Network Working Group
Request for Comments: 2774
Category: Experimental

H. Nielsen
P. Leach
Microsoft
S. Lawrence
Agranat Systems
February 2000

An HTTP Extension Framework

Status of this Memo

This memo defines an Experimental Protocol for the Internet community. It does not specify an Internet standard of any kind. Discussion and suggestions for improvement are requested. Distribution of this memo is unlimited.

IESG Note

This document was originally requested for Proposed Standard status. However, due to mixed reviews during Last Call and within the HTTP working group, it is being published as an Experimental document. This is not necessarily an indication of technical flaws in the document; rather, there is a more general concern about whether this document actually represents community consensus regarding the evolution of HTTP. Additional study and discussion are needed before this can be determined.

Note also that when HTTP is used as a substrate for other protocols, it may be necessary or appropriate to use other extension mechanisms in addition to, or instead of, those defined here. This document should therefore not be taken as a blueprint for adding extensions to HTTP, but it defines mechanisms that might be useful in such circumstances.

Copyright Notice

Abstract

A wide range of applications have proposed various extensions of the HTTP protocol. Current efforts span an enormous range, including distributed authoring, collaboration, printing, and remote procedure call mechanisms. These HTTP extensions are not coordinated, since there has been no standard framework for defining extensions and thus, separation of concerns. This document describes a generic extension mechanism for HTTP, which is designed to address the tension between private agreement and public specification and to accommodate extension of applications using HTTP clients, servers, and proxies. The proposal associates each extension with a globally unique identifier, and uses HTTP header fields to carry the extension identifier and related information between the parties involved in the extended communication.

Table of Contents

1. Introduction

This proposal is designed to address the tension between private agreement and public specification; and to accommodate dynamic extension of HTTP clients and servers by software components. The kind of extensions capable of being introduced range from:

- extending a single HTTP message;
- introducing new encodings;
- initiating HTTP-derived protocols for new applications; to...
- switching to protocols which, once initiated, run independent of the original protocol stack.

The proposal is intended to be used as follows:

- Some party designs and specifies an extension; the party assigns the extension a globally unique URI, and makes one or more representations of the extension available at that address (see Section 8).
- An HTTP client or server that implements this extension mechanism (hereafter called an agent) declares the use of the extension by referencing its URI in an extension declaration in an HTTP message (see Section 3).
- The HTTP application which the extension declaration is intended for (hereafter called the ultimate recipient) can deduce how to properly interpret the extended message based on the extension declaration.

The proposal uses features in HTTP/1.1 but is compatible with HTTP/1.0 applications in such a way that extended applications can coexist with existing HTTP applications. Applications implementing this proposal MUST be based on HTTP/1.1 (or later versions of HTTP).

2. Notational Conventions

This specification uses the same notational conventions and basic parsing constructs as RFC 2068 [5]. In particular the BNF constructs "token", "quoted-string", "Request-Line", "field-name", and "absoluteURI" in this document are to be interpreted as described in RFC 2068 [5].

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 RFC 2119 [6].

This proposal does not rely on particular features defined in URLs [8] that cannot potentially be expressed using URNs (see Section 8). Therefore, the more generic term URI [8] is used throughout the specification.

3. Extension Declarations

An extension declaration can be used to indicate that an extension has been applied to a message and possibly to reserve a part of the header namespace identified by a header field prefix (see 3.1). This section defines the extension declaration itself; Section 4 defines a set of header fields using the extension declaration.

This specification does not define any ramifications of applying an extension to a message nor whether two extensions can or cannot logically coexist within the same message. It is simply a framework for describing which extensions have been applied and what the ultimate recipient either must or may do in order to properly interpret any extension declarations within that message.

The grammar for an extension declaration is as follows:

```
ext-decl        = <"> ( absoluteURI | field-name ) <">
                  [ namespace ] [ decl-extensions ]

namespace       = ";" "ns" "=" header-prefix
header-prefix   = 2*DIGIT

decl-extensions = *( decl-ext )
decl-ext        = ";" token [ "=" ( token | quoted-string ) ]
```

An extension is identified by an absolute, globally unique URI or a field-name. A field-name MUST specify a header field uniquely defined in an IETF Standards Track RFC [3]. A URI can unambiguously be distinguished from a field-name by the presence of a colon (":").

The support for header field names as extension identifiers provides a transition strategy from decentralized extensions to extensions defined by IETF Standards Track RFCs until a mapping between the globally unique URI space and features defined in IETF Standards Track RFCs has been defined according to the guidelines described in Section 8.

