Internet Engineering Task Force (IETF)
Request for Comments: 7239
Category: Standards Track
ISSN: 2070-1721

A. Petersson
M. Nilsson
Opera Software

June 2014

# Forwarded HTTP Extension

## Abstract

This document defines an HTTP extension header field that allows proxy components to disclose information lost in the proxying process, for example, the originating IP address of a request or IP address of the proxy on the user-agent-facing interface. In a path of proxying components, this makes it possible to arrange it so that each subsequent component will have access to, for example, all IP addresses used in the chain of proxied HTTP requests.

This document also specifies guidelines for a proxy administrator to anonymize the origin of a request.

## 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.

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/rfc7239.

Table of Contents

## 1. Introduction

In today's HTTP landscape, there are a multitude of different applications that act as proxies for the user agents. In many cases, these proxies exists without the action or knowledge of the end-user. These cases occur, for example, when the proxy exists as a part of the infrastructure within the organization running the web server. Such proxies may be used for features such as load balancing or crypto offload. Another example is when the proxy is used within the same organization as the user, and the proxy is used to cache resources. However, these proxies make the requests appear as if they originated from the proxy's IP address, and they may change other information in the original request. This represents a loss of information from the original request.

This loss of information can cause problems for a web server that has a specific use for the clients' IP addresses that will not be met by using the address of the proxy or other information changed by the proxy. The main uses of this information are for diagnostics, access control, and abuse management. Diagnostic functions can include event logging, troubleshooting, and statistics gathering, and the information collected is usually only stored for short periods of time and only gathered in response to a particular problem or a complaint from the client. Access control can be operated by configuring a list of client IP addresses from which access is permitted, but this approach will not work if a proxy is used, unless the proxy is trusted and is, itself, configured with a list of allowed client addresses for the server. Cases of abuse require identification of the abuser and this uses many of the same features identified for diagnostics.

Most of the time that a proxy is used, this loss of information is not the primary purpose, or even a desired effect, of using the proxy. Thus, to restore the desired functionality when a proxy is in use, a way of disclosing the original information at the HTTP level is needed. Clearly, however, when the purpose of using a proxy is to provide client anonymity, the proxy will not use the feature defined in this document.

It should be noted that the use of a reverse proxy also hides information. Again, where the loss of information is not a deliberate function of the use of the reverse proxy, it can be desirable to find a way to encode the information within the HTTP messages so that the consumer can see it.

A common way to disclose this information is by using the non-standard header fields such as X-Forwarded-For, X-Forwarded-By, and X-Forwarded-Proto. There are many benefits to using a standardized

approach to commonly desired protocol function: not least is interoperability between implementations. This document standardizes a header field called "Forwarded" and provides the syntax and semantics for disclosing such information. "Forwarded" also combines all the information within one single header field, making it possible to correlate that information. With the header field format described in this document, it is possible to know what information belongs together, as long as the proxies are trusted. Such conclusions are not possible to make with the X-Forwarded class of header fields. The header field defined in this document is optional such that implementations of proxies that are intended to provide privacy are not required to operate or implement the header field.

Note that similar issues to those described for proxies also arise with use of NATs. This is discussed further in [RFC6269].

## 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].

## 3. Syntax Notations

This specification uses the Augmented Backus-Naur Form (ABNF) notation of [RFC5234] with the list rule extension defined in Section 7 of [RFC7230].

## 4. Forwarded HTTP Header Field

The "Forwarded" HTTP header field is an OPTIONAL header field that, when used, contains a list of parameter-identifier pairs that disclose information that is altered or lost when a proxy is involved in the path of the request. Due to the sensitive nature of the data passed in this header field (see Sections 8.2 and 8.3), this header field should be turned off by default. Further, each parameter should be configured individually. "Forwarded" is only for use in HTTP requests and is not to be used in HTTP responses. This applies to forwarding proxies, as well as reverse proxies. Information passed in this header field can be, for example, the source IP address of the request, the IP address of the incoming interface on the proxy, or whether HTTP or HTTPS was used. If the request is passing through several proxies, each proxy can add a set of parameters; it can also remove previously added "Forwarded" header fields.

