HTTP Working Group
Internet-Draft
Intended status: Standards Track
Expires: September 9, 2016

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

March 8, 2016

HTTP Alternative Services
draft-ietf-httpbis-alt-svc-14

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.

Editorial Note (To be removed by RFC Editor)

Discussion of this draft takes place on the HTTPBIS working group mailing list (ietf-http-wg@w3.org), which is archived at <https://lists.w3.org/Archives/Public/ietf-http-wg/>.

Working Group information can be found at <http://httpwg.github.io/>; source code and issues list for this draft can be found at <https://github.com/httpwg/http-extensions>.

The changes in this draft are summarized in Appendix A.

Status of This Memo

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 communication options for that protocol.

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, 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 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 the alternative service is considered fresh for.

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

See Section 4.2.3 of [RFC7234] for details of determining 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), so 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.

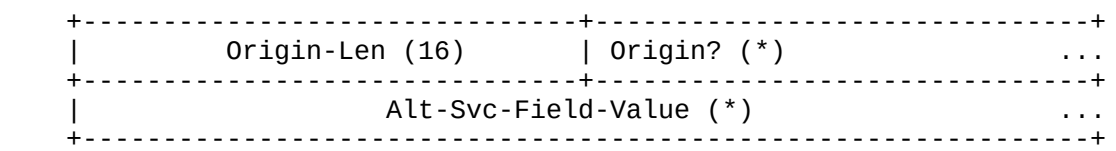
The ALTSVC frame is a non-critical extension to HTTP/2. Endpoints

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).



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) that the alternative service is applicable to.

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 indicate the identity of 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 shall be 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 Types registry ([RFC7540], Section 11.2).

Frame Type: ALTSVC

Code: 0xa

Specification: Section 4 of this document

7.3. Alt-Svc Parameter Registry

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

7.3.1. Procedure

A registration MUST include the following fields:

- Parameter Name

o 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 HTTP Alt-Svc Parameter Registry is to be 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. An attacker that can inject alternative services and listen at the advertised port is therefore 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 to 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, it 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), it violates the expectations for security that the URI scheme implies. Therefore, clients cannot blindly use alternative services, but

instead evaluate the option(s) presented to assure 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/rfc2119>.

[RFC2818] Rescorla, E., "HTTP Over TLS", RFC 2818, DOI 10.17487/RFC2818, May 2000, <http://www.rfc-editor.org/info/rfc2818>.

[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/rfc3986>.

[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/rfc5226>.

[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/rfc5234>.

[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/rfc5890>.

[RFC6066] Eastlake, D., "Transport Layer Security (TLS) Extensions: Extension Definitions", RFC 6066, DOI 10.17487/RFC6066, January 2011, <http://www.rfc-editor.org/info/rfc6066>.

[RFC6454] Barth, A., "The Web Origin Concept", RFC 6454, DOI 10.17487/RFC6454, December 2011, <http://www.rfc-editor.org/info/rfc6454>.

[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/rfc7230>.

[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/rfc7234>.

[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/rfc7301>.

[RFC7405] Kyzivat, P., "Case-Sensitive String Support in ABNF", RFC 7405, DOI 10.17487/RFC7405, December 2014, <http://www.rfc-editor.org/info/rfc7405>.

[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/rfc7540>.

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/bcp90>.

[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/rfc5246>.

[RFC6265] Barth, A., "HTTP State Management Mechanism", RFC 6265, DOI 10.17487/RFC6265, April 2011, <http://www.rfc-editor.org/info/rfc6265>.

[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/rfc7469>.

Appendix A. Change Log (to be removed by RFC Editor before publication)

A.1. Since draft-nottingham-httpbis-alt-svc-05

This is the first version after adoption of draft-nottingham-httpbis-alt-svc-05 as Working Group work item. It only contains editorial changes.

A.2. Since draft-ietf-httpbis-alt-svc-00

Selected 421 as proposed status code for "Not Authoritative".

Changed header field syntax to use percent-encoding of ALPN protocol names (<https://github.com/http2/http2-spec/issues/446>).

A.3. Since draft-ietf-httpbis-alt-svc-01

Updated HTTP/1.1 references.

Renamed "Service" to "Alt-Svc-Used" and reduced information to a flag to address fingerprinting concerns (<https://github.com/http2/http2-spec/issues/502>).

Note that ALTSVC frame is preferred to Alt-Svc header field (<https://github.com/http2/http2-spec/pull/503>).

Incorporate ALTSRV frame (<https://github.com/http2/http2-spec/pull/507>).