Examples of extension declarations are

```
"http://www.company.com/extension"; ns=11
"Range"
```

An agent MAY use the decl-extensions mechanism to include optional extension declaration parameters but cannot assume these parameters to be recognized by the recipient. An agent MUST NOT use decl-extensions to pass extension instance data, which MAY be passed using header field prefix values (see Section 3.1). Unrecognized decl-ext parameters SHOULD be ignored and MUST NOT be removed by proxies when forwarding the extension declaration.

3.1. Header Field Prefixes

The header-prefix is a dynamically generated string. All header fields in the message that match this string, using string prefix-matching, belong to that extension declaration. Header field prefixes allow an extension declaration to dynamically reserve a subspace of the header space in a protocol message in order to prevent header field name clashes and to allow multiple declarations using the same extension to be applied to the same message without conflicting.

Header fields using a header-prefix are of the form:

```
prefixed-header = prefix-match field-name
prefix-match    = header-prefix "-"
```

Linear white space (LWS) MUST NOT be used between the header-prefix and the dash ("-") or between the prefix-match and the field-name. The string prefix matching algorithm is applied to the prefix-match string.

The format of the prefix using a combination of digits and the dash ("-") guarantees that no extension declaration can reserve the whole header field name space. The header-prefix mechanism was preferred over other solutions for exchanging extension instance parameters because it is header based and therefore allows for easy integration of new extensions with existing HTTP features.

Agents MUST NOT reuse header-prefix values in the same message unless explicitly allowed by the extension (see Section 4.1 for a discussion of the ultimate recipient of an extension declaration).

Clients SHOULD be as consistent as possible when generating header-prefix values as this facilitates use of the Vary header field in responses that vary as a function of the request extension declaration(s) (see [5], section 13.6).

Servers including prefixed-header header fields in a Vary header field value MUST also include the corresponding extension declaration field-name as part of that value. For example, if a response depends on the value of the 16-use-transform header field defined by an optional extension declaration in the request, the Vary header field in the response could look like this:

```
Vary: Opt, 16-use-transform
```

Note, that header-prefix consistency is no substitute for including an extension declaration in the message: header fields with header-prefix values not defined by an extension declaration in the same message are not defined by this specification.

Examples of header-prefix values are

```
12
15
23
```

Old applications may introduce header fields independent of this extension mechanism, potentially conflicting with header fields introduced by the prefix mechanism. In order to minimize this risk, prefixes MUST contain at least 2 digits.

4. Extension Header Fields

This proposal introduces two types of extension declaration strength: mandatory and optional, and two types of extension declaration scope: hop-by-hop and end-to-end (see section 4.1 and 4.2).

A mandatory extension declaration indicates that the ultimate recipient MUST consult and adhere to the rules given by the extension when processing the message or reporting an error (see section 5 and 7).

An optional extension declaration indicates that the ultimate recipient of the extension MAY consult and adhere to the rules given by the extension when processing the message, or ignore the extension declaration completely. An agent may not be able to distinguish whether the ultimate recipient does not understand an extension referred to by an optional extension or simply ignores the extension declaration.

The combination of the declaration strength and scope defines a 2x2 matrix which is distinguished by four new general HTTP header fields: Man, Opt, C-Man, and C-Opt. (See sections 4.1 and 4.2); also see Appendix 14, which has a table of interactions with origin servers and proxies.)

The header fields are general header fields as they describe which extensions actually are applied to an HTTP message. Optional declarations MAY be applied to any HTTP message if appropriate (see Section 5 for how to apply mandatory extension declarations to requests and Section 6 for how to apply them to responses).

4.1. End-to-End Extensions

End-to-end declarations MUST be transmitted to the ultimate recipient of the declaration. The Man and the Opt general header fields are end-to-end header fields and are defined as follows:

```
mandatory       = "Man" ":" 1#ext-decl
optional        = "Opt" ":" 1#ext-decl
```

For example

```
HTTP/1.1 200 OK
Content-Length: 421
Opt: "http://www.digest.org/Digest"; ns=15
15-digest: "snfksjgor2tsajkt52"
...
```

The ultimate recipient of a mandatory end-to-end extension declaration MUST handle that extension declaration as described in section 5 and 6.