The top-level list is represented as a list of HTTP header field-values as defined in Section 3.2 of [RFC7230]. The first element in this list holds information added by the first proxy that implements and uses this header field, and each subsequent element holds information added by each subsequent proxy. Because this header field is optional, any proxy in the chain may choose not to update this header field. Each field-value is a semicolon-separated list; this sublist consists of parameter-identifier pairs. Parameter-identifier pairs are grouped together by an equals sign. Each parameter MUST NOT occur more than once per field-value. The parameter names are case-insensitive. The header field value can be defined in ABNF syntax as:

```
Forwarded   = 1#forwarded-element

forwarded-element =
    [ forwarded-pair ] *( ";" [ forwarded-pair ] )

forwarded-pair = token "=" value
value          = token / quoted-string

token = <Defined in [RFC7230], Section 3.2.6>
quoted-string = <Defined in [RFC7230], Section 3.2.6>
```

Examples:

```
Forwarded: for=" gazonk"
Forwarded: For="[2001:db8:cafe::17]:4711"
Forwarded: for=192.0.2.60;proto=http;by=203.0.113.43
Forwarded: for=192.0.2.43, for=198.51.100.17
```

Note that as ":" and "[]" are not valid characters in "token", IPv6 addresses are written as "quoted-string".

A proxy server that wants to add a new "Forwarded" header field value can either append it to the last existing "Forwarded" header field after a comma separator or add a new field at the end of the header block. A proxy MAY remove all "Forwarded" header fields from a request. It MUST, however, ensure that the correct header field is updated in case of multiple "Forwarded" header fields.

## 5. Parameters

This document specifies a number of parameters and valid values for each of them:

- "by" identifies the user-agent facing interface of the proxy.
- "for" identifies the node making the request to the proxy.
- "host" is the host request header field as received by the proxy.
- "proto" indicates what protocol was used to make the request.

### 5.1. Forwarded By

The "by" parameter is used to disclose the interface where the request came in to the proxy server. When proxies choose to use the "by" parameter, its default configuration SHOULD contain an obfuscated identifier as described in Section 6.3. If the server receiving proxied requests requires some address-based functionality, this parameter MAY instead contain an IP address (and, potentially, a port number). A third option is the "unknown" identifier described in Section 6.2.

The syntax of a "by" value, after potential quoted-string unescaping, conforms to the "node" ABNF described in Section 6.

This is primarily added by reverse proxies that wish to forward this information to the backend server. It can also be interesting in a multihomed environment to signal to backend servers from which the request came.

### 5.2. Forwarded For

The "for" parameter is used to disclose information about the client that initiated the request and subsequent proxies in a chain of proxies. When proxies choose to use the "for" parameter, its default configuration SHOULD contain an obfuscated identifier as described in Section 6.3. If the server receiving proxied requests requires some address-based functionality, this parameter MAY instead contain an IP address (and, potentially, a port number). A third option is the "unknown" identifier described in Section 6.2.

The syntax of a "for" value, after potential quoted-string unescaping, conforms to the "node" ABNF described in Section 6.

In a chain of proxy servers where this is fully utilized, the first "for" parameter will disclose the client where the request was first made, followed by any subsequent proxy identifiers. The last proxy in the chain is not part of the list of "for" parameters. The last proxy's IP address, and optionally a port number, are, however, readily available as the remote IP address at the transport layer. It can, however, be more relevant to read information about the last proxy from preceding "Forwarded" header field's "by" parameter, if present.

### 5.3. Forwarded Host

The "host" parameter is used to forward the original value of the "Host" header field. This can be used, for example, by the origin server if a reverse proxy is rewriting the "Host" header field to some internal host name.

The syntax for a "host" value, after potential quoted-string unescaping, MUST conform to the Host ABNF described in Section 5.4 of [RFC7230].

### 5.4. Forwarded Proto

The "proto" parameter has the value of the used protocol type. The syntax of a "proto" value, after potential quoted-string unescaping, MUST conform to the URI scheme name as defined in Section 3.1 in [RFC3986] and registered with IANA according to [RFC4395]. Typical values are "http" or "https".

For example, in an environment where a reverse proxy is also used as a crypto offloader, this allows the origin server to rewrite URLs in a document to match the type of connection as the user agent requested, even though all connections to the origin server are unencrypted HTTP.

### 5.5. Extensions

Extensions allow for additional parameters and values. Extensions can be particularly useful in reverse proxy environments. All extension parameters SHOULD be registered in the "HTTP Forwarded Parameter" registry. If certain extensions are expected to have widespread deployment, they SHOULD also be standardized. This is further discussed in Section 9.

