

[MS-IPHTTPS]: IP over HTTPS (IP-HTTPS) Tunneling Protocol Specification

The IP over HTTPS (IP-HTTPS) Protocol allows for a secure IP tunnel to be established using a secure HTTP connection.

Intellectual Property Rights Notice for Open Specifications Documentation

- **Technical Documentation.** Microsoft publishes Open Specifications documentation for protocols, file formats, languages, standards as well as overviews of the interaction among each of these technologies.
- **Copyrights.** This documentation is covered by Microsoft copyrights. Regardless of any other terms that are contained in the terms of use for the Microsoft website that hosts this documentation, you may make copies of it in order to develop implementations of the technologies described in the Open Specifications and may distribute portions of it in your implementations using these technologies or your documentation as necessary to properly document the implementation. You may also distribute in your implementation, with or without modification, any schema, IDL's, or code samples that are included in the documentation. This permission also applies to any documents that are referenced in the Open Specifications.
- **No Trade Secrets.** Microsoft does not claim any trade secret rights in this documentation.
- **Patents.** Microsoft has patents that may cover your implementations of the technologies described in the Open Specifications. Neither this notice nor Microsoft's delivery of the documentation grants any licenses under those or any other Microsoft patents. However, a given Open Specification may be covered by Microsoft's Open Specification Promise (available here: <http://www.microsoft.com/interop/osp>) or the Community Promise (available here: <http://www.microsoft.com/interop/cp/default.mspx>). If you would prefer a written license, or if the technologies described in the Open Specifications are not covered by the Open Specifications Promise or Community Promise, as applicable, patent licenses are available by contacting iplg@microsoft.com.
- **Trademarks.** The names of companies and products contained in this documentation may be covered by trademarks or similar intellectual property rights. This notice does not grant any licenses under those rights.
- **Fictitious Names.** The example companies, organizations, products, domain names, e-mail addresses, logos, people, places, and events depicted in this documentation are fictitious. No association with any real company, organization, product, domain name, email address, logo, person, place, or event is intended or should be inferred.

Reservation of Rights. All other rights are reserved, and this notice does not grant any rights other than specifically described above, whether by implication, estoppel, or otherwise.

Tools. The Open Specifications do not require the use of Microsoft programming tools or programming environments in order for you to develop an implementation. If you have access to Microsoft programming tools and environments you are free to take advantage of them. Certain

Open Specifications are intended for use in conjunction with publicly available standard specifications and network programming art, and assumes that the reader either is familiar with the aforementioned material or has immediate access to it.

Revision Summary

Date	Revision History	Revision Class	Comments
12/05/2008	0.1	Major	Initial Availability
01/16/2009	0.1.1	Editorial	Revised and edited the technical content.
02/27/2009	0.1.2	Editorial	Revised and edited the technical content.
04/10/2009	0.1.3	Editorial	Revised and edited the technical content.
05/22/2009	1.0	Major	Updated and revised the technical content.
07/02/2009	1.0.1	Editorial	Revised and edited the technical content.
08/14/2009	1.0.2	Editorial	Revised and edited the technical content.
09/25/2009	2.0	Major	Updated and revised the technical content.
11/06/2009	3.0	Major	Updated and revised the technical content.
12/18/2009	4.0	Major	Updated and revised the technical content.
01/29/2010	5.0	Major	Updated and revised the technical content.
03/12/2010	5.0.1	Editorial	Revised and edited the technical content.
04/23/2010	5.0.2	Editorial	Revised and edited the technical content.
06/04/2010	5.0.3	Editorial	Revised and edited the technical content.
07/16/2010	5.0.3	No change	No changes to the meaning, language, or formatting of the technical content.
08/27/2010	5.0.3	No change	No changes to the meaning, language, or formatting of the technical content.
10/08/2010	5.0.3	No change	No changes to the meaning, language, or formatting of the technical content.
11/19/2010	5.0.3	No change	No changes to the meaning, language, or formatting of the technical content.
01/07/2011	5.0.3	No change	No changes to the meaning, language, or formatting of the technical content.
02/11/2011	5.0.3	No change	No changes to the meaning, language, or formatting of the technical content.

Contents

1 Introduction	5
1.1 Glossary	5
1.2 References	5
1.2.1 Normative References	5
1.2.2 Informative References	6
1.3 Overview	6
1.4 Relationship to Other Protocols	6
1.5 Prerequisites/Preconditions	7
1.6 Applicability Statement	8
1.7 Versioning and Capability Negotiation	8
1.8 Vendor-Extensible Fields	8
1.9 Standards Assignments	8
2 Messages	9
2.1 Transport	9
2.2 Message Syntax	10
3 Protocol Details	11
3.1 IP-HTTPS Client Details	11
3.1.1 Abstract Data Model	12
3.1.2 Timers	13
3.1.2.1 Reconnect Timer	13
3.1.3 Initialization	14
3.1.4 Higher-Layer Triggered Events	14
3.1.4.1 Enable IP-HTTPS Link	14
3.1.4.2 Disable IP-HTTPS Link	14
3.1.5 Processing Events and Sequencing Rules	14
3.1.5.1 Establishing the HTTPS Connection	14
3.1.5.2 Bringing the IP-HTTPS Link Up	14
3.1.5.3 Data Transfer	15
3.1.5.4 Error Handling	15
3.1.6 Timer Events	15
3.1.7 Other Local Events	15
3.2 IP-HTTPS Server Details	15
3.2.1 Abstract Data Model	15
3.2.2 Timers	15
3.2.3 Initialization	16
3.2.4 Higher-Layer Triggered Events	16
3.2.4.1 Enable IP-HTTPS Link	16
3.2.4.2 Disable IP-HTTPS Link	16
3.2.5 Processing Events and Sequencing Rules	16
3.2.5.1 Accepting IP-HTTPS Clients	16
3.2.5.2 Data Transfer	16
3.2.5.3 Error Handling	16
3.2.6 Timer Events	16
3.2.7 Other Local Events	17
4 Protocol Examples	18
4.1 Packet Flow and Connection Establishment	18
4.2 Attack Scenarios	19

4.2.1	Unauthorized Client Connecting to an IP-HTTPS Server	19
4.2.2	Unauthorized IP-HTTPS Server Accepting Connections from a Genuine IP-HTTPS Client.....	19
4.2.3	Man in the Middle.....	20
5	Security.....	21
5.1	Security Considerations for Implementers.....	21
5.2	Index of Security Parameters	21
6	Appendix A: Product Behavior.....	22
7	Change Tracking.....	23
8	Index	24

1 Introduction

This document specifies the IP over HTTPS (IP-HTTPS) Protocol, a mechanism to transport IPv6 packets on an HTTPS connection.

