Internet Engineering Task Force (IETF)
Request for Comments: 7838
Category: Standards Track
ISSN: 2070-1721

M. Nottingham
Akamai
P. McManus
Mozilla
J. Reschke
greenbytes
April 2016

HTTP Alternative Services

Abstract

This document specifies "Alternative Services" for HTTP, which allow an origin's resources to be authoritatively available at a separate network location, possibly accessed with a different protocol configuration.

Status of This Memo

This is an Internet Standards Track document.

This document is a product of the Internet Engineering Task Force (IETF). It represents the consensus of the IETF community. It has received public review and has been approved for publication by the Internet Engineering Steering Group (IESG). Further information on Internet Standards is available in Section 2 of RFC 5741[1].

Information about the current status of this document, any errata, and how to provide feedback on it may be obtained at http://www.rfc-editor.org/info/rfc7838[2].

Copyright Notice

[1] https://www.rfc-editor.org/rfc/rfc5741.html#section-2
[2] http://www.rfc-editor.org/info/rfc7838
[3] http://trustee.ietf.org/license-info

Table of Contents

1. Introduction

HTTP [RFC7230] conflates the identification of resources with their location. In other words, "http://" and "https://" URIs are used to both name and find things to interact with.

In some cases, it is desirable to separate identification and location in HTTP; keeping the same identifier for a resource, but interacting with it at a different location on the network.

For example:

- An origin server might wish to redirect a client to a different server when it is under load, or it has found a server in a location that is more local to the client.
- An origin server might wish to offer access to its resources using a new protocol, such as HTTP/2 [RFC7540], or one using improved security, such as Transport Layer Security (TLS) [RFC5246].
- An origin server might wish to segment its clients into groups of capabilities, such as those supporting Server Name Indication (SNI) (Section 3 of [RFC6066]), for operational purposes.

This specification defines a new concept in HTTP, "Alternative Services", that allows an origin server to nominate additional means of interacting with it on the network. It defines a general framework for this in Section 2, along with specific mechanisms for advertising their existence using HTTP header fields (Section 3) or HTTP/2 frames (Section 4), plus a way to indicate that an alternative service was used (Section 5).

It also endorses the status code 421 (Misdirected Request) (Section 6) that origin servers or their nominated alternatives can use to indicate that they are not authoritative for a given origin, in cases where the wrong location is used.

1.1. Notational Conventions

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC2119].

This document uses the Augmented BNF defined in [RFC5234] and updated by [RFC7405] along with the "#rule" extension defined in Section 7 of [RFC7230]. The rules below are defined in [RFC5234], [RFC7230], and [RFC7234]:

```
OWS           = <OWS, see [RFC7230], Section 3.2.3>
delta-seconds = <delta-seconds; see [RFC7234], Section 1.2.1>
port          = <port, see [RFC7230], Section 2.7>
quoted-string = <quoted-string, see [RFC7230], Section 3.2.6>
token         = <token, see [RFC7230], Section 3.2.6>
uri-host      = <uri-host, see [RFC7230], Section 2.7>
```

2. Alternative Services Concepts

This specification defines a new concept in HTTP, the "*Alternative Service*". When an origin [RFC6454] has resources that are accessible through a different protocol/host/port combination, it is said to have an alternative service available.

An alternative service can be used to interact with the resources on an origin server at a separate location on the network, possibly using a different protocol configuration. Alternative services are considered authoritative for an origin's resources, in the sense of [RFC7230], Section 9.1.

For example, an origin:

```
("http", "www.example.com", "80")
```

might declare that its resources are also accessible at the alternative service:

```
("h2", "new.example.com", "81")
```

By their nature, alternative services are explicitly at the granularity of an origin; they cannot be selectively applied to resources within an origin.

Alternative services do not replace or change the origin for any given resource; in general, they are not visible to the software "above" the access mechanism. The alternative service is essentially alternative routing information that can also be used to reach the origin in the same way that DNS CNAME or SRV records define routing information at the name resolution level. Each origin maps to a set of these routes — the default route is derived from the origin itself and the other routes are introduced based on alternative-service information.

Furthermore, it is important to note that the first member of an alternative service tuple is different from the "scheme" component of an origin; it is more specific, identifying not only the major version of the protocol being used, but potentially the communication options for that protocol as well.