Moved definition of status code 421 to HTTP/2.

Partly resolved <https://github.com/httpwg/http-extensions/issues/5>.

A.4. Since draft-ietf-httpbis-alt-svc-02

Updated ALPN reference.

Resolved <https://github.com/httpwg/http-extensions/issues/2>.

A.5. Since draft-ietf-httpbis-alt-svc-03

Renamed "Alt-Svc-Used" to "Alt-Used" (<https://github.com/httpwg/http-extensions/issues/17>).

Clarify ALTSVC Origin information requirements (<https://github.com/httpwg/http-extensions/issues/19>).

Remove/tune language with respect to tracking risks (see <https://github.com/httpwg/http-extensions/issues/34>).

A.6. Since draft-ietf-httpbis-alt-svc-04

Mention tracking by alt-svc host name in Security Considerations (<https://github.com/httpwg/http-extensions/issues/36>).

"421 (Not Authoritative)" -> "421 (Misdirected Request)".

Allow the frame to carry multiple indicator and use the same payload formats for both (<https://github.com/httpwg/http-extensions/issues/37>).

A.7. Since draft-ietf-httpbis-alt-svc-05

Go back to specifying the origin in Alt-Used, but make it a "SHOULD" (<https://github.com/httpwg/http-extensions/issues/34>).

Restore Origin field in ALT-SVC frame (<https://github.com/httpwg/http-extensions/issues/38>).

A.8. Since draft-ietf-httpbis-alt-svc-06

Disallow use of alternative services when the protocol might not carry the scheme (<https://github.com/httpwg/http-extensions/issues/12>).

Align opp-sec and alt-svc (<https://github.com/httpwg/http-extensions/issues/33>).

alt svc frame on pushed (even and non-0) frame (<https://github.com/httpwg/http-extensions/issues/44>).

"browser" -> "user agent" (<https://github.com/httpwg/http-extensions/pull/61>).

ABNF for "parameter" (<https://github.com/httpwg/http-extensions/issues/65>).

Updated HTTP/2 reference.

A.9. Since draft-ietf-httpbis-alt-svc-07

Alt-Svc alternative cache invalidation (<https://github.com/httpwg/http-extensions/issues/16>).

Unexpected Alt-Svc frames (<https://github.com/httpwg/http-extensions/issues/18>).

Associating Alt-Svc header with an origin (<https://github.com/httpwg/http-extensions/issues/21>).

ALPN identifiers in Alt-Svc (<https://github.com/httpwg/http-extensions/issues/43>).

Number of alternate services used (<https://github.com/httpwg/http-extensions/issues/58>).

Proxy and .pac interaction
(<https://github.com/httpwg/http-extensions/issues/62>).

Need to define extensibility for alt-svc parameters
(<https://github.com/httpwg/http-extensions/issues/69>).

Persistence of alternates across network changes
(<https://github.com/httpwg/http-extensions/issues/71>).

Alt-Svc header with 421 status
(<https://github.com/httpwg/http-extensions/issues/75>).

Incorporate several editorial improvements suggested by Mike Bishop
(<https://github.com/httpwg/http-extensions/pull/77>,
<https://github.com/httpwg/http-extensions/pull/78>).

Alt-Svc response header field in HTTP/2 frame
(<https://github.com/httpwg/http-extensions/issues/87>).

A.10. Since draft-ietf-httpbis-alt-svc-08

Remove left over text about ext-params, applying to an earlier
version of Alt-Used (see
<https://github.com/httpwg/http-extensions/issues/34>).

Conflicts between Alt-Svc and ALPN
(<https://github.com/httpwg/http-extensions/issues/72>).

Elevation of privilege
(<https://github.com/httpwg/http-extensions/issues/73>).

Alternates of alternates
(<https://github.com/httpwg/http-extensions/issues/74>).

Alt-Svc and Cert Pinning
(<https://github.com/httpwg/http-extensions/issues/76>).

Using alt-svc on localhost (no change to spec, see
<https://github.com/httpwg/http-extensions/issues/89>).

IANA procedure for alt-svc parameters
(<https://github.com/httpwg/http-extensions/issues/96>).

Alt-svc from https (1.1) to https (1.1)
(<https://github.com/httpwg/http-extensions/issues/91>).

Alt-svc vs the ability to convey the scheme inside the protocol
(<https://github.com/httpwg/http-extensions/issues/92>).

Reconciling MAY/can vs. SHOULD
(<https://github.com/httpwg/http-extensions/issues/101>).