1.1 Glossary

The following terms are defined in [\[MS-GLOS\]](#):

The following terms are specific to this document:

IP-HTTPS client: A computer that implements the IP over HTTPS (IP-HTTPS) Protocol and that initiates an IP-HTTPS connection to an **IP-HTTPS server** over TCP port 443.

IP-HTTPS server: A computer that implements the IP over HTTPS (IP-HTTPS) Protocol and listens and accepts IP-HTTPS connections from **IP-HTTPS clients** over TCP port 443.

IP-HTTPS endpoint: An entity that communicates to an **IP-HTTPS client** via the **IP-HTTPS server**.

Uniform Resource Identifier (URI): A resource identifier, which conforms to the rules for Uniform Resource Identifiers as defined in [\[RFC3986\]](#).

MAY, SHOULD, MUST, SHOULD NOT, MUST NOT: These terms (in all caps) are used as described in [\[RFC2119\]](#). All statements of optional behavior use either MAY, SHOULD, or SHOULD NOT.

1.2 References

1.2.1 Normative References

We conduct frequent surveys of the normative references to assure their continued availability. If you have any issue with finding a normative reference, please contact dochelp@microsoft.com. We will assist you in finding the relevant information. Please check the archive site, <http://msdn2.microsoft.com/en-us/library/E4BD6494-06AD-4aed-9823-445E921C9624>, as an additional source.

[RFC1945] Berners-Lee, T., Fielding, R., and Frystyk, H., "Hypertext Transfer Protocol -- HTTP/1.0", RFC 1945, May 1996, <http://www.ietf.org/rfc/rfc1945.txt>

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997, <http://www.ietf.org/rfc/rfc2119.txt>

[RFC2460] Deering, S., and Hinden, R., "Internet Protocol, Version 6 (IPv6) Specification", RFC 2460, December 1998, <http://www.ietf.org/rfc/rfc2460.txt>

[RFC2616] Fielding, R., Gettys, J., Mogul, J., et al., "Hypertext Transfer Protocol -- HTTP/1.1", RFC 2616, June 1999, <http://www.ietf.org/rfc/rfc2616.txt>

[RFC2818] Rescorla, E., "HTTP Over TLS", RFC 2818, May 2000, <http://www.ietf.org/rfc/rfc2818.txt>

[RFC3986] Berners-Lee, T., Fielding, R., and Masinter, L., "Uniform Resource Identifier (URI): Generic Syntax", STD 66, RFC 3986, January 2005, <http://www.ietf.org/rfc/rfc3986.txt>

[RFC4346] Dierks, T., and Rescorla, E., "The Transport Layer Security (TLS) Protocol Version 1.1", RFC 4346, April 2006, <http://www.ietf.org/rfc/rfc4346.txt>

[RFC4861] Narten, T., Nordmark, E., Simpson, W., and Soliman, H., "Neighbor Discovery for IPv6", RFC 4861, September 2007, <http://www.ietf.org/rfc/rfc4861.txt>

[SSLPROXY] Luotonen, A., "Tunneling SSL Through a WWW Proxy", March 1997, <http://tools.ietf.org/html/draft-luotonen-ssl-tunneling-03>

1.2.2 Informative References

[MS-GLOS] Microsoft Corporation, "[Windows Protocols Master Glossary](#)", March 2007.

[RFC1661] Simpson, W., Ed., "The Point-to-Point Protocol (PPP)", STD 51, RFC 1661, July 1994, <http://www.ietf.org/rfc/rfc1661.txt>

1.3 Overview

Many virtual private network (VPN) services provide a way for mobile and home users to access the corporate network remotely by using the Point-to-Point Tunneling Protocol (PPTP) and the Layer Two Tunneling Protocol/Internet Protocol security (L2TP/IPsec). However, with the popularization of firewalls and Web proxies, many service providers (for example, hotels) do not allow the PPTP and L2TP/IPsec traffic. This results in users not receiving ubiquitous connectivity to their corporate networks. For example, generic routing encapsulation (GRE) port blocking by many Internet service providers (ISPs) is a common problem when using PPTP.

The IP over HTTPS (IP-HTTPS) Tunneling Protocol Specification defines the IP over HTTPS (IP-HTTPS) Protocol. IP-HTTPS is a mechanism to encapsulate IP traffic over an HTTPS protocol, as defined in [\[RFC1945\]](#), [\[RFC2616\]](#), and [\[RFC2818\]](#). This protocol enables remote users behind a protocol blocking firewall or proxy server to access a private network using HTTPS. The use of HTTPS enables traversal of most firewalls and Web proxies.

This protocol employs two main roles: client and server. The **IP-HTTPS client** and **IP-HTTPS server** can use either HTTPS or HTTP as a transport.

An IP-HTTPS client: This component is similar to a VPN client. The IP-HTTPS client initiates connections to a configured IP-HTTPS server. The client could become active either automatically (for example, when the client machine is located behind an HTTP firewall and/or HTTP proxy), or based on administrative policy (for example, always on), or based on an explicit user action.

When an IP-HTTPS client is behind an HTTP proxy, the client first establishes a tunnel to the IP-HTTPS server using the CONNECT method, as described in [\[SSLPROXY\]](#).