## 6. Node Identifiers

The node identifier is one of the following:

- The client's IP address, with an optional port number
- A token indicating that the IP address of the client is not known to the proxy server
- A generated token, allowing for tracing and debugging, while allowing the internal structure or sensitive information to be hidden

The node identifier is defined by the ABNF syntax as:

```
node     = nodename [ ":" node-port ]
nodename = IPv4address / "[" IPv6address "]" /
           "unknown" / obfnode

IPv4address = <Defined in [RFC3986], Section 3.2.2>
IPv6address = <Defined in [RFC3986], Section 3.2.2>
obfnode = "_" 1*( ALPHA / DIGIT / "." / "_" / "-")

node-port     = port / obfport
port          = 1*5DIGIT
obfport       = "_" 1*(ALPHA / DIGIT / "." / "_" / "-")

DIGIT = <Defined in [RFC5234], Section 3.4>
ALPHA = <Defined in [RFC5234], Section B.1>
```

Each of the identifiers may optionally have the port identifier, for example, allowing the identification of the endpoint in a NATed environment. The "node-port" can be identified either by its port number or by a generated token obfuscating the real port number. An obfuscated port may be used in situations where the possessor of the proxy wants the ability to trace requests -- for example, in debug purposes -- but does not want to reveal internal information.

Note that the ABNF above also allows port numbers to be appended to the "unknown" identifier. Interpretation of such notation is, however, left to the possessor of a proxy adding such a value to the header field. To distinguish an "obfport" from a port, the "obfport" MUST have a leading underscore. Further, it MUST also consist of only "ALPHA", "DIGIT", and the characters ".", "_", and "-".

It is important to note that an IPv6 address and any nodename with node-port specified MUST be quoted, since ":" is not an allowed character in "token".

Examples:

```
"192.0.2.43:47011"
"[2001:db8:cafe::17]:47011"
```

### 6.1. IPv4 and IPv6 Identifiers

The ABNF rules for "IPv6address" and "IPv4address" are defined in [RFC3986]. The "IPv6address" SHOULD comply with textual representation recommendations [RFC5952] (for example, lowercase, compression of zeros).

Note that the IP address may be one from the internal nets, as defined in [RFC1918] and [RFC4193]. Also, note that an IPv6 address is always enclosed in square brackets.

### 6.2. The "unknown" Identifier

The "unknown" identifier is used when the identity of the preceding entity is not known, but the proxy server still wants to signal that a forwarding of the request was made. One example would be a proxy server process generating an outgoing request without direct access to the incoming request TCP socket.

### 6.3. Obfuscated Identifier

A generated identifier may be used where there is a wish to keep the internal IP addresses secret, while still allowing the "Forwarded" header field to be used for tracing and debugging. This can also be useful if the proxy uses some sort of interface labels and there is a desire to pass them rather than an IP address. Unless static assignment of identifiers is necessary for the server's use of the identifiers, obfuscated identifiers SHOULD be randomly generated for each request. If the server requires that identifiers persist across requests, they SHOULD NOT persist longer than client IP addresses. To distinguish the obfuscated identifier from other identifiers, it MUST have a leading underscore "_". Furthermore, it MUST also consist of only "ALPHA", "DIGIT", and the characters ".", "_", and "-".

Example:

```
Forwarded: for=_hidden, for=_SEVKISEK
```

## 7. Implementation Considerations

### 7.1. HTTP Lists

Note that an HTTP list allows white spaces to occur between the identifiers, and the list may be split over multiple header fields. As an example, the header field

```
Forwarded: for=192.0.2.43,for="[2001:db8:cafe::17]",for=unknown
```

is equivalent to the header field

```
Forwarded: for=192.0.2.43, for="[2001:db8:cafe::17]", for=unknown
```

which is equivalent to the header fields

```
Forwarded: for=192.0.2.43
Forwarded: for="[2001:db8:cafe::17]", for=unknown
```

### 7.2. Header Field Preservation

There are some cases when this header field should be kept and some cases where it should not be kept. A directly forwarded request should preserve and possibly extend it. If a single incoming request causes the proxy to make multiple outbound requests, special care must be taken to decide whether or not the header field should be preserved. In many cases, the header field should be preserved, but if the outbound request is not a direct consequence of the incoming request, the header field should not be preserved. Consider also the case when a proxy has detected a content mismatch in a 304 response and is following the instructions in [RFC7232], Section 4.1 to repeat the request unconditionally, in which case the new request is still basically a direct consequence of the origin request, and the header field should probably be kept.