This means that clients using an alternative service can change the host, port, and protocol that they are using to fetch resources, but these changes MUST NOT be propagated to the application that is using HTTP; from that standpoint, the URI being accessed and all information derived from it (scheme, host, and port) are the same as before.

Importantly, this includes its security context; in particular, when TLS [RFC5246] is used to authenticate, the alternative service will need to present a certificate for the origin's host name, not that of the alternative. Likewise, the Host header field ([RFC7230], Section 5.4) is still derived from the origin, not the alternative service (just as it would if a CNAME were being used).

The changes MAY, however, be made visible in debugging tools, consoles, etc.

Formally, an alternative service is identified by the combination of:

- An Application Layer Protocol Negotiation (ALPN) protocol name, as per [RFC7301]
- A host, as per [RFC3986], Section 3.2.2
- A port, as per [RFC3986], Section 3.2.3

The ALPN protocol name is used to identify the application protocol or suite of protocols used by the alternative service. Note that for the purpose of this specification, an ALPN protocol name implicitly includes TLS in the suite of protocols it identifies, unless specified otherwise in its definition. In particular, the ALPN name "http/1.1", registered by Section 6 of [RFC7301], identifies HTTP/1.1 over TLS.

Additionally, each alternative service MUST have a freshness lifetime, expressed in seconds (see Section 2.2).

There are many ways that a client could discover the alternative service(s) associated with an origin. This document describes two such mechanisms: the "Alt-Svc" HTTP header field (Section 3) and the "ALTSVC" HTTP/2 frame type (Section 4).

The remainder of this section describes requirements that are common to alternative services, regardless of how they are discovered.

2.1. Host Authentication

Clients MUST have reasonable assurances that the alternative service is under control of and valid for the whole origin. This mitigates the attack described in Section 9.2.

For the purposes of this document, "reasonable assurances" can be established through use of a TLS-based protocol with the certificate checks defined in [RFC2818]. Clients MAY impose additional criteria for establishing reasonable assurances.

For example, if the origin's host is "www.example.com" and an alternative is offered on "other.example.com" with the "h2" protocol, and the certificate offered is valid for "www.example.com", the client can use the alternative. However, if either is offered with the "h2c" protocol, the client cannot use it, because there is no mechanism (at the time of the publication of this specification) in that protocol to establish the relationship between the origin and the alternative.

2.2. Alternative Service Caching

Mechanisms for discovering alternative services also associate a freshness lifetime with them; for example, the Alt-Svc header field uses the "ma" parameter.

Clients can choose to use an alternative service instead of the origin at any time when it is considered fresh; see Section 2.4 for specific recommendations.

Clients with existing connections to an alternative service do not need to stop using it when its freshness lifetime ends; the caching mechanism is intended for limiting how long an alternative service can be used for establishing new connections, not limiting the use of existing ones.

Alternative services are fully authoritative for the origin in question, including the ability to clear or update cached alternative service entries, extend freshness lifetimes, and any other authority the origin server would have.

When alternative services are used to send a client to the most optimal server, a change in network configuration can result in cached values becoming suboptimal. Therefore, clients SHOULD remove from cache all alternative services that lack the "persist" flag with the value "1" when they detect such a change, when information about network state is available.

2.3. Requiring Server Name Indication

A client MUST NOT use a TLS-based alternative service unless the client supports TLS Server Name Indication (SNI). This supports the conservation of IP addresses on the alternative service host.

Note that the SNI information provided in TLS by the client will be that of the origin, not the alternative (as will the Host HTTP header field value).

2.4. Using Alternative Services

By their nature, alternative services are OPTIONAL: clients do not need to use them. However, it is advantageous for clients to behave in a predictable way when alternative services are used by servers, to aid in purposes like load balancing.

Therefore, if a client supporting this specification becomes aware of an alternative service, the client SHOULD use that alternative service for all requests to the associated origin as soon as it is available, provided the alternative service information is fresh (Section 2.2) and the security properties of the alternative service protocol are desirable, as compared to the existing connection. A viable alternative service is then treated in every way as the origin; this includes the ability to advertise alternative services.