An IP-HTTPS server: This component is similar to a VPN server, and it is typically positioned at the edge of a network. The IP-HTTPS server directly accepts HTTPS connections made by IP-HTTPS clients. When positioned behind a device that terminates HTTPS on its behalf (such as a reverse proxy or a TLS/SSL load balancer), the server can be configured to listen over HTTP.

1.4 Relationship to Other Protocols

The IP over HTTPS (IP-HTTPS) Protocol allows encapsulation of IPv6 traffic over HTTPS. To do so, it depends on the following protocols:

- Hypertext Transfer Protocol -- HTTP/1.0 [\[RFC1945\]](#).

- Hypertext Transfer Protocol -- HTTP/1.1 [\[RFC2616\]](#).
- HTTP Over TLS [\[RFC2818\]](#).
- Tunneling SSL Through a WWW Proxy [\[SSLPROXY\]](#).
- The Transport Layer Security (TLS) Protocol Version 1.1 [\[RFC4346\]](#).

Once the underlying transport is established, IP-HTTPS enables IPv6 traffic exchanges per usual IPv6 specifications such as:

- Neighbor Discovery for IP Version 6 (IPv6) [\[RFC4861\]](#).
- Protocol, Version 6 (IPv6) Specification [\[RFC2460\]](#).

Note The IP-HTTPS Protocol itself does not have any security or authentication methods. Instead, it relies on HTTPS for mutual authentication, data integrity, and confidentiality.

The relationship between these protocols is illustrated in the following diagram:

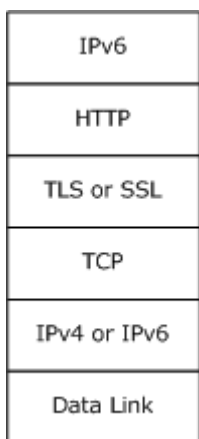


Figure 1: Protocol relationships

1.5 Prerequisites/Preconditions

The IP-HTTPS Protocol requires IPv6 and either HTTP or HTTPS. If HTTPS is used, TLS/SSL is also required for operation.

IP-HTTPS supports both HTTP and HTTPS transports. Unless otherwise noted, the rest of this document uses the term "HTTPS" to also refer to operation over HTTP.

When the transport is HTTPS (as opposed to HTTP), the IP-HTTPS Protocol requires a certificate to be installed on each client and server machine. It also requires the client and the server to be configured with the **Uniform Resource Identifier (URI)** of the server.

The HTTPS connection can be set up over an IPv4 or an IPv6 network.

The IP-HTTPS Protocol does not support proxy authentication; therefore IP-HTTPS clients that are behind Web proxies that require authentication will fail to connect to the IP-HTTPS server.

1.6 Applicability Statement

The IP-HTTPS Protocol is useful for enabling remote users behind a protocol blocking firewall or proxy server to access a private network using HTTPS.

IP-HTTPS only supports IPv6 encapsulation over HTTPS.

1.7 Versioning and Capability Negotiation

The IP over HTTPS (IP-HTTPS) Protocol is a simple encapsulation of IP over HTTPS. It does not have any versioning or capability negotiations with peers.

1.8 Vendor-Extensible Fields

The IP-HTTPS Protocol has no vendor-extensible fields.

1.9 Standards Assignments

There are no standards assignments for this protocol.

2 Messages

2.1 Transport

The following diagrams show the IP-HTTPS protocol stack options for client and server.



Figure 2: IP-HTTPS client protocol stack



Figure 3: IP-HTTPS server protocol stack when using HTTPS encapsulation

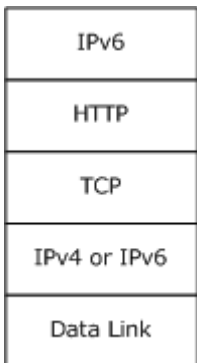


Figure 4: IP-HTTPS server protocol stack when using HTTP encapsulation

IPv6 packets are carried directly within the HTTP payload. HTTP requests and responses are transmitted as described in [\[RFC2616\]](#). The content type of the HTTP entity body carrying IPv6 packets is "application/octet-stream". The sender of the HTTP payload MAY include a Content-Type header in the HTTP request or response to indicate that the content type is an application/octet-stream.

2.2 Message Syntax

The IP-HTTPS Protocol uses HTTP, HTTPS, TLS/SSL and IPv6 as described in section [2.1](#). The IP-HTTPS Protocol does not introduce any new packet formats.

3 Protocol Details

The IP-HTTPS Protocol encapsulates IPv6 packets over an HTTPS connection. This is achieved by creating a tunneled interface, which provides a symmetric link, with multicast and neighbor discovery capabilities, including unreachable neighbor detection and duplicate address detection per [RFC4861]. Like other point-to-point links, for instance PPP [RFC1661], IP-HTTPS uses a static network-layer to link-layer address mapping.

There are two main components to IP-HTTPS: IP-HTTPS clients that are connecting to resources inside a private network (such as an enterprise network) and IP-HTTPS servers that facilitate the connection, as illustrated in the diagram below.

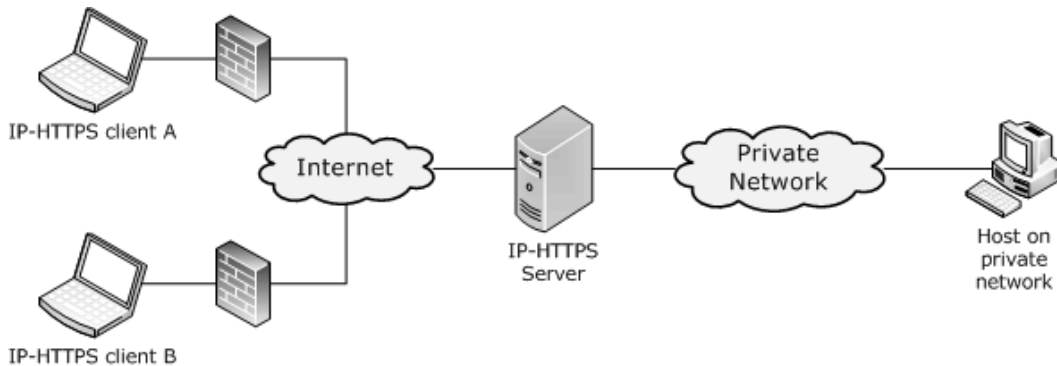


Figure 5: IP-HTTPS clients connected to an IP-HTTPS server

3.1 IP-HTTPS Client Details

The following figure shows the state machine when a client establishes the outgoing IP-HTTPS tunnel.

