Authentication (Ensures Client Identity)** | **Client Authentication (Ensures Server Identity)** |
| **Level 0** | No | No | No |
| **Level 1** | No | Yes (Very weak) | No |
| **Level 2** | Yes | Yes | Possible |
| **Level 3** | Yes | Yes (Very strong) | Possible |

## 3.2 Security Profile Level 0

Security Profile Level 0 (referred to hereafter as "Level 0") simply refers to any situations where the profiles below are not used.

Note that the use of these security profiles is not mandatory in order to be conformant to the HL7 over HTTP specification. There are many situations where the use of this type of transport encryption is not neccesary due to network design or other aspects of the solution design.

## 3.3 Security Profile Level 1

Security Profile Level 1 (referred to hereafter as "Level 1") consists only of a security token which allows the receiving system to verify the identity of the sending system.

Level 1 does not include any encryption mechanism, and therefore does not protect messages in transit against eavesdropping or tampering. This is an important consideration when choosing a security profile, and Level 1 should therefore only be used within closed/protected networks.

To implement Level 1, the **sending application** MUST include a header called "Authorization", containing a basic authorization token as described in Section 11.1 of RFC 1945. The basic authorization token consists of the token "Basic", followed by a space, and then followed by a token in the form "username:password" which has been base64 encoded.

The **receiving application** must be capable of validating this authorization token. If the credentials can not be successfully validated, the **receiving application** SHALL return an HTTP response code of "HTTP/1.1 401 Unauthorized".

The following is an example of an authorization header for Level 1:

Authorization: Basic QWxhZGRpbjpvcGVuIHNlc2FtZQ==

The **sending application** MUST use basic authorization in order to be conformant to Level 1. Other types such as "digest access" authorization MUST NOT be used.

## 3.4 Security Profile Level 2

Security Profile Level 2 (referred to hereafter as "Level 2") consists of a security token used to authenticate the client with the server, but additionally adds an encrypted connection.

**Sending applications** and **receiving applications** implementing this profile MUST implement all functionality described in [Security Profile Level 1](https://hapifhir.github.io/hapi-hl7v2/hapi-hl7overhttp/specification.html#SEC_L1).

**Sending applications** and **receiving applications** implementing this profile MUST additionally support the ability to communicate using HTTPS, i.e. using Transport Layer Security (TLS), version 1.2 (RFC 5246, [http://www.ietf.org/rfc/rfc2246.txt](http://tools.ietf.org/html/rfc5246)) or later. Transport Layer Security provides the possibility for end-to-end encryption.

The certificates used SHALL be X.509 certificates based on RSA key with a key length in the range between 1024 and 4096. Supported cipher suites SHALL include TLS\_RSA\_WITH\_AES\_128\_CBC\_SHA, other ciphers MAY be chosen based on site-specific policies.

In addition, **sending applications** implementing this profile MUST be capable of importing a pu5blic key in order to support the use of self-signed certificates.

## 3.5 Security Profile Level 3

**Sending applications** and **receiving applications** implementing this profile MUST support mutual authentication as specified in the TLS specification. Here, the client additionally presents a X.509 client certificate to the server in order to authenticate its access.

Where Level 3 is being used, HTTP Basic Authentication (specified in Level 1) MAY be omitted.

# 4.0 - Implementation Considerations

The following section of this specification provides further details for specific implementation considerations.

## 4.1 Sequencing and Threading

HTTP implementations are commonly multi-threaded, particularly in HTTP server implementations. This can present a risk that messages will be processed in a different order than the one in which they were generated. Because message sequence is often important in HL7 v2.x transactional messaging, implementers should consider how to ensure that messages are processed sequentially. In the event that conventional sequence management cannot be employed then the sequence number protocol SHALL be used.