If a client becomes aware of multiple alternative services, it chooses the most suitable according to its own criteria, keeping security properties in mind. For example, an origin might advertise multiple alternative services to notify clients of support for multiple versions of HTTP.

A client configured to use a proxy for a given request SHOULD NOT directly connect to an alternative service for this request, but instead route it through that proxy.

When a client uses an alternative service for a request, it can indicate this to the server using the Alt-Used header field (Section 5).

The client does not need to block requests on any existing connection; it can be used until the alternative connection is established. However, if the security properties of the existing connection are weak (for example, cleartext HTTP/1.1), then it might make sense to block until the new connection is fully available in order to avoid information leakage.

Furthermore, if the connection to the alternative service fails or is unresponsive, the client MAY fall back to using the origin or another alternative service. Note, however, that this could be the basis of a downgrade attack, thus losing any enhanced security properties of the alternative service. If the connection to the alternative service does not negotiate the expected protocol (for example, ALPN fails to negotiate h2, or an Upgrade request to h2c is not accepted), the connection to the alternative service MUST be considered to have failed.

3. The Alt-Svc HTTP Header Field

An HTTP(S) origin server can advertise the availability of alternative services to clients by adding an Alt-Svc header field to responses.

```
Alt-Svc       = clear / 1#alt-value
clear         = %s"clear"; "clear", case-sensitive
alt-value     = alternative *( OWS ";" OWS parameter )
alternative   = protocol-id "=" alt-authority
protocol-id   = token ; percent-encoded ALPN protocol name
alt-authority = quoted-string ; containing [ uri-host ] ":" port
parameter     = token "=" ( token / quoted-string )
```

The field value consists either of a list of values, each of which indicates one alternative service, or the keyword "clear".

A field value containing the special value "clear" indicates that the origin requests all alternatives for that origin to be invalidated (including those specified in the same response, in case of an invalid reply containing both "clear" and alternative services).

ALPN protocol names are octet sequences with no additional constraints on format. Octets not allowed in tokens ([RFC7230], Section 3.2.6) MUST be percent-encoded as per Section 2.1 of [RFC3986]. Consequently, the octet representing the percent character "%" (hex 25) MUST be percent-encoded as well.

In order to have precisely one way to represent any ALPN protocol name, the following additional constraints apply:

1. Octets in the ALPN protocol name MUST NOT be percent-encoded if they are valid token characters except "%", and
2. When using percent-encoding, uppercase hex digits MUST be used.

With these constraints, recipients can apply simple string comparison to match protocol identifiers.

The "alt-authority" component consists of an OPTIONAL uri-host ("host" in Section 3.2.2 of [RFC3986]), a colon (":"), and a port number.

For example:

```
Alt-Svc: h2=":8000"
```

This indicates the "h2" protocol ([RFC7540]) on the same host using the indicated port 8000.

An example involving a change of host:

```
Alt-Svc: h2="new.example.org:80"
```

This indicates the "h2" protocol on the host "new.example.org", running on port 80. Note that the "quoted-string" syntax needs to be used because ":" is not an allowed character in "token".

Examples for protocol name escaping:

ALPN protocol name	protocol-id	Note
h2	h2	No escaping needed
w=x:y#z	w%3Dx%3Ay#z	"=" and ":" escaped
x%y	x%25y	"%" needs escaping

Alt-Svc MAY occur in any HTTP response message, regardless of the status code. Note that recipients of Alt-Svc can ignore the header field (and are required to in some situations; see Sections 2.1 and 6).

The Alt-Svc field value can have multiple values:

```
Alt-Svc: h2="alt.example.com:8000", h2=":443"
```

When multiple values are present, the order of the values reflects the server's preference (with the first value being the most preferred alternative).

The value(s) advertised by Alt-Svc can be used by clients to open a new connection to an alternative service. Subsequent requests can start using this new connection immediately or can continue using the existing connection while the new connection is created.

When using HTTP/2 ([RFC7540]), servers SHOULD instead send an ALTSVC frame (Section 4). A single ALTSVC frame can be sent for a connection; a new frame is not needed for every request. Note that, despite this recommendation, Alt-Svc header fields remain valid in responses delivered over HTTP/2.

Each "alt-value" is followed by an OPTIONAL semicolon-separated list of additional parameters, each such "parameter" comprising a name and a value.