Typo in alt-svc caching example
(<https://github.com/httpwg/http-extensions/issues/117>).

A.11. Since draft-ietf-httpbis-alt-svc-09

Editorial improvements
(<https://github.com/httpwg/http-extensions/issues/118>,
<https://github.com/httpwg/http-extensions/issues/119>,
<https://github.com/httpwg/http-extensions/issues/120>,
<https://github.com/httpwg/http-extensions/issues/121>,
<https://github.com/httpwg/http-extensions/issues/122>,
<https://github.com/httpwg/http-extensions/issues/123>,
<https://github.com/httpwg/http-extensions/issues/125>,
<https://github.com/httpwg/http-extensions/issues/126>).

A.12. Since draft-ietf-httpbis-alt-svc-10

Editorial improvements
(<https://github.com/httpwg/http-extensions/issues/130>).

Use RFC 7405 ABNF extension
(<https://github.com/httpwg/http-extensions/issues/131>).

A.13. Since draft-ietf-httpbis-alt-svc-11

Security considerations wrt system ports
(<https://github.com/httpwg/http-extensions/issues/139>).

A.14. Since draft-ietf-httpbis-alt-svc-12

Editorial changes triggered by <https://lists.w3.org/Archives/Public/ietf-http-wg/2016JanMar/0243.html>.

Reasonable Assurances and H2C
(<https://github.com/httpwg/http-extensions/issues/148>).

Appendix B. 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/

HTTP Working Group
Internet-Draft
Intended status: Experimental
Expires: September 18, 2017

M. Nottingham

M. Thomson
Mozilla

March 17, 2017

Opportunistic Security for HTTP/2
draft-ietf-httpbis-http2-encryption-11

Abstract

This document describes how "http" URIs can be accessed using Transport Layer Security (TLS) and HTTP/2 to mitigate pervasive monitoring attacks. This mechanism not a replacement for "https" URIs; it is vulnerable to active attacks.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at http://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on September 18, 2017.

Table of Contents

1. Introduction

This document describes a use of HTTP Alternative Services [RFC7838] to decouple the URI scheme from the use and configuration of underlying encryption. It allows an "http" URI to be accessed using HTTP/2 [RFC7230] and Transport Layer Security (TLS) [RFC5246] with Opportunistic Security [RFC7435].

This document describes a usage model whereby sites can serve "http" URIs over TLS, thereby avoiding the problem of serving Mixed Content (described in [W3C.CR-mixed-content-20160802]) while still providing protection against passive attacks.

Opportunistic Security does not provide the same guarantees as using TLS with "https" URIs, because it is vulnerable to active attacks, and does not change the security context of the connection. Normally, users will not be able to tell that it is in use (i.e., there will be no "lock icon").

1.1. Goals and Non-Goals

The immediate goal is to make the use of HTTP more robust in the face of pervasive passive monitoring [RFC7258].

A secondary (but significant) goal is to provide for ease of implementation, deployment and operation. This mechanism is expected

to have a minimal impact upon performance, and require a trivial administrative effort to configure.

Preventing active attacks (such as a Man-in-the-Middle) is a non-goal for this specification. Furthermore, this specification is not intended to replace or offer an alternative to "https", since "https" both prevents active attacks and invokes a more stringent security model in most clients.

1.2. 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].

2. Using HTTP URIs over TLS

An origin server that supports the resolution of "http" URIs can indicate support for this specification by providing an alternative service advertisement [RFC7838] for a protocol identifier that uses TLS, such as "h2" [RFC7540]. Such a protocol MUST include an explicit indication of the scheme of the resource. This excludes HTTP/1.1; HTTP/1.1 clients are forbidden from including the absolute form of a URI in requests to origin servers (see Section 5.3.1 of [RFC7230]).

A client that receives such an advertisement MAY make future requests intended for the associated origin [RFC6454] to the identified service (as specified by [RFC7838]), provided that the alternative service opts in as described in Section 2.1.

A client that places the importance of protection against passive attacks over performance might choose to withhold requests until an encrypted connection is available. However, if such a connection cannot be successfully established, the client can resume its use of the cleartext connection.

A client can also explicitly probe for an alternative service advertisement by sending a request that bears little or no sensitive information, such as one with the OPTIONS method. Likewise, clients with existing alternative services information could make such a request before they expire, in order minimize the delays that might be incurred.

Client certificates are not meaningful for URLs with the "http" scheme, and therefore clients creating new TLS connections to alternative services for the purposes of this specification MUST NOT present them. A server that also provides "https" resources on the

same port can request a certificate during the TLS handshake, but it MUST NOT abort the handshake if the client does not provide one.