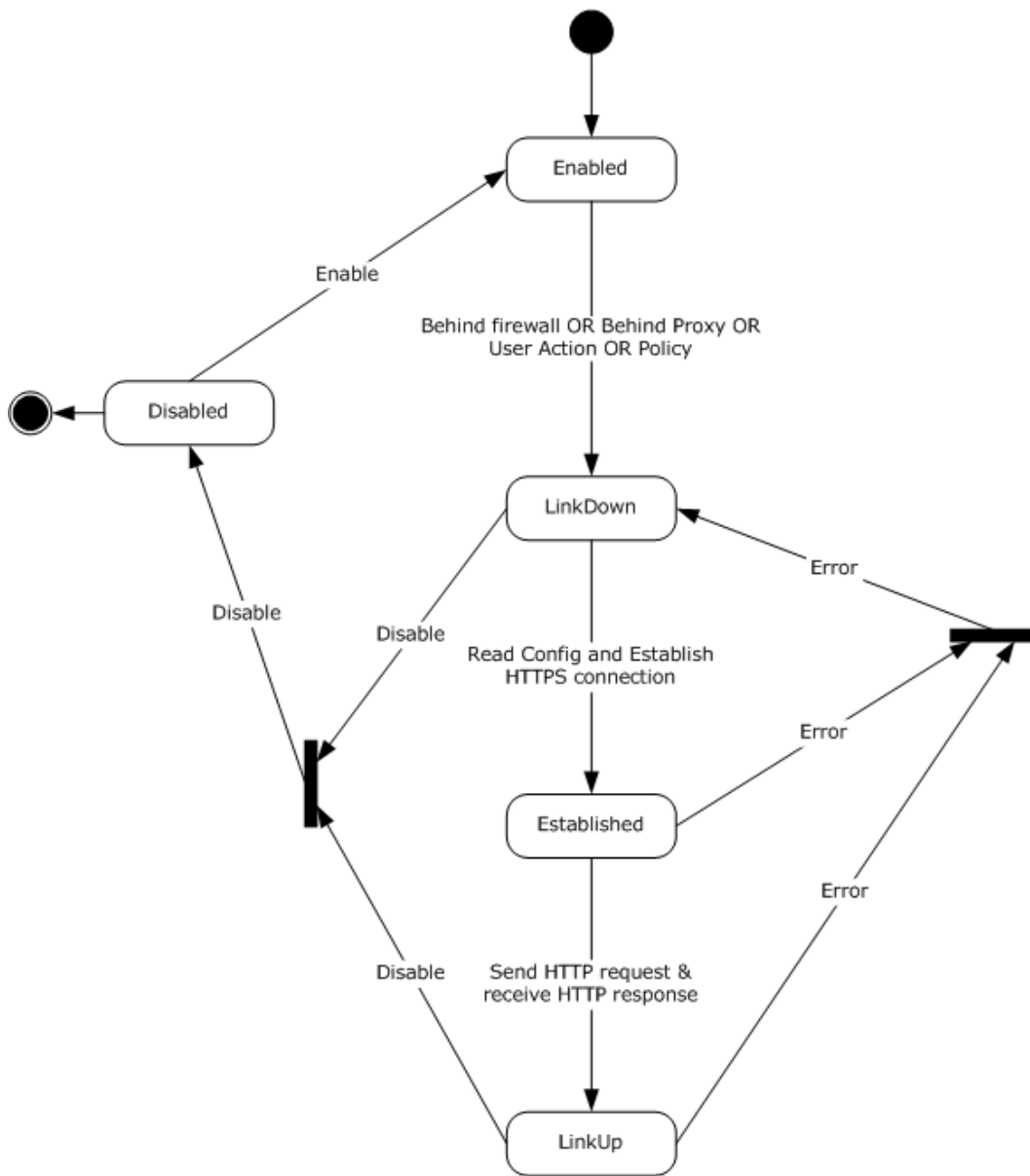


Figure 6: IP-HTTPS client state diagram

3.1.1 Abstract Data Model

This section describes a conceptual model of possible data organization that an implementation maintains to participate in this protocol. The described organization is provided to facilitate the explanation of how the protocol behaves. This document does not mandate that implementations adhere to this model as long as their external behavior is consistent with that described in this document.

URI: The URI on which the IP-HTTPS server will accept incoming IP-HTTPS connections. IP-HTTPS clients must be configured with this address in order to use it.

State: Specifies the current state of the IP-HTTPS client. The possible states are:

Enabled: In this state, the IP-HTTPS client waits for the pre-conditions specified in Section [3.1.3](#) to happen. These preconditions transition the IP-HTTPS client to the **LinkDown** state.

LinkDown: In this state, the IP-HTTPS client reads the configuration and tries to establish an HTTPS connection to the server. Once the HTTPS connection is established, the IP-HTTPS client transitions to the **Established** state.

Established: In this state the IP-HTTPS client establishes a bidirectional HTTP stream. It achieves this by sending an HTTP request message to the server and waiting for a HTTP response. Upon receiving a successful HTTP response, it transitions to the **LinkUp** state.

LinkUp: In this state the IP-HTTPS link is up and usable for sending and receiving IPv6 traffic.

Disabled: A higher-layer trigger (such as a user action to disable IP-HTTPS) transitions the IP-HTTPS client to the disabled state. IP-HTTPS operations are not possible in this state.

Further details about state processing, handling error conditions, and state transitions are specified below.

3.1.2 Timers

3.1.2.1 Reconnect Timer

To work around transient failures (for example, lack of resources on the local machine, on the network, proxy, or at the IP-HTTPS server), the IP-HTTPS clients SHOULD use a reconnect timer to retry an unsuccessful HTTPS connection or to reestablish a successful HTTPS connection that was terminated abnormally.