This specification defines two parameters: "ma" and "persist", defined in Section 3.1. Unknown parameters MUST be ignored. That is, the values (alt-value) they appear in MUST be processed as if the unknown parameter was not present.

New parameters can be defined in extension specifications (see Section 7.3 for registration details).

Note that all field elements that allow "quoted-string" syntax MUST be processed as per Section 3.2.6 of [RFC7230].

3.1. Caching Alt-Svc Header Field Values

When an alternative service is advertised using Alt-Svc, it is considered fresh for 24 hours from generation of the message. This can be modified with the "ma" (max-age) parameter.

Syntax:

```
ma = delta-seconds; see [RFC7234], Section 1.2.1
```

The delta-seconds value indicates the number of seconds since the response was generated for which the alternative service is considered fresh.

```
Alt-Svc: h2=":443"; ma=3600
```

See Section 4.2.3 of [RFC7234] for details on determining the response age.

For example, a response:

```
HTTP/1.1 200 OK
Content-Type: text/html
Cache-Control: max-age=600
Age: 30
Alt-Svc: h2=":8000"; ma=60
```

indicates that an alternative service is available and usable for the next 60 seconds. However, the response has already been cached for 30 seconds (as per the Age header field value); therefore, the alternative service is only fresh for the 30 seconds from when this response was received, minus estimated transit time.

Note that the freshness lifetime for HTTP caching (here, 600 seconds) does not affect caching of Alt-Svc values.

When an Alt-Svc response header field is received from an origin, its value invalidates and replaces all cached alternative services for that origin.

By default, cached alternative services will be cleared when the client detects a network change. Alternative services that are intended to be longer lived (such as those that are not specific to the client access network) can carry the "persist" parameter with a value "1" as a hint that the service is potentially useful beyond a network configuration change.

Syntax:

```
persist = "1"
```

For example:

```
Alt-Svc: h2=":443"; ma=2592000; persist=1
```

This specification only defines a single value for "persist". Clients MUST ignore "persist" parameters with values other than "1".

See Section 2.2 for general requirements on caching alternative services.

4. The ALTSVC HTTP/2 Frame

The ALTSVC HTTP/2 frame ([RFC7540], Section 4) advertises the availability of an alternative service to an HTTP/2 client.

The ALTSVC frame is a non-critical extension to HTTP/2. Endpoints that do not support this frame will ignore it (as per the extensibility rules defined in Section 4.1 of [RFC7540]).

An ALTSVC frame from a server to a client on a stream other than stream 0 indicates that the conveyed alternative service is associated with the origin of that stream.

An ALTSVC frame from a server to a client on stream 0 indicates that the conveyed alternative service is associated with the origin contained in the Origin field of the frame. An association with an origin that the client does not consider authoritative for the current connection MUST be ignored.

The ALTSVC frame type is 0xa (decimal 10).

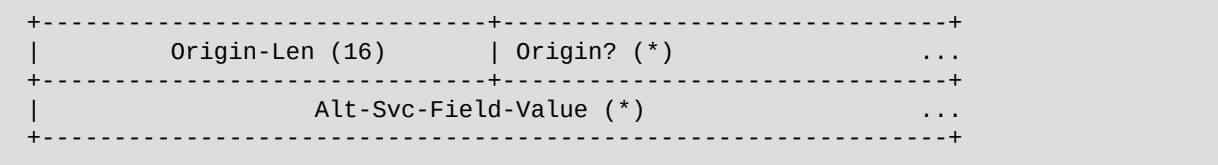

Figure 1: ALTSVC Frame Payload

The ALTSVC frame contains the following fields:

Origin-Len: An unsigned, 16-bit integer indicating the length, in octets, of the Origin field.

Origin: An OPTIONAL sequence of characters containing the ASCII serialization of an origin ([RFC6454], Section 6.2) to which the alternative service is applicable.

Alt-Svc-Field-Value: A sequence of octets (length determined by subtracting the length of all preceding fields from the frame length) containing a value identical to the Alt-Svc field value defined in Section 3 (ABNF production "Alt-Svc").

The ALTSVC frame does not define any flags.

The ALTSVC frame is intended for receipt by clients. A device acting as a server MUST ignore it.