2.1. Alternative Server Opt-In

It is possible that the server might become confused about whether requests' URLs have a "http" or "https" scheme, for various reasons; see Section 4.4. To ensure that the alternative service has opted into serving "http" URLs over TLS, clients are required to perform additional checks before directing "http" requests to it.

Clients MUST NOT send "http" requests over a secured connection, unless the chosen alternative service presents a certificate that is valid for the origin as defined in [RFC2818]. Using an authenticated alternative service establishes "reasonable assurances" for the purposes of [RFC7838]. In addition to authenticating the server, the client MUST have obtained a valid http-opportunistic response for an origin (as per Section 2.3) using the authenticated connection. An exception to the latter restriction is made for requests for the "http-opportunistic" well-known URI.

For example, assuming the following request is made over a TLS connection that is successfully authenticated for those origins, the following request/response pair would allow requests for the origins "http://www.example.com" or "http://example.com" to be sent using a secured connection:

```
HEADERS
  + END_STREAM
  + END_HEADERS
    :method = GET
    :scheme = http
    :authority = example.com
    :path = /.well-known/http-opportunistic

HEADERS
    :status = 200
    content-type = application/json
DATA
  + END_STREAM
[ "http://www.example.com", "http://example.com" ]
```

Though this document describes multiple origins, this is only for operational convenience. Only a request made to an origin (over an authenticated connection) can be used to acquire this resource for that origin. Thus in the example, the request to "http://example.com" cannot be assumed to also provide an http-opportunistic response for "http://www.example.com".

2.2. Interaction with "https" URIs

Clients MUST NOT send "http" requests and "https" requests on the same connection. Similarly, clients MUST NOT send "http" requests for multiple origins on the same connection.

2.3. The "http-opportunistic" well-known URI

This specification defines the "http-opportunistic" well-known URI [RFC5785]. A client is said to have a valid http-opportunistic response for a given origin when:

- The client has requested the well-known URI from the origin over an authenticated connection and a 200 (OK) response was provided, and
- That response is fresh [RFC7234] (potentially through revalidation [RFC7232]), and
- That response has the media type "application/json", and
- That response's payload, when parsed as JSON [RFC7159], contains an array as the root, and
- The array contains a string that is a case-insensitive character-for-character match for the origin in question, serialised into Unicode as per Section 6.1 of [RFC6454].

A client MAY treat an "http-opportunistic" resource as invalid if values it contains are not strings.

This document does not define semantics for "http-opportunistic" resources on an "https" origin, nor does it define semantics if the resource includes "https" origins.

Allowing clients to cache the http-opportunistic resource means that all alternative services need to be able to respond to requests for "http" resources. A client is permitted to use an alternative service without acquiring the http-opportunistic resource from that service.

A client MUST NOT use any cached copies of an http-opportunistic resource that was acquired (or revalidated) over an unauthenticated connection. To avoid potential errors, a client can request or revalidate the http-opportunistic resource before using any connection to an alternative service.

Clients that use cached http-opportunistic responses MUST ensure that their cache is cleared of any responses that were acquired over an unauthenticated connection. Revalidating an unauthenticated response using an authenticated connection does not ensure the integrity of the response.

3. IANA Considerations

This specification registers a Well-Known URI [RFC5785]:

- URI Suffix: http-opportunistic
- Change Controller: IETF
- Specification Document(s): Section 2.3 of [this specification]
- Related Information:

4. Security Considerations

4.1. Security Indicators

User Agents MUST NOT provide any special security indicators when an "http" resource is acquired using TLS. In particular, indicators that might suggest the same level of security as "https" MUST NOT be used (e.g., a "lock device").

4.2. Downgrade Attacks

A downgrade attack against the negotiation for TLS is possible.

For example, because the "Alt-Svc" header field [RFC7838] likely appears in an unauthenticated and unencrypted channel, it is subject to downgrade by network attackers. In its simplest form, an attacker that wants the connection to remain in the clear need only strip the "Alt-Svc" header field from responses.

4.3. Privacy Considerations

Cached alternative services can be used to track clients over time; e.g., using a user-specific hostname. Clearing the cache reduces the ability of servers to track clients; therefore clients MUST clear cached alternative service information when clearing other origin-based state (i.e., cookies).

4.4. Confusion Regarding Request Scheme

HTTP implementations and applications sometimes use ambient signals to determine if a request is for an "https" resource; for example, they might look for TLS on the stack, or a server port number of 443.