The reconnection attempts follow an exponential back-off algorithm. The first failure marks the beginning of the reconnection algorithm. Upon the first failure, the reconnection timer is started for an initial timeout of 30 seconds. Each subsequent failure starts a reconnection timer with a timeout set to the time elapsed since the beginning of the algorithm, up to a maximum timeout of 15 minutes. Following this algorithm, reconnection attempts will be made after the following intervals since the first failure, where T represents the time taken for the connection attempt to fail.

- 30 seconds
- $30 + T$ seconds
- $60 + 2T$ seconds
- $120 + 3T$ seconds
- ... and so on, up to 15 minutes.
- Every 15 minutes thereafter.

A successful HTTPS connection resets the timer.

It is possible that other timing logic is dictated by the other protocols upon which the IP-HTTPS Protocol relies.

3.1.3 Initialization

If the IP-HTTPS link is **Enabled** (section [3.1.4.1](#)), the IP-HTTPS link **MUST** be initialized when any of the following conditions are met.

- Client detects that it is behind an HTTP proxy.
- Client is detected behind a port/protocol blocking firewall.
- Administrative policy requires the IP-HTTPS link to be always on.
- User chooses to bring up the IP-HTTPS link.

Initialization of the client involves the client reading the **URI** of the server.

3.1.4 Higher-Layer Triggered Events

The following events are triggered by an end user or by administrative policy.

3.1.4.1 Enable IP-HTTPS Link

The Enable IP-HTTPS Link event transitions the client from the **Disabled** to **Enabled** state, making initialization possible as defined in section [3.1.3](#).

3.1.4.2 Disable IP-HTTPS Link

The Disable IP-HTTPS Link event transitions the client to the **Disabled** state. The underlying transport protocols tear down their state. No further IP-HTTPS operations are possible while in this state.

3.1.5 Processing Events and Sequencing Rules

The IP-HTTPS client waits for the pre-conditions specified in section [3.1.3](#) to happen and triggers the following events.

3.1.5.1 Establishing the HTTPS Connection

Immediately following initialization per section [3.1.3](#), the IP-HTTPS client **MUST** establish an HTTPS connection to the IP-HTTPS server (per the configuration in section [3.1.3](#)), as defined in [\[RFC1945\]](#), [\[RFC2616\]](#), [\[RFC2818\]](#), [\[RFC4346\]](#), and [\[SSLPROXY\]](#).

3.1.5.2 Bringing the IP-HTTPS Link Up

After the HTTPS connection is established, the IP-HTTPS client **MUST** send an HTTP POST request to the configured **URI**. The POST request **SHOULD** use content length encoding [\[RFC2616\]](#), and the value **SHOULD** be set to 18446744073709551615. After receiving a successful HTTP response, the IP-HTTPS client transitions to the **LinkUp** state.

The IP-HTTPS link is now usable to send and receive IPv6 traffic (including neighbor and router discovery). The default MTU for the IP-HTTPS link is set to the minimum IPv6 MTU of 1280 [\[RFC2460\]](#).

Sections [3.1.5.3](#) and [3.1.5.4](#) apply when the IP-HTTPS client is in **LinkUp** state.

3.1.5.3 Data Transfer

Framing in the IP-HTTPS Protocol relies exclusively on the length of the IPv6 header.

Outgoing IPv6 packets MUST be sent within the HTTP request entity body, and incoming IPv6 packets MUST be received as HTTP response entity body. The HTTP entity body is received as a byte stream; therefore, the IP-HTTPS client MUST delimit the IPv6 packet boundaries using the Payload Length field in the IPv6 header. The IP-HTTPS client MUST discard non IPv6 packets.

3.1.5.4 Error Handling

Upon loss of connectivity in any of the underlying transports that IP-HTTPS client relies on (section [2.1](#)), the client MUST attempt to establish the HTTPS connection as specified above.

3.1.6 Timer Events

Re-connect timer: When this timer expires, the IP-HTTPS tunnel MUST be re-established, as described in [3.1.5.2](#).

3.1.7 Other Local Events

None.

3.2 IP-HTTPS Server Details

3.2.1 Abstract Data Model

This section describes a conceptual model of possible data organization that an implementation maintains to participate in this protocol. The described organization is provided to facilitate the explanation of how the protocol behaves. This document does not mandate that implementations adhere to this model as long as their external behavior is consistent with that described in this document.

URI: The URI on which the IP-HTTPS server will accept incoming IP-HTTPS connections.

State: Specifies the current state of the IP-HTTPS server. The possible states are:

Listen: In this state, the IP-HTTPS server enables its HTTP stack to listen on the configured **URI** waiting for IP-HTTPS clients to connect. When an IP-HTTPS client successfully establishes an HTTPS connection, the IP-HTTPS server moves to the **Accept** state.

Accept: In this state, the IP-HTTPS server tries to move the IP-HTTPS client to the **LinkUp** state by responding to the client's HTTP request message. Upon successfully completing the request and response exchange with the client, the IP-HTTPS server moves back to the **Listen** state.

Disabled: IP-HTTPS operations are not possible in this state. The IP-HTTPS server transitions to this state based on user actions or administrative policy.

Further details about state processing, handling error conditions, and state transitions are specified below.

3.2.2 Timers

The IP-HTTPS Protocol does not introduce any new timers. It is possible that other timing logic is dictated by the other protocols upon which IP-HTTPS relies.

3.2.3 Initialization

The IP-HTTPS server MUST transition from **Disabled** to **Listen** state by reading the **URI**. The IP-HTTPS link at the server is considered to be up. The server SHOULD act as a router between its IP-HTTPS clients and other networks it is connected to.

The IP-HTTPS server is in **Listen** state and is usable for sending and receiving IPv6 traffic. In the **Listen** state IP-HTTPS server uses neighbor discovery [\[RFC4861\]](#) to maintain the link to which the IP-HTTPS clients connect. The default MTU for the IP-HTTPS link is set to the minimum IPv6 MTU, which is 1280 per [\[RFC2460\]](#).