An ALTSVC frame on stream 0 with empty (length 0) "Origin" information is invalid and MUST be ignored. An ALTSVC frame on a stream other than stream 0 containing non-empty "Origin" information is invalid and MUST be ignored.

The ALTSVC frame is processed hop-by-hop. An intermediary MUST NOT forward ALTSVC frames, though it can use the information contained in ALTSVC frames in forming new ALTSVC frames to send to its own clients.

Receiving an ALTSVC frame is semantically equivalent to receiving an Alt-Svc header field. As a result, the ALTSVC frame causes alternative services for the corresponding origin to be replaced. Note that it would be unwise to mix the use of Alt-Svc header fields with the use of ALTSVC frames, as the sequence of receipt might be hard to predict.

5. The Alt-Used HTTP Header Field

The Alt-Used header field is used in requests to identify the alternative service in use, just as the Host header field (Section 5.4 of [RFC7230]) identifies the host and port of the origin.

```
Alt-Used     = uri-host [ ":" port ]
```

Alt-Used is intended to allow alternative services to detect loops, differentiate traffic for purposes of load balancing, and generally to ensure that it is possible to identify the intended destination of traffic, since introducing this information after a protocol is in use has proven to be problematic.

When using an alternative service, clients SHOULD include an Alt-Used header field in all requests.

For example:

```
GET /thing HTTP/1.1
Host: origin.example.com
Alt-Used: alternate.example.net
```

6. The 421 (Misdirected Request) HTTP Status Code

The 421 (Misdirected Request) status code is defined in Section 9.1.2 of [RFC7540] to indicate that the current server instance is not authoritative for the requested resource. This can be used to indicate that an alternative service is not authoritative; see Section 2).

Clients receiving 421 (Misdirected Request) from an alternative service MUST remove the corresponding entry from its alternative service cache (see Section 2.2) for that origin. Regardless of the idempotency of the request method, they MAY retry the request, either at another alternative server, or at the origin.

An Alt-Svc header field in a 421 (Misdirected Request) response MUST be ignored.

7. IANA Considerations

7.1. Header Field Registrations

HTTP header fields are registered within the "Message Headers" registry maintained at <https://www.iana.org/assignments/message-headers/>.

This document defines the following HTTP header fields, so their associated registry entries have been added according to the permanent registrations below (see [BCP90]):

Header Field Name	Protocol	Status	Reference
Alt-Svc	http	standard	Section 3
Alt-Used	http	standard	Section 5

The change controller is: "IETF (iesg@ietf.org) -- Internet Engineering Task Force".

7.2. The ALTSVC HTTP/2 Frame Type

This document registers the ALTSVC frame type in the "HTTP/2 Frame Type" registry ([RFC7540], Section 11.2).

Frame Type: ALTSVC

Code: 0xa

Specification: Section 4 of this document

7.3. Alt-Svc Parameter Registry

The "Hypertext Transfer Protocol (HTTP) Alt-Svc Parameter Registry" defines the name space for parameters. It has been created and will be maintained at <http://www.iana.org/assignments/http-alt-svc-parameters>.

7.3.1. Procedure

A registration MUST include the following fields:

- Parameter Name
- Pointer to specification text

Values to be added to this name space require Expert Review (see [RFC5226], Section 4.1).

7.3.2. Registrations

The "Hypertext Transfer Protocol (HTTP) Alt-Svc Parameter Registry" has been populated with the registrations below:

Alt-Svc Parameter	Reference
ma	Section 3.1
persist	Section 3.1

8. Internationalization Considerations

An internationalized domain name that appears in either the header field (Section 3) or the HTTP/2 frame (Section 4) MUST be expressed using A-labels ([RFC5890], Section 2.3.2.1).

9. Security Considerations

9.1. Changing Ports

Using an alternative service implies accessing an origin's resources on an alternative port, at a minimum. Therefore, an attacker that can inject alternative services and listen at the advertised port is able to hijack an origin. On certain servers, it is normal for users to be able to control some personal pages available on a shared port and also to accept requests on less-privileged ports.

For example, an attacker that can add HTTP response header fields to some pages can redirect traffic for an entire origin to a different port on the same host using the Alt-Svc header field; if that port is under the attacker's control, they can thus masquerade as the HTTP server.