This might be due to expected limitations in the protocol (the most common HTTP/1.1 request form does not carry an explicit indication of the URI scheme and the resource might have been developed assuming HTTP/1.1), or it may be because how the server and application are implemented (often, they are two separate entities, with a variety of possible interfaces between them).

Any security decisions based upon this information could be misled by the deployment of this specification, because it violates the assumption that the use of TLS (or port 443) means that the client is accessing a HTTPS URI, and operating in the security context implied by HTTPS.

Therefore, server implementers and administrators need to carefully examine the use of such signals before deploying this specification.

4.5. Server Controls

This specification requires that a server send both an Alternative Service advertisement and host content in a well-known location to send HTTP requests over TLS. Servers SHOULD take suitable measures to ensure that the content of the well-known resource remains under their control. Likewise, because the Alt-Svc header field is used to describe policies across an entire origin, servers SHOULD NOT permit user content to set or modify the value of this header.

5. References

5.1. 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/rfc2119>.

[RFC2818] Rescorla, E., "HTTP Over TLS", RFC 2818, DOI 10.17487/RFC2818, May 2000, <http://www.rfc-editor.org/info/rfc2818>.

[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/rfc5246>.

[RFC5785] Nottingham, M. and E. Hammer-Lahav, "Defining Well-Known Uniform Resource Identifiers (URIs)", RFC 5785, DOI 10.17487/RFC5785, April 2010, <http://www.rfc-editor.org/info/rfc5785>.

[RFC6454] Barth, A., "The Web Origin Concept", RFC 6454, DOI 10.17487/RFC6454, December 2011, <http://www.rfc-editor.org/info/rfc6454>.

[RFC7159] Bray, T., Ed., "The JavaScript Object Notation (JSON) Data Interchange Format", RFC 7159, DOI 10.17487/RFC7159, March 2014, <http://www.rfc-editor.org/info/rfc7159>.

[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/rfc7230>.

[RFC7232] Fielding, R., Ed. and J. Reschke, Ed., "Hypertext Transfer Protocol (HTTP/1.1): Conditional Requests", RFC 7232, DOI 10.17487/RFC7232, June 2014, <http://www.rfc-editor.org/info/rfc7232>.

[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/rfc7234>.

[RFC7540] Belshe, M., Peon, R., and M. Thomson, Ed., "Hypertext Transfer Protocol Version 2 (HTTP/2)", RFC 7540, DOI 10.17487/RFC7540, May 2015, <http://www.rfc-editor.org/info/rfc7540>.

[RFC7838] Nottingham, M., McManus, P., and J. Reschke, "HTTP Alternative Services", RFC 7838, DOI 10.17487/RFC7838, April 2016, <http://www.rfc-editor.org/info/rfc7838>.

5.2. Informative References

[RFC7258] Farrell, S. and H. Tschofenig, "Pervasive Monitoring Is an Attack", BCP 188, RFC 7258, DOI 10.17487/RFC7258, May 2014, <http://www.rfc-editor.org/info/rfc7258>.

[RFC7435] Dukhovni, V., "Opportunistic Security: Some Protection Most of the Time", RFC 7435, DOI 10.17487/RFC7435, December 2014, <http://www.rfc-editor.org/info/rfc7435>.

[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/rfc7469>.

[W3C.CR-mixed-content-20160802] West, M., "Mixed Content", World Wide Web Consortium CR CR-mixed-content-20160802, August 2016, <https://www.w3.org/TR/2016/CR-mixed-content-20160802>.

Appendix A. Acknowledgements

Mike Bishop contributed significant text to this document.

Thanks to Patrick McManus, Stefan Eissing, Eliot Lear, Stephen Farrell, Guy Podjarny, Stephen Ludin, Erik Nygren, Paul Hoffman, Adam Langley, Eric Rescorla, Julian Reschke, Kari Hurtta, and Richard Barnes for their feedback and suggestions.

Authors' Addresses

Mark Nottingham

Email: mnot@mnot.net
URI: https://www.mnot.net/

Martin Thomson
Mozilla

Email: martin.thomson@gmail.com

HTTP Working Group
Internet-Draft
Intended status: Standards Track
Expires: May 13, 2016

T. Bray
Textuality

November 10, 2015

An HTTP Status Code to Report Legal Obstacles
draft-ietf-httpbis-legally-restricted-status-04

Abstract

This document specifies a Hypertext Transfer Protocol (HTTP) status code for use when resource access is denied as a consequence of legal demands.