4.2. Hop-by-Hop Extensions

Hop-by-hop extension declarations are meaningful only for a single HTTP connection. In HTTP/1.1, C-Man, C-Opt, and all header fields with matching header-prefix values defined by C-Man and C-Opt MUST be protected by a Connection header field. That is, these header fields are to be included as Connection header field directives (see [5], section 14.10). The two header fields have the following grammar:

```
c-mandatory     = "C-Man" ":" 1#ext-decl
c-optional      = "C-Opt" ":" 1#ext-decl
```

For example

```
M-GET / HTTP/1.1
Host: some.host
C-Man: "http://www.digest.org/ProxyAuth"; ns=14
14-Credentials="g5gj262jdw@4df"
Connection: C-Man, 14-Credentials
```

The ultimate recipient of a mandatory hop-by-hop extension declaration MUST handle that extension declaration as described in section 5 and 6.

4.3. Extension Response Header Fields

Two extension response header fields are used to indicate that a request containing mandatory extension declarations has been fulfilled by the ultimate recipient as described in Section 5.1. The extension response header fields are exclusively intended to serve as extension acknowledgements, and can not carry any other information.

The Ext header field is used to indicate that all end-to-end mandatory extension declarations in the request were fulfilled:

```
ext             = "Ext" ":"
```

The C-Ext response header field is used to indicate that all hop-by-hop mandatory extension declarations in the request were fulfilled.

```
c-ext           = "C-Ext" ":"
```

In HTTP/1.1, the C-Ext header fields MUST be protected by a Connection header (see [5], section 14.10).

The Ext and the C-Ext header fields are not mutually exclusive; they can both occur within the same message as described in Section 5.1.

5. Mandatory HTTP Requests

An HTTP request is called a mandatory request if it includes at least one mandatory extension declaration (using the Man or the C-Man header fields). The method name of a mandatory request MUST be prefixed by "M-". For example, a client might express the binding rights-management constraints in an HTTP PUT request as follows:

```
M-PUT /a-resource HTTP/1.1
Man: "http://www.copyright.org/rights-management"; ns=16
16-copyright: http://www.copyright.org/COPYRIGHT.html
16-contributions: http://www.copyright.org/PATCHES.html
Host: www.w3.org
Content-Length: 1203
Content-Type: text/html

<!doctype html ...
```

An ultimate recipient conforming to this specification receiving a mandatory request MUST process the request by performing the following actions in the order listed below:

1. Identify all mandatory extension declarations (both hop-by-hop and end-to-end); the server MAY ignore optional declarations without affecting the result of processing the HTTP message;
2. Examine all extensions identified in 1) and determine if they are supported for this message. If not, respond with a 510 (Not Extended) status-code (see Section 7);
3. If 2) did not result in a 510 (Not Extended) status code, then process the request according to the semantics of the extensions and of the existing HTTP method name as defined in HTTP/1.1 [5] or later versions of HTTP. The HTTP method name can be obtained by ignoring the "M-" method name prefix.
4. If the evaluation in 3) was successful and the mandatory request fulfilled, the server MUST respond as defined in Section 5.1. A server MUST NOT fulfill a request without understanding and obeying all mandatory extension declaration(s) in a request.

A proxy that does not act as the ultimate recipient of a mandatory extension declaration MUST NOT remove the extension declaration or the "M-" method name prefix when forwarding the message (see Section 5.1 for how to detect when a mandatory extension has been fulfilled).

A server receiving an HTTP/1.0 (or earlier versions of HTTP) message that includes a Connection header MUST, for each connection-token in this field, remove and ignore any header field(s) from the message with the same name as the connection-token.

A server receiving a mandatory request including the "M-" method name prefix without any mandatory extension declarations to follow MUST return a 510 (Not Extended) response.

The "M-" prefix is reserved by this proposal and MUST NOT be used by other HTTP extensions.

5.1. Fulfilling a Mandatory Request

A server MUST NOT claim to have fulfilled any mandatory request unless it understood and obeyed all the mandatory extension declarations in the request. This section defines a mechanism for conveying this information to the client in such a way that it interoperates with existing HTTP applications and prevents broken servers from giving the false impression that an extended request was fulfilled by responding with a 200 (Ok) response without understanding the method.