This risk is mitigated by the requirements in Section 2.1.

On servers, this risk can also be reduced by restricting the ability to advertise alternative services, and restricting who can open a port for listening on that host.

9.2. Changing Hosts

When the host is changed due to the use of an alternative service, this presents an opportunity for attackers to hijack communication to an origin.

For example, if an attacker can convince a user agent to send all traffic for "innocent.example.org" to "evil.example.com" by successfully associating it as an alternative service, they can masquerade as that origin. This can be done locally (see mitigations in Section 9.1) or remotely (e.g., by an intermediary as a man-in-the-middle attack).

This is the reason for the requirement in Section 2.1 that clients have reasonable assurances that the alternative service is under control of and valid for the whole origin; for example, presenting a certificate for the origin proves that the alternative service is authorized to serve traffic for the origin.

Note that this assurance is only as strong as the method used to authenticate the alternative service. In particular, when TLS authentication is used to do so, there are well-known exploits to make an attacker's certificate appear as legitimate.

Alternative services could be used to persist such an attack. For example, an intermediary could man-in-the-middle TLS-protected communication to a target and then direct all traffic to an alternative service with a large freshness lifetime so that the user agent still directs traffic to the attacker even when not using the intermediary.

Implementations MUST perform any certificate-pinning validation (such as [RFC7469]) on alternative services just as they would on direct connections to the origin. Implementations might also choose to add other requirements around which certificates are acceptable for alternative services.

9.3. Changing Protocols

When the ALPN protocol is changed due to the use of an alternative service, the security properties of the new connection to the origin can be different from that of the "normal" connection to the origin, because the protocol identifier itself implies this.

For example, if an "https://" URI has a protocol advertised that does not use some form of end-to-end encryption (most likely, TLS), this violates the expectations for security that the URI scheme implies. Therefore, clients cannot use alternative services blindly, but instead evaluate the option(s) presented to ensure that security requirements and expectations of specifications, implementations, and end users are met.

9.4. Tracking Clients Using Alternative Services

Choosing an alternative service implies connecting to a new, server-supplied host name. By using unique names, servers could conceivably track client requests. Such tracking could follow users across multiple networks, when the "persist" flag is used.

Clients that wish to prevent requests from being correlated can decide not to use alternative services for multiple requests that would not otherwise be allowed to be correlated.

In a user agent, any alternative service information MUST be removed when origin-specific data is cleared (typically, when cookies [RFC6265] are cleared).

9.5. Confusion regarding Request Scheme

Some server-side HTTP applications make assumptions about security based upon connection context; for example, equating being served upon port 443 with the use of an "https://" URI and the various security properties that implies.

This affects not only the security properties of the connection itself, but also the state of the client at the other end of it; for example, a Web browser treats "https://" URIs differently than "http://" URIs in many ways, not just for purposes of protocol handling.

Since one of the uses of Alternative Services is to allow a connection to be migrated to a different protocol and port, these applications can become confused about the security properties of a given connection, sending information (for example, cookies and content) that is intended for a secure context (such as an "https://" URI) to a client that is not treating it as one.

This risk can be mitigated in servers by using the URI scheme explicitly carried by the protocol (such as ":scheme" in HTTP/2 or the "absolute form" of the request target in HTTP/1.1) as an indication of security context, instead of other connection properties ([RFC7540], Section 8.1.2.3 and [RFC7230], Section 5.3.2).

When the protocol does not explicitly carry the scheme (as is usually the case for HTTP/1.1 over TLS), servers can mitigate this risk by either assuming that all requests have an insecure context, or by refraining from advertising alternative services for insecure schemes (for example, HTTP).

10. References

10.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <http://www.rfc-editor.org/info/rfc>.

[RFC2818] Rescorla, E., "HTTP Over TLS", RFC 2818, DOI 10.17487/RFC2818, May 2000, <http://www.rfc-editor.org/info/rfc>.

[RFC3986] Berners-Lee, T., Fielding, R., and L. Masinter, "Uniform Resource Identifier (URI): Generic Syntax", STD 66, RFC 3986, DOI 10.17487/RFC3986, January 2005, <http://www.rfc-editor.org/info/rfc>.