Editorial Note (To be removed by RFC Editor before publication)

Discussion of this draft takes place on the HTTPBIS working group mailing list (ietf-http-wg@w3.org), which is archived at [1].

Working Group information can be found at [2] and [3]; source code and issues list for this draft can be found at [4].

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at http://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on May 13, 2016.

Copyright Notice

Table of Contents

1. Introduction

This document specifies a Hypertext Transfer Protocol (HTTP) status code for use when a server operator has received a legal demand to deny access to a resource or to a set of resources which includes the requested resource.

This status code can be used to provide transparency in circumstances where issues of law or public policy affect server operations. This transparency may be beneficial both to these operators and to end users.

[RFC4924] discusses the forces working against transparent operation of the Internet; these clearly include legal interventions to restrict access to content. As that document notes, and as Section 4 of [RFC4084] states, such restrictions should be made explicit.

Feedback should occur on the ietf-http-wg@w3.org mailing list.

2. Requirements

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].

3. 451 Unavailable For Legal Reasons

This status code indicates that the server is denying access to the resource as a consequence of a legal demand.

The server in question might not be an origin server. This type of legal demand typically most directly affects the operations of ISPs and search engines.

Responses using this status code SHOULD include an explanation, in the response body, of the details of the legal demand: the party making it, the applicable legislation or regulation, and what classes of person and resource it applies to. For example:

```
HTTP/1.1 451 Unavailable For Legal Reasons
Link: <https://spqr.example.org/legislatione>; rel="blocked-by"
Content-Type: text/html

<html>
 <head><title>Unavailable For Legal Reasons</title></head>
 <body>
  <h1>Unavailable For Legal Reasons</h1>
  <p>This request may not be serviced in the Roman Province
  of Judea due to the Lex Julia Majestatis, which disallows
  access to resources hosted on servers deemed to be
  operated by the People’s Front of Judea.</p>
 </body>
</html>
```

The use of the 451 status code implies neither the existence nor non-existence of the resource named in the request. That is to say, it is possible that if the legal demands were removed, a request for the resource still might not succeed.

Note that in many cases clients can still access the denied resource by using technical countermeasures such as a VPN or the Tor network.

A 451 response is cacheable by default; i.e., unless otherwise indicated by the method definition or explicit cache controls; see [RFC7234].

4. Identifying Blocking Entities

As noted above, when an attempt to access a resource fails with status 451, the entity blocking access might or might not be the origin server. There are a variety of entities in the resource-

access path which could choose to deny access, for example ISPs, cache providers, and DNS servers.

It is useful, when legal blockages occur, to be able to identify the entities actually implementing the blocking.

When an entity blocks access to a resource and returns status 451, it SHOULD include a "Link" HTTP header field [RFC5988] whose value is a URI reference [RFC3986] identifying itself. When used for this purpose, the "Link" header field MUST have a "rel" parameter whose value is "blocked-by".

The intent is that the header be used to identify the entity actually implementing blockage, not any other entity mandating it. A human readable response body, as discussed above, is the appropriate location for discussion of administrative and policy issues.

5. Security Considerations

Clients cannot rely upon the use of the 451 status code. It is possible that certain legal authorities might wish to avoid transparency, and not only demand the restriction of access to certain resources, but also avoid disclosing that the demand was made.

6. IANA Considerations

The HTTP Status Codes Registry should be updated with the following entry:

- Code: 451
- Description: Unavailable for Legal Reasons
- Specification: [this document]

The Link Relation Type Registry should updated with the following entry:

- Relation Name: blocked-by
- Description: Identifies the entity blocking access to a resource folllowing on receipt of a legal demand.
- Reference: This document

7. References

7.1. 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/rfc2119>.

[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/rfc3986>.

[RFC5988] Nottingham, M., "Web Linking", RFC 5988, DOI 10.17487/RFC5988, October 2010, <http://www.rfc-editor.org/info/rfc5988>.

[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/rfc7234>.

7.2. Informative References

[RFC4084] Klensin, J., "Terminology for Describing Internet Connectivity", BCP 104, RFC 4084, DOI 10.17487/RFC4084, May 2005, <http://www.rfc-editor.org/info/rfc4084>.

[RFC4924] Aboba, B., Ed. and E. Davies, "Reflections on Internet Transparency", RFC 4924, DOI 10.17487/RFC4924, July 2007, <http://www.rfc-editor.org/info/rfc4924>.

Appendix A. Acknowledgements

Thanks to Terence Eden, who observed that the existing status code 403 was not really suitable for this situation, and suggested the creation of a new status code.

Thanks also to Ray Bradbury.