### 7.3. Relation to Via

The "Via" header field (see [RFC7230], Section 5.7.1) is a header field with a similar use case as this header field. The "Via" header field, however, only provides information about the proxy itself, and thereby leaves out the information about the client connecting to the proxy server. The "Forwarded" header field, on the other hand, has relaying information from the client-facing side of the proxy server as its main purpose. As "Via" is already widely deployed, its format cannot be changed to address the problems that "Forwarded" addresses.

Note that it is not possible to combine information from this header field with the information from the Via header field. Some proxies will not update the "Forwarded" header field, some proxies will not update the Via header field, and some proxies will update both.

### 7.4. Transition

If a proxy gets incoming requests with X-Forwarded-* header fields present, it is encouraged to convert these into the header field described in this document, if it can be done in a sensible way. If the request only contains one type -- for example, X-Forwarded-For -- this can be translated to "Forwarded", by prepending each element with "for=". Note that IPv6 addresses may not be quoted in X-Forwarded-For and may not be enclosed by square brackets, but they are quoted and enclosed in square brackets in "Forwarded".

```
X-Forwarded-For: 192.0.2.43, 2001:db8:cafe::17
```

becomes:

```
Forwarded: for=192.0.2.43, for="[2001:db8:cafe::17]"
```

However, special care must be taken if, for example, both X-Forwarded-For and X-Forwarded-By exist. In such cases, it may not be possible to do a conversion, since it is not possible to know in which order the already existing fields were added. Also, note that removing the X-Forwarded-For header field may cause issues for parties that have not yet implemented support for this new header field.

### 7.5. Example Usage

A request from a client with IP address 192.0.2.43 passes through a proxy with IP address 198.51.100.17, then through another proxy with IP address 203.0.113.60 before reaching an origin server. This could, for example, be an office client behind a corporate malware filter talking to a origin server through a reverse proxy.

- The HTTP request between the client and the first proxy has no "Forwarded" header field.
- The HTTP request between the first and second proxy has a "Forwarded: for=192.0.2.43" header field.
- The HTTP request between the second proxy and the origin server has a "Forwarded: for=192.0.2.43, for=198.51.100.17;by=203.0.113.60;proto=http;host=example.com" header field.

Note that, at some points in a connection chain, the information might not be updated in the "Forwarded" header field, either because of lack of support of this HTTP extension or because of a policy decision not to disclose information about this network component.

## 8. Security Considerations

### 8.1. Header Validity and Integrity

The "Forwarded" HTTP header field cannot be relied upon to be correct, as it may be modified, whether mistakenly or for malicious reasons, by every node on the way to the server, including the client making the request.

One approach to ensure that the "Forwarded" HTTP header field is correct is to verify the correctness of proxies and to whitelist them as trusted. This approach has at least two weaknesses. First, the chain of IP addresses listed before the request came to the proxy cannot be trusted. Second, unless the communication between proxies and the endpoint is secured, the data can be modified by an attacker with access to the network.

### 8.2. Information Leak

The "Forwarded" HTTP header field can reveal internal structures of the network setup behind the NAT or proxy setup, which may be undesired. This can be addressed either by using obfuscated elements, by preventing the internal nodes from updating the HTTP header field, or by having an egress proxy remove entries that reveal internal network information.

This header field should never be copied into response messages by origin servers or intermediaries, as it can reveal the whole proxy chain to the client. As a side effect, special care must be taken in hosting environments not to allow the TRACE request where the "Forwarded" field is used, as it would appear in the body of the response message.

### 8.3. Privacy Considerations

In recent years, there have been growing concerns about privacy. There is a trade-off between ensuring privacy for users versus disclosing information that is useful, for example, for debugging, statistics, and generating location-dependent content. The "Forwarded" HTTP header field, by design, exposes information that some users consider privacy sensitive, in order to allow for such uses. For any proxy, if the HTTP request contains header fields that

specifically request privacy semantics, the proxy SHOULD NOT use the "Forwarded" header field, nor in any other manner pass private information, such as IP addresses, on to the next hop.

The client's IP address, that may be forwarded in the "for" parameter of this header field, is considered to be privacy sensitive by many people, as the IP address may be able to uniquely identify a client, what operator the user is using, and possibly a rough estimation of where the user is geographically located.

Proxies using this extension will preserve the information of a direct connection. This has an end-user privacy impact regardless of whether the end-user or deployer knows or expects that this is the case.

Implementers and deployers of such proxies need to consider whether, and how, deploying this extension affects user privacy.