[RFC5226] Narten, T. and H. Alvestrand, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 5226, DOI 10.17487/RFC5226, May 2008, <http://www.rfc-editor.org/info/rfc>.

[RFC5234] Crocker, D., Ed. and P. Overell, "Augmented BNF for Syntax Specifications: ABNF", STD 68, RFC 5234, DOI 10.17487/RFC5234, January 2008, <http://www.rfc-editor.org/info/rfc>.

[RFC5890] Klensin, J., "Internationalized Domain Names for Applications (IDNA): Definitions and Document Framework", RFC 5890, DOI 10.17487/RFC5890, August 2010, <http://www.rfc-editor.org/info/rfc>.

[RFC6066] Eastlake, D., "Transport Layer Security (TLS) Extensions: Extension Definitions", RFC 6066, DOI 10.17487/RFC6066, January 2011, <http://www.rfc-editor.org/info/rfc>.

[RFC6454] Barth, A., "The Web Origin Concept", RFC 6454, DOI 10.17487/RFC6454, December 2011, <http://www.rfc-editor.org/info/rfc>.

[RFC7230] Fielding, R., Ed. and J. Reschke, Ed., "Hypertext Transfer Protocol (HTTP/1.1): Message Syntax and Routing", RFC 7230, DOI 10.17487/RFC7230, June 2014, <http://www.rfc-editor.org/info/rfc>.

[RFC7234] Fielding, R., Ed., Nottingham, M., Ed., and J. Reschke, Ed., "Hypertext Transfer Protocol (HTTP/1.1): Caching", RFC 7234, DOI 10.17487/RFC7234, June 2014, <http://www.rfc-editor.org/info/rfc>.

[RFC7301] Friedl, S., Popov, A., Langley, A., and S. Emile, "Transport Layer Security (TLS) Application-Layer Protocol Negotiation Extension", RFC 7301, DOI 10.17487/RFC7301, July 2014, <http://www.rfc-editor.org/info/rfc>.

[RFC7405] Kyzivat, P., "Case-Sensitive String Support in ABNF", RFC 7405, DOI 10.17487/RFC7405, December 2014, <http://www.rfc-editor.org/info/rfc>.

[RFC7540] Belshe, M., Peon, R., and M. Thomson, Ed., "Hypertext Transfer Protocol version 2", RFC 7540, DOI 10.17487/RFC7540, May 2015, <http://www.rfc-editor.org/info/rfc>.

10.2. Informative References

[BCP90] Klyne, G., Nottingham, M., and J. Mogul, "Registration Procedures for Message Header Fields", BCP 90, RFC 3864, September 2004, <http://www.rfc-editor.org/info/bcp>.

[RFC5246] Dierks, T. and E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.2", RFC 5246, DOI 10.17487/RFC5246, August 2008, <http://www.rfc-editor.org/info/rfc>.

[RFC6265] Barth, A., "HTTP State Management Mechanism", RFC 6265, DOI 10.17487/RFC6265, April 2011, <http://www.rfc-editor.org/info/rfc>.

[RFC7469] Evans, C., Palmer, C., and R. Sleevi, "Public Key Pinning Extension for HTTP", RFC 7469, DOI 10.17487/RFC7469, April 2015, <http://www.rfc-editor.org/info/rfc>.

Acknowledgements

Thanks to Adam Langley, Bence Beky, Chris Lonvick, Eliot Lear, Erik Nygren, Guy Podjarny, Herve Ruellan, Lucas Pardue, Martin Thomson, Matthew Kerwin, Mike Bishop, Paul Hoffman, Richard Barnes, Richard Bradbury, Stephen Farrell, Stephen Ludin, and Will Chan for their feedback and suggestions.

The Alt-Svc header field was influenced by the design of the Alternate-Protocol header field in SPDY.

Authors' Addresses

Mark Nottingham
Akamai
Email: mnot@mnot.net
URI: https://www.mnot.net/

Patrick McManus
Mozilla
Email: mcmanus@ducksong.com
URI: https://mozillians.org/u/pmcmanus/

Julian F. Reschke
greenbytes GmbH
Email: julian.reschke@greenbytes.de
URI: https://greenbytes.de/tech/webdav/