Author’s Address

Tim Bray
Textuality

Email: tbray@textuality.com
URI: http://www.tbray.org/ongoing/

Individual Submission
Internet-Draft
Intended status: Standards Track
Expires: March 23, 2019

J. Reschke

A. Malhotra

J. Snell
September 19, 2018

HTTP SEARCH Method
draft-snell-search-method-01

Abstract

This specification updates the definition and semantics of the HTTP SEARCH request method originally defined by [RFC5323].

Table of Contents

1. Introduction

This specification updates the HTTP SEARCH method originally defined in [RFC5323].

Many existing HTTP-based applications use the HTTP GET and POST methods in various ways to implement the functionality provided by SEARCH.

Using a GET request with some combination of query parameters included within the request URI (as illustrated in the example below) is arguably the most common mechanism for implementing search in web applications. With this approach, implementations are required to parse the request URI into distinct path (everything before the '?') and query elements (everything after the '?'). The path identifies the resource processing the query (in this case 'http://example.org/feed') while the query identifies the specific parameters of the search operation.

A typical use of HTTP GET for requesting a search

```
GET /feed?q=foo&limit=10&sort=-published HTTP/1.1
Host: example.org
```

While there are definite advantages to using GET requests in this manner, the disadvantages should not be overlooked. Specifically:

- Without specific knowledge of the resource and server to which the GET request is being sent, there is no way for the client to know that a search operation is being requested. Identical requests sent to two different servers can implement entirely different semantics.

- Encoding query parameters directly into the request URI effectively casts every possible combination of query inputs as

distinct resources. For instance, because mechanisms such as HTTP caching handle request URIs as opaque character sequences, queries such as 'http://example.org/?q=foo' and 'http://example.org/?q=Foo' will be treated as entirely separate resources even if they yield identical results.

- While most modern browser and server implementations allow for long request URIs, there is no standardized minimum or maximum length for URIs in general. Many resource constrained devices enforce strict limits on the maximum number of characters that can be included in a URI. Such limits can prove impractical for large or complex query parameters.

- Query expressions included within a request URI must either be restricted to relatively simple key value pairs or encoded such that the query can be safely represented in the limited character-set allowed by URL standards. Such encoding can add significant complexity, introduce bugs, or otherwise reduce the overall visibility of the query being requested.

As an alternative to using GET, many implementations make use of the HTTP POST method to perform queries, as illustrated in the example below. In this case, the input parameters to the search operation are passed along within the request payload as opposed to using the request URI.

A typical use of HTTP GET for requesting a search

```
POST /feed HTTP/1.1
Host: example.org
Content-Type: application/x-www-form-urlencoded

q=foo&limit=10&sort=-published
```

This variation, however, suffers from the same basic limitation as GET in that it is not readily apparent -- absent specific knowledge of the resource and server to which the request is being sent -- that a search operation is what is being requested. Web applications use the POST method for a wide variety of uses including the creation or modification of existing resources. Sending the request above to a different server, or even repeatedly sending the request to the same server could have dramatically different effects.

The SEARCH method provides a solution that spans the gap between the use of GET and POST. As with POST, the input to the query operation is passed along within the payload of the request rather than as part of the request URI. Unlike POST, however the semantics of the SEARCH method are specifically defined.

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

2. SEARCH

The SEARCH method is used to initiate a server-side search. Unlike the HTTP GET method, which requests that a server return a representation of the resource identified by the effective request URI (as defined by [RFC7230]), the SEARCH method is used to ask the server to perform a query operation (described by the request payload) over some set of data scoped to the effective request URI. The payload returned in response to a SEARCH cannot be assumed to be a representation of the resource identified by the effective request URI.

The body payload of the request defines the query. Implementations MAY use a request body of any content type with the SEARCH method; however, for backwards compatibility with existing WebDAV implementations, SEARCH requests that use the text/xml or application/xml content types MUST be processed per the requirements established by [RFC5323].

SEARCH requests are both safe and idempotent with regards to the resource identified by the request URI. That is, SEARCH requests do not alter the state of the targeted resource. However, while processing a search request, a server can be expected to allocate computing and memory resources or even create additional HTTP resources through which the response can be retrieved.

A successful response to a SEARCH request is expected to provide some indication as to the final disposition of the search operation. For instance, a successful search that yields no results can be represented by a 204 No Content response. If the response includes a body payload, the payload is expected to describe the results of the search operation. In some cases, the server may choose to respond indirectly to the SEARCH request by returning a 3xx Redirection with a Location header specifying an alternate Request URI from which the search results can be retrieved using an HTTP GET request. Various non-normative examples of successful SEARCH responses are illustrated in Section 4.