3.2.4 Higher-Layer Triggered Events

The following events are triggered by an end user or administrative policy.

3.2.4.1 Enable IP-HTTPS Link

This transitions the server from **Disabled** to **Listen** state, making initialization possible as defined in section [3.2.3](#).

3.2.4.2 Disable IP-HTTPS Link

This transitions the server to the **Disabled** state. The underlying transport protocols tear down their state. No further IP-HTTPS operations are possible while in this state.

3.2.5 Processing Events and Sequencing Rules

The IP-HTTPS server stays in the **Listen** state waiting for clients to connect to it. When a client tries to connect to the server the following events are triggered.

3.2.5.1 Accepting IP-HTTPS Clients

When a client successfully establishes an HTTPS connection to the configured server **URI**, the server then moves to the **Accept** state. After the HTTP request and response exchange with the client completes, the server accepts a new IP-HTTPS client and moves back to the **Listen** state

Sections [3.2.5.2](#) and [3.2.5.3](#) apply when the IP-HTTPS server is in the **Listen** state.

3.2.5.2 Data Transfer

Framing in the IP-HTTPS Protocol relies exclusively on the length of the IPv6 header.

Outgoing IPv6 packets are sent within the HTTP response entity body. HTTP entity body is received as a byte stream, therefore, the IP-HTTPS server MUST delimit the IPv6 packet boundaries using the Payload Length field in the IPv6 header. The IP-HTTPS server MUST discard non IPv6 packets and SHOULD terminate the IP-HTTPS session with the offending client.

3.2.5.3 Error Handling

When any of the underlying transport connections to an IP-HTTPS client terminate, the IP-HTTPS server tears down the state for that client and remains in or transitions to **Listen** state.

3.2.6 Timer Events

None.

3.2.7 Other Local Events

None.

4 Protocol Examples

These examples assume HTTPS is used as transport for IP-HTTPS.

4.1 Packet Flow and Connection Establishment

The following diagram shows packet flows between an IP-HTTPS client, IP-HTTPS server, and an **IP-HTTPS endpoint**.

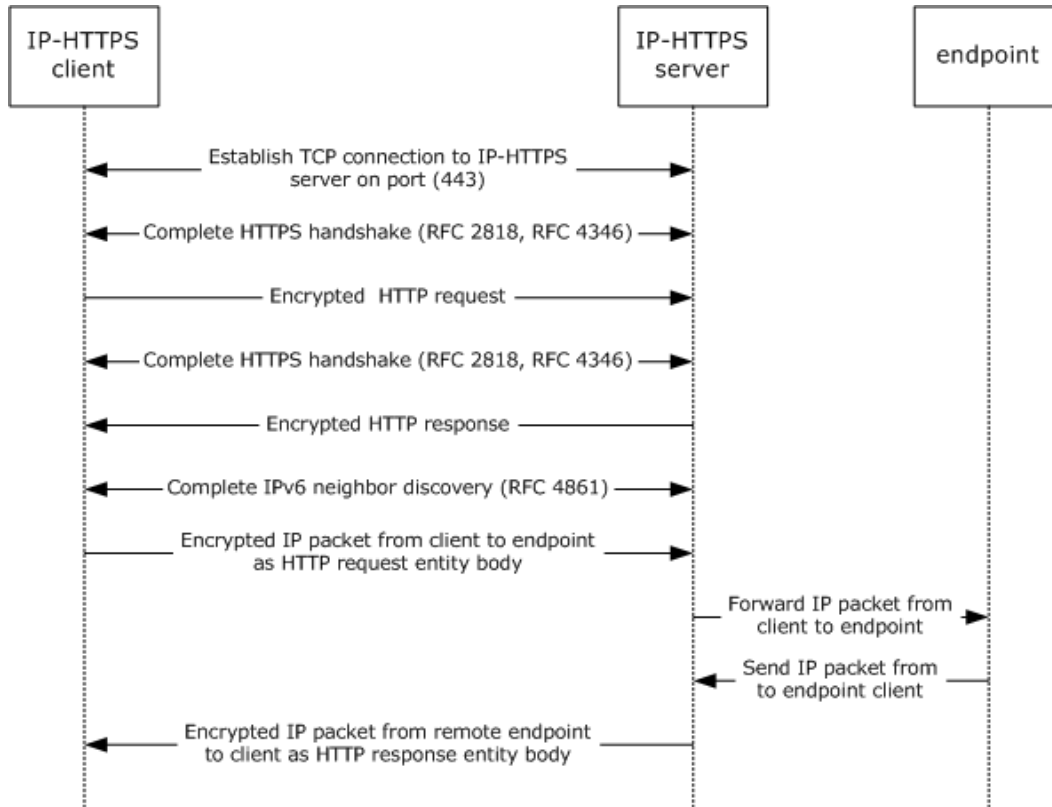


Figure 7: IP-HTTPS packet flow

When an IP-HTTPS client is behind an HTTP proxy, the client first establishes a tunnel using the CONNECT method, as described in [\[SSLPROXY\]](#). Subsequent message exchanges are identical to the above diagram and happen over the established tunnel.

The HTTP request sent to the IP-HTTPS server is described below.

```
POST /IPTLS HTTP/1.1 \r\n
Content-Length: 18446744073709551615 \r\n
Host: <IP-address of IP-HTTPS server> \r\n
\r\n
```

The HTTP response sent from the IP-HTTPS server to the IP-HTTPS client is described below.

```
HTTP/1.1 200 OK \r\n
Server: Microsoft-HTTPAPI/2.0 \r\n
Date: Sun, 10 Aug 2008 03:51:52 GMT \r\n
\r\n
```

4.2 Attack Scenarios

4.2.1 Unauthorized Client Connecting to an IP-HTTPS Server

In this scenario, an unauthorized attacker poses as a valid IP-HTTPS client and tries to connect to a valid IP-HTTPS server. The HTTPS connection is not allowed to go through, as the attacker will not have a valid client certificate to present to the IP-HTTPS server.