If any end-to-end mandatory extension declarations were among the fulfilled extensions then the server MUST include an Ext response header field in the response. In order to avoid that the Ext header field inadvertently is cached in an HTTP/1.1 cache, the response MUST contain a no-cache cache-control directive. If the response is otherwise cachable, the no-cache cache-control directive SHOULD be limited to only affect the Ext header field:

```
HTTP/1.1 200 OK
Ext:
Cache-Control: no-cache="Ext"
...
```

If the mandatory request has been forwarded by an HTTP/1.0 intermediary proxy then this is indicated either directly in the Request-Line or by the presence of an HTTP/1.1 Via header field. In this case, the server MUST include an Expires header field with a date equal to or earlier than the value of the Date header field (see Section 9 for a discussion on caching considerations):

```
HTTP/1.1 200 OK
Date: Sun, 25 Oct 1998 08:12:31 GMT
Expires: Sun, 25 Oct 1998 08:12:31 GMT
Ext:
Cache-Control: no-cache="Ext", max-age=3600
...
```

If any hop-by-hop mandatory extension declarations were among the fulfilled extensions then the server MUST include a C-Ext response header field in the response. The C-Ext header field MUST be protected by a Connection header field (see [5], section 14.10).

```
HTTP/1.1 200 OK
C-Ext:
Connection: C-Ext
```

Note, that the Ext and C-Ext header fields are not mutually exclusive; they can be both be present in a response when fulfilling mandatory request containing both hop-by-hop as well as end-to-end mandatory extension declarations.

6. Mandatory HTTP Responses

A server MUST NOT include mandatory extension declarations in an HTTP response unless it is responding to a mandatory HTTP request whose definition allowed for the mandatory response or the server has some a priori knowledge that the recipient can handle the extended response. A server MAY include optional extension declarations in any HTTP response (see Section 4).

If a client is the ultimate recipient of a mandatory HTTP response containing mandatory extension declarations that either the client does not understand or does not want to use, then it SHOULD discard the complete response as if it were a 500 (Internal Server Error) response.

7. 510 Not Extended

The policy for accessing the resource has not been met in the request. The server should send back all the information necessary for the client to issue an extended request. It is outside the scope of this specification to specify how the extensions inform the client.

If the 510 response contains information about extensions that were not present in the initial request then the client MAY repeat the request if it has reason to believe it can fulfill the extension policy by modifying the request according to the information provided in the 510 response. Otherwise the client MAY present any entity included in the 510 response to the user, since that entity may include relevant diagnostic information.

8. Publishing an Extension

While the protocol extension definition should be published at the address of the extension identifier, this specification does not require it. The only absolute requirement is that extension identifiers MUST be globally unique identifiers, and that distinct names be used for distinct semantics.

Likewise, applications are not required to attempt resolving extension identifiers included in an extension declaration. The only absolute requirement is that an application MUST NOT claim conformance with an extension that it does not recognize (regardless of whether it has tried to resolve the extension identifier or not). This document does not provide any policy for how long or how often an application may attempt to resolve an extension identifier.

The association between the extension identifier and the specification might be made by distributing a specification, which references the extension identifier.

It is strongly recommended that the integrity and persistence of the extension identifier be maintained and kept unquestioned throughout the lifetime of the extension. Care should be taken not to distribute conflicting specifications that reference the same name. Even when an extension specification is made available at the address of the URI, care must be taken that the specification made available at that address does not change over time. One agent may associate the identifier with the old semantics, while another might associate it with the new semantics.

The extension definition may be made available in different representations ranging from

- a human-readable specification defining the extension semantics (see for example [7]),
- downloadable code which implements the semantics defined by the extension,
- a formal interface description provided by the extension, to
- a machine-readable specification defining the extension semantics.

For example, a software component that implements the specification may reside at the same address as a human-readable specification (distinguished by content negotiation). The human-readable representation serves to document the extension and encourage deployment, while the software component would allow clients and servers to be dynamically extended.

9. Caching Considerations

Use of extensions using the syntax defined by this document may have additional implications on the cachability of HTTP response messages other than the ones described in Section 5.1.

The originator of an extended message should be able to determine from the semantics of the extension whether or not the extension's presence impacts the caching constraints of the response message. If an extension does require tighter constraints on the cachebility of the response, the originator MUST include the appropriate combination of cache header fields (Cache-Control, Vary, Expires) corresponding to the required level of constraints of the extended semantics.

10. Security Considerations

Dynamic installation of extension facilities as described in the introduction involves software written by one party (the provider of the implementation) to be executed under the authority of another (the party operating the host software). This opens the host party to a variety of "Trojan horse" attacks by the provider, or a malicious third party that forges implementations under a provider's name. See, for example RFC2046 [4], section 4.5.2 for a discussion of these risks.