The response to a SEARCH request is not cacheable. It ought to be noted, however, that because SEARCH requests are safe and idempotent, responses to a SEARCH MUST NOT invalidate previously cached responses to other requests directed at the same effective request URI.

The semantics of the SEARCH method change to a "conditional SEARCH" if the request message includes an If-Modified-Since, If-Unmodified-Since, If-Match, If-None-Match, or If-Range header field ([RFC7232]). A conditional SEARCH requests that the query be performed only under the circumstances described by the conditional header field(s). It is important to note, however, that such conditions are evaluated against the state of the target resource itself as opposed to the collected results of the search operation.

3. The "Accept-Search" Header Field

The "Accept-Search" response header field MAY be used by a server to directly signal support for the SEARCH method while identifying the specific query format Content-Type's that may be used.

Accept-Search = "Accept-Search" ":" 1#media-type

The Accept-Search header specifies a comma-separated listing of media types (with optional parameters) as defined by [RFC7231], Section 3.1.1.1.

The order of types listed by the Accept-Search header is insignificant.

4. Examples

The non-normative examples in this section make use of a simple, hypothetical plain-text based query syntax based on SQL with results returned as comma-separated values. This is done for illustration purposes only. Implementations are free to use any format they wish on both the request and response.

4.1. Simple SEARCH with a Direct Response

A simple query with a direct response:

```
SEARCH /contacts HTTP/1.1
Host: example.org
Content-Type: text/query
Accept: text/csv

select surname, givenname, email limit 10
```

Response:

```
HTTP/1.1 200 OK
Content-Type: text/csv

surname, givenname, email
Smith, John, john.smith@example.org
Jones, Sally, sally.jones@example.com
Dubois, Camille, camille.dubois@example.net
```

4.2. Simple SEARCH with indirect response (303 See Other)

A simple query with an Indirect Response (303 See Other)

```
SEARCH /contacts HTTP/1.1
Host: example.org
Content-Type: text/query
Accept: text/csv

select surname, givenname, email limit 10
```

Response:

```
HTTP/1.1 303 See Other
Location: http://example.org/contacts/query123
```

Fetch Query Response:

```
GET /contacts/query123 HTTP/1.1
Host: example.org
```

Response:

```
HTTP/1.1 200 OK
Content-Type: text/csv

surname, givenname, email
Smith, John, john.smith@example.org
Jones, Sally, sally.jones@example.com
Dubois, Camille, camille.dubois@example.net
```

5. Security Considerations

The SEARCH method is subject to the same general security considerations as all HTTP methods as described in [RFC7231].

6. IANA Considerations

IANA is requested to update the registration of the SEARCH method in the permanent registry at <http://www.iana.org/assignments/http-methods> (see Section 8.1 of [RFC7231]).

Method Name	Safe	Idempotent	Specification
SEARCH	Yes	Yes	Section 2

7. 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, <https://www.rfc-editor.org/info/rfc2119>.

[RFC4918] Dusseault, L., Ed., "HTTP Extensions for Web Distributed Authoring and Versioning (WebDAV)", RFC 4918, DOI 10.17487/RFC4918, June 2007, <https://www.rfc-editor.org/info/rfc4918>.

[RFC5323] Reschke, J., Ed., Reddy, S., Davis, J., and A. Babich, "Web Distributed Authoring and Versioning (WebDAV) SEARCH", RFC 5323, DOI 10.17487/RFC5323, November 2008, <https://www.rfc-editor.org/info/rfc5323>.

[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, <https://www.rfc-editor.org/info/rfc7230>.

[RFC7231] Fielding, R., Ed. and J. Reschke, Ed., "Hypertext Transfer Protocol (HTTP/1.1): Semantics and Content", RFC 7231, DOI 10.17487/RFC7231, June 2014, <https://www.rfc-editor.org/info/rfc7231>.

[RFC7232] Fielding, R., Ed. and J. Reschke, Ed., "Hypertext Transfer Protocol (HTTP/1.1): Conditional Requests", RFC 7232, DOI 10.17487/RFC7232, June 2014, <https://www.rfc-editor.org/info/rfc7232>.

[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, <https://www.rfc-editor.org/info/rfc7234>.

Authors' Addresses

Julian Reschke

Email: julian.reschke@greenbytes.de

Ashok Malhotra

Email: ashok.malhotra@oracle.com

James M Snell

Email: jasnell@gmail.com