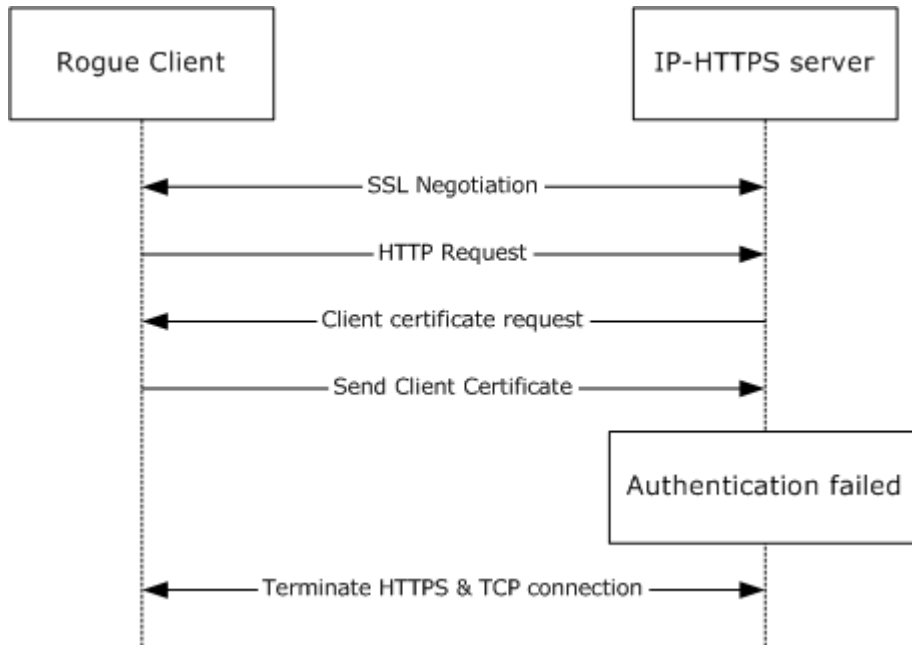


Figure 8: Unauthorized IP-HTTPS client

4.2.2 Unauthorized IP-HTTPS Server Accepting Connections from a Genuine IP-HTTPS Client

In this scenario, a valid IP-HTTPS client is redirected by an attacker to an unauthorized IP-HTTPS server (for example, by DNS poisoning). In this scenario, the HTTPS connection is terminated by the client when the client is unable to validate the server's identity using its certificate. It is recommended that the IP-HTTPS client validate the certificate per normal TLS certificate validation procedures.

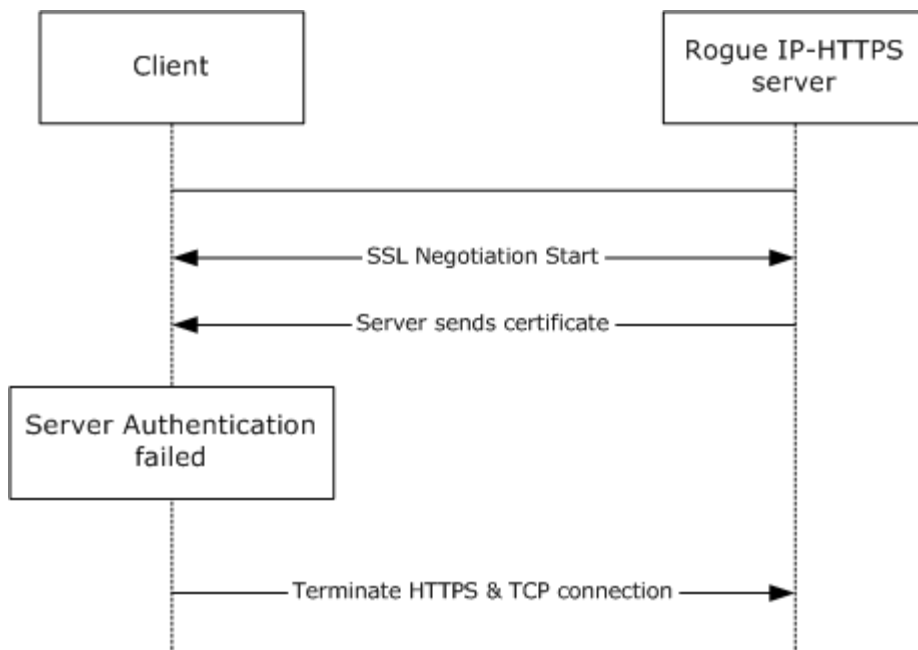


Figure 9: Unauthorized IP-HTTPS Server

4.2.3 Man in the Middle

In this scenario, an attacker poses as a man in the middle (MITM). For example, an MITM could be using a rogue wireless access point in a wireless-enabled enterprise environment.

This attack is prevented by requiring mutual authentication using certificates at the HTTPS layers.

5 Security

5.1 Security Considerations for Implementers

IP-HTTPS does not implement any form of security mechanism. It can use security mechanisms implemented by HTTPS (TLS), as described in [\[RFC2818\]](#) and [\[RFC4346\]](#).

As a transport for IP-HTTPS, HTTP provides none of the security services offered by HTTPS. Accordingly, when using HTTP as a transport, it is recommended that the IP-HTTPS payload be protected by some other means. Additionally, some proxies are known to inspect HTTP exchanges, and prevent communications when these do not appear to be normal Web traffic. Because of these security and operational considerations, it is recommended that IP-HTTPS use HTTPS as a transport.

When the IP-HTTPS server is configured to use the HTTP transport, it is possible to use it in conjunction with another device (such as a reverse proxy or an SSL/TLS load balancer), which terminates the HTTPS connection from the client on behalf of the server. In this case, the security mechanisms are ensured by that device. There is an implicit relationship of trust between the IP-HTTPS server and the device that terminates HTTPS on its behalf.

5.2 Index of Security Parameters

IP-HTTPS assumes that security has already been negotiated by the HTTPS/TLS layers, as described in section [\(1.5\)](#).