The default configuration for both the "by" and "for" parameters SHOULD contain obfuscated identifiers. These identifiers SHOULD be randomly generated per request. If identifiers that persist across requests are required, their lifetimes SHOULD be limited and they SHOULD NOT persist longer than client IP addresses. When generating obfuscated identifiers, care must be taken not to include potentially sensitive information in them.

Note that users' IP addresses may already be forwarded by proxies using the header field X-Forwarded-For, which is widely used. It should also be noted that if the user were doing the connection directly without passing the proxy, the client's IP address would be sent to the web server. Users that do not actively choose an anonymizing proxy cannot rely on having their IP address shielded. These users who want to minimize the risk of being tracked must also note that there are other ways information may leak, for example, by browser header field fingerprinting. The Forwarded header field itself, even when used without a uniquely identifying client identifier, may make fingerprinting more feasible by revealing the chain of proxies traversed by the client's request.

## 9. IANA Considerations

This document specifies the HTTP header field listed below, which has been added to the "Permanent Message Header Field Names" registry defined in [RFC3864].

Header field: Forwarded
Applicable protocol: http
Status: standard
Author/Change controller:
    IETF (iesg@ietf.org)
    Internet Engineering Task Force
Specification document(s): this specification (Section 4)
Related information: None

The "Forwarded" header field contains parameters for which IANA has created and now maintains a new registry entitled "HTTP Forwarded Parameters". Initial registrations are given below. For future assignments, the registration procedure is IETF Review [RFC5226]. The security and privacy implications of all new parameters should be thoroughly documented. New parameters and their values MUST conform with the forwarded-pair as defined in ABNF in Section 4. Further, a short description should be provided in the registration.

| Parameter name | Description | Reference |
|---|---|---|
| by | IP address of incoming interface of a proxy | Section 5.1 |
| for | IP address of client making a request through a proxy | Section 5.2 |
| host | Host header field of the incoming request | Section 5.3 |
| proto | Application protocol used for incoming request | Section 5.4 |

Table 1: Initial Assignments

## 10. References

### 10.1. Normative References

[RFC1918] Rekhter, Y., Moskowitz, R., Karrenberg, D., Groot, G., and E. Lear, "Address Allocation for Private Internets", BCP 5, RFC 1918, February 1996.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[RFC3864] Klyne, G., Nottingham, M., and J. Mogul, "Registration Procedures for Message Header Fields", BCP 90, RFC 3864, September 2004.

[RFC3986] Berners-Lee, T., Fielding, R., and L. Masinter, "Uniform Resource Identifier (URI): Generic Syntax", STD 66, RFC 3986, January 2005.

[RFC4193] Hinden, R. and B. Haberman, "Unique Local IPv6 Unicast Addresses", RFC 4193, October 2005.

[RFC4395] Hansen, T., Hardie, T., and L. Masinter, "Guidelines and Registration Procedures for New URI Schemes", BCP 35, RFC 4395, February 2006.

[RFC5226] Narten, T. and H. Alvestrand, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 5226, May 2008.

[RFC5234] Crocker, D. and P. Overell, "Augmented BNF for Syntax Specifications: ABNF", STD 68, RFC 5234, January 2008.

[RFC5952] Kawamura, S. and M. Kawashima, "A Recommendation for IPv6 Address Text Representation", RFC 5952, August 2010.

[RFC7230] Fielding, R., Ed. and J. Reschke, Ed., "Hypertext Transfer Protocol (HTTP/1.1): Message Syntax and Routing", RFC 7230, June 2014.

[RFC7232] Fielding, R., Ed. and J. Reschke, Ed., "Hypertext Transfer Protocol (HTTP/1.1): Conditional Requests", RFC 7232, June 2014.

## 10.2. Informative References

[RFC6269] Ford, M., Boucadair, M., Durand, A., Levis, P., and P. Roberts, "Issues with IP Address Sharing", RFC 6269, June 2011.

## Appendix A. Acknowledgments

Thanks to Per Cederqvist, Alissa Cooper, Adrian Farrel, Stephen Farrell, Ned Freed, Per Hedbor, Amos Jeffries, Poul-Henning Kamp, Murray S. Kucherawy, Barry Leiba, Salvatore Loreto, Alexey Melnikov, S. Moonesamy, Susan Nichols, Mark Nottingham, Julian Reschke, John Sullivan, Willy Tarreau, and Dan Wing for their feedback.

## Authors' Addresses

Andreas Petersson
Opera Software

EMail: andreas@sbin.se

Martin Nilsson
Opera Software
S:t Larsgatan 12
Linkoping SE-582 24

EMail: nilsson@opera.com