11. Acknowledgements

Roy Fielding, Rohit Khare, Yaron Y. Goland, and Koen Holtman, deserve special recognition for their efforts in commenting in all phases of this specification. Also thanks to Josh Cohen, Ross Patterson, Jim Gettys, Larry Masinter, and to the people involved in PEP [9].

The contribution of World Wide Web Consortium (W3C) staff is part of the W3C HTTP Activity (see "http://www.w3.org/Protocols/Activity").

12. References

[1] Crocker, D., "Standard for the format of ARPA Internet text messages", STD 11, RFC 822, August 1982.

[2] Berners-Lee, T., Fielding, R., and H. Nielsen, "Hypertext Transfer Protocol -- HTTP/1.0", RFC 1945, May 1996.

[3] Bradner, S., "The Internet Standards Process -- Revision 3", BCP 9, RFC 2026, October 1996.

[4] Freed, N. and N. Borenstein, "Multipurpose Internet Mail Extensions (MIME) Part Two: Media Types", RFC 2046, November 1996.

[5] Fielding, R., Gettys, J., Mogul, J., Nielsen, H., and T. Berners-Lee, "Hypertext Transfer Protocol -- HTTP/1.1", RFC 2068, January 1997.

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

[7] Masinter, L., "Hyper Text Coffee Pot Control Protocol (HTCPCP/1.0)", RFC 2324, April 1998.

[8] Berners-Lee, T., Fielding, R., and L. Masinter, "Uniform Resource Identifiers (URI): Generic Syntax", RFC 2396, August 1998.

[9] Nielsen, H., Conolly, D., and R. Khare, "PEP - an extension mechanism for HTTP".
Work in progress.

14. Summary of Protocol Interactions

The following tables summarize the outcome of strength and scope rules of the mandatory proposal of compliant and non-compliant HTTP proxies and origin servers. The summary is intended as a guide and index to the text, but is necessarily cryptic and incomplete. This summary should never be used or referenced separately from the complete specification.

Table 1: Origin Server

Scope	Hop-by-hop		End-to-end	
Strength	Optional (may)	Required (must)	Optional (may)	Required (must)
Mandatory unsupported	Standard processing	501 (Not Implemented)	Standard processing	501 (Not Implemented)
Extension unsupported	Standard processing	510 (Not Extended)	Standard processing	510 (Not Extended)
Extension supported	Extended processing	Extended processing	Extended processing	Extended processing

Table 2: Proxy Server

Scope	Hop-by-hop		End-to-end	
Strength	Optional (may)	Required (must)	Optional (may)	Required (must)
Mandatory unsupported	Strip extension	501 (Not Implemented) or tunnel	Forward extension	501 (Not Implemented) or tunnel
Extension unsupported	Strip extension	510 (Not Extended)	Forward extension	Forward extension
Extension supported	Extended processing and strip	Extended processing and strip	Extended processing, may strip	Extended processing, may strip

15. Examples

The following examples show various scenarios using mandatory in HTTP/1.1 requests and responses. Information not essential for illustrating the examples is left out (referred to as "...")

15.1. User Agent to Origin Server

Table 3: User Agent directly to origin server

```
Client issues a request  M-GET /some-document HTTP/1.1
with one optional and    Opt: "http://www.my.com/tracking"
one mandatory extension  Man: "http://www.foo.com/privacy"
                         ...

Origin server accepts    HTTP/1.1 200 OK
the mandatory extension  Ext:
but ignores the          Cache-Control: max-age=120, no-cache="Ext"
optional one. The        ...
client can not see in
this case that the
optional extension was
ignored.
```

Table 4: Origin server with Vary header field

```
Client issues a request  M-GET /p/q HTTP/1.1
with one mandatory       Man: "http://www.x.y/transform"; ns=16
extension                16-use-transform: xyzzy
                         ...

Origin server accepts    HTTP/1.1 200 OK
the mandatory but        Ext:
indicates that the       Vary: Man, 16-use-transform
response varies on the   Date: Sun, 25 Oct 1998 08:12:31 GMT
request extension        Expires: Sun, 25 Oct 1998 08:12:31 GMT
declaration              Cache-Control: no-cache="Ext", max-age=1000
                         ...
```

15.2. User Agent to Origin Server via HTTP/1.1 Proxy

These two examples show how an extended request interacts with an HTTP/1.1 proxy.