Security Parameter	Section
Mutual authentication, Hashing/Encryption algorithms, and so on.	1.5

6 Appendix A: Product Behavior

The information in this specification is applicable to the following Microsoft products or supplemental software. References to product versions include released service packs:

- Windows® 7 operating system
- Windows Server® 2008 R2 operating system

Exceptions, if any, are noted below. If a service pack or Quick Fix Engineering (QFE) number appears with the product version, behavior changed in that service pack or QFE. The new behavior also applies to subsequent service packs of the product unless otherwise specified. If a product edition appears with the product version, behavior is different in that product edition.

Unless otherwise specified, any statement of optional behavior in this specification that is prescribed using the terms SHOULD or SHOULD NOT implies product behavior in accordance with the SHOULD or SHOULD NOT prescription. Unless otherwise specified, the term MAY implies that the product does not follow the prescription.

7 Change Tracking

No table of changes is available. The document is either new or has had no changes since its last release.

8 Index

A

Abstract data model
[client](#) 12
[server](#) 15
[Applicability](#) 8
Attack scenarios
[man in the middle](#) 20
[unauthorized client connecting to server](#) 19
[unauthorized server accepting connections from genuine client](#) 19

C

[Capability negotiation](#) 8
[Change tracking](#) 23
Client
[abstract data model](#) 12
higher-layer triggered events
[Disable IP-HTTPS Link](#) 14
[Enable IP-HTTPS Link](#) 14
[overview](#) 14
[initialization](#) 14
[local events](#) 15
message processing
[bringing IP-HTTPS link up](#) 14
[data transfer](#) 15
[error handling](#) 15
[establishing HTTPS connection](#) 14
[overview](#) 14
[overview](#) 11
sequencing rules
[bringing IP-HTTPS link up](#) 14
[data transfer](#) 15
[error handling](#) 15
[establishing HTTPS connection](#) 14
[overview](#) 14
[timer events](#) 15
timers
[Reconnect](#) 13
[unauthorized connecting to server](#) 19

D

Data model - abstract
[client](#) 12
[server](#) 15
Data transfer
[client](#) 15
[server](#) 16
Disable IP-HTTPS Link event
[client](#) 14
[server](#) 16

E

Enable IP-HTTPS Link event
[client](#) 14
[server](#) 16

Error handling
[client](#) 15
[server](#) 16
Examples
attack scenarios
[man in the middle](#) 20
[unauthorized client connecting to server](#) 19
[unauthorized server accepting connections from genuine client](#) 19
[packet flow and connection establishment](#) 18

F

[Fields - vendor-extensible](#) 8

G

[Glossary](#) 5

H

Higher-layer triggered events
client
[Disable IP-HTTPS Link event](#) 14
[Enable IP-HTTPS Link event](#) 14
[overview](#) 14
server
[Disable IP-HTTPS Link event](#) 16
[Enable IP-HTTPS Link event](#) 16
[overview](#) 16
[HTTPS connection - establishing](#) 14

I

[Implementer - security considerations](#) 21
[Index of security parameters](#) 21
[Informative references](#) 6
Initialization
[client](#) 14
[server](#) 16
[Introduction](#) 5
[IP-HTTPS link - bringing up](#) 14

L

Local events
[client](#) 15
[server](#) 17

M

[Man-in-the-middle attack scenario](#) 20
Message processing
client
[bringing IP-HTTPS link up](#) 14
[data transfer](#) 15
[error handling](#) 15
[establishing HTTPS connection](#) 14
[overview](#) 14

- server
 - [accepting clients](#) 16
 - [data transfer](#) 16
 - [error handling](#) 16
 - [overview](#) 16
- Messages
 - [syntax](#) 10
 - [transport](#) 9
- N**
- [Normative references](#) 5
- O**
- [Overview \(synopsis\)](#) 6
- P**
- [Packet flow and connection establishment example](#) 18
- [Parameters - security index](#) 21
- [Preconditions](#) 7
- [Prerequisites](#) 7
- [Product behavior](#) 22
- R**
- [Reconnect timer](#) 13
- References
 - [informative](#) 6
 - [normative](#) 5
- [Relationship to other protocols](#) 6
- S**
- Security
 - [implementer considerations](#) 21
 - [parameter index](#) 21
- Sequencing rules
 - client
 - [bringing IP-HTTPS link up](#) 14
 - [data transfer](#) 15
 - [error handling](#) 15
 - [establishing HTTPS connection](#) 14
 - [overview](#) 14
 - server
 - [accepting clients](#) 16
 - [data transfer](#) 16
 - [error handling](#) 16
 - [overview](#) 16
- Server
 - [abstract data model](#) 15
 - higher-layer triggered events
 - [Disable IP-HTTPS Link](#) 16
 - [Enable IP-HTTPS Link](#) 16
 - [overview](#) 16
 - [initialization](#) 16
 - [local events](#) 17
 - message processing
 - [accepting clients](#) 16
 - [data transfer](#) 16
 - [error handling](#) 16
 - [overview](#) 16
 - sequencing rules
 - [accepting clients](#) 16
 - [data transfer](#) 16
 - [error handling](#) 16
 - [overview](#) 16
 - [timer events](#) 16
 - [timers](#) 15
 - [unauthorized accepting connections from genuine client](#) 19
 - [Standards assignments](#) 8
 - [Syntax](#) 10
- T**
- Timer events
 - [client](#) 15
 - [server](#) 16
- Timers
 - client
 - [Reconnect timer](#) 13
 - [server](#) 15
 - [Tracking changes](#) 23
 - [Transport](#) 9
- Triggered events - higher-layer
 - client
 - [Disable IP-HTTPS Link event](#) 14
 - [Enable IP-HTTPS Link event](#) 14
 - [overview](#) 14
 - server
 - [Disable IP-HTTPS Link event](#) 16
 - [Enable IP-HTTPS Link event](#) 16
 - [overview](#) 16
- V**
- [Vendor-extensible fields](#) 8
- [Versioning](#) 8