Table 5: HTTP/1.1 Proxy forwards extended request

```
Client issues a request  M-GET /some-document HTTP/1.1
with one optional and    C-Opt: "http://www.meter.org/hits"
one mandatory hop-by-    C-Man: "http://www.copy.org/rights"
hop extension            Connection: C-Opt, C-Man
                         ...

HTTP/1.1 proxy forwards  M-GET /some-document HTTP/1.1
the request and takes    Via: 1.1 new
out the connection       ...
headers

Origin server fails as   HTTP/1.1 510 Not Extended
the request does not     ...
contain any information
belonging to the M-GET
method
```

Table 6: HTTP/1.1 Proxy does not forward extended request

```
Client issues a request  M-GET /some-document HTTP/1.1
with one optional and    C-Opt: "http://www.meter.org/hits"
one mandatory hop-by-    C-Man: "http://www.copy.org/rights"
hop extension            Connection: C-Opt, C-Man
                         ...

HTTP/1.1 proxy refuses   HTTP/1.1 501 Not Implemented
to forward the M-GET     ...
method and returns an
error

Origin server never
sees the extended
request
```

15.3. User Agent to Origin Server via HTTP/1.0 Proxy

These two examples show how an extended request interacts with an HTTP/1.0 proxy in the message path

Table 7: HTTP/1.0 Proxy forwards extended request

```
Client issues a request  M-GET /some-document HTTP/1.1
with one mandatory       Man: "http://www.price.com/sale"
extension                ...

HTTP/1.0 proxy forwards  M-GET /some-document HTTP/1.0
the request as a         Man: "http://www.price.com/sale"
HTTP/1.0 request         ...
without changing the
method

Origin server accepts    HTTP/1.1 200 OK
declaration and returns  Ext:
a 200 response and an    Date: Sun, 25 Oct 1998 08:12:31 GMT
extension                Expires: Sun, 25 Oct 1998 08:12:31 GMT
acknowledgement. The     Cache-Control: no-cache="Ext", max-age=600
response can be cached   ...
by HTTP/1.1 caches for
10 minutes.
```

Table 8: HTTP/1.0 and HTTP/1.1 Proxy Chain

```
Client issues request    M-GET /some-document HTTP/1.1
with one mandatory and   Man: "http://www.copy.org/rights"
one hop-by-hop optional  C-Opt: "http://www.ads.org/noads"
extension                Connection: C-Opt
                         ...

HTTP/1.0 proxy forwards  M-GET /some-document HTTP/1.0
request as HTTP/1.0      Man: "http://www.copy.org/rights"
request without          C-Opt: "http://www.ads.org/noads"
changing the method and  Connection: C-Man
without honoring the     ...
Connection directives

HTTP/1.1 proxy deletes   M-GET /some-document HTTP/1.1
(and ignores) optional   Man: "http://www.copy.org/rights"
extension and forwards   C-Man: "http://www.ads.org/givemeads"
the rest including a     Connection: C-Man
via header field. It     Via: 1.0 new
also add a hop-by-hop    ...
mandatory extension

Origin server accepts    HTTP/1.1 200 OK
both mandatory           Ext:
extensions. The          C-Ext
response is not          Connection: C-Ext
cachable by the          Date: Sun, 25 Oct 1998 08:12:31 GMT
HTTP/1.0 cache but can   Expires: Sun, 25 Oct 1998 08:12:31 GMT
be cached for 1 hour by  Cache-Control: no-cache="Ext", max-age=3600
HTTP/1.1 caches.         ...

HTTP/1.1 proxy removes   HTTP/1.1 200 OK
the hop-by-hop           Ext:
extension                Date: Sun, 25 Oct 1998 08:12:31 GMT
acknowledgement and      Expires: Sun, 25 Oct 1998 08:12:31 GMT
forwards the remainder   Cache-Control: no-cache="Ext", max-age=3600
of the response.         ...
```

Index

13. Authors' Addresses

Henrik Frystyk Nielsen
Microsoft Corporation
1 Microsoft Way
Redmond, WA 98052
USA
EMail: frystyk@microsoft.com

Paul J. Leach
Microsoft Corporation
1 Microsoft Way
Redmond, WA 98052
USA
EMail: paulle@microsoft.com

Scott Lawrence
Agranat Systems, Inc.
5 Clocktower Place, Suite 400
Maynard, MA 01754
USA
EMail: lawrence@agranat.com

Full Copyright Statement

Intellectual Property

Acknowledgment

Funding for the RFC Editor function is currently provided by the Internet Society.