Office de la Propriété Intellectuelle du Canada
Un organisme d'Industrie Canada

Canadian Intellectual Property Office
An agency of Industry Canada

CA 2617059 C 2013/07/23

(11)(21) 2 617 059

(12) BREVET CANADIEN
CANADIAN PATENT

(13) C

(86) Date de dépôt PCT/PCT Filing Date: 2006/08/15
(87) Date publication PCT/PCT Publication Date: 2007/03/08
(45) Date de délivrance/Issue Date: 2013/07/23
(85) Entrée phase nationale/National Entry: 2008/01/28
(86) N° demande PCT/PCT Application No.: US 2006/031705
(87) N° publication PCT/PCT Publication No.: 2007/027426
(30) Priorité/Priority: 2005/08/31 (US11/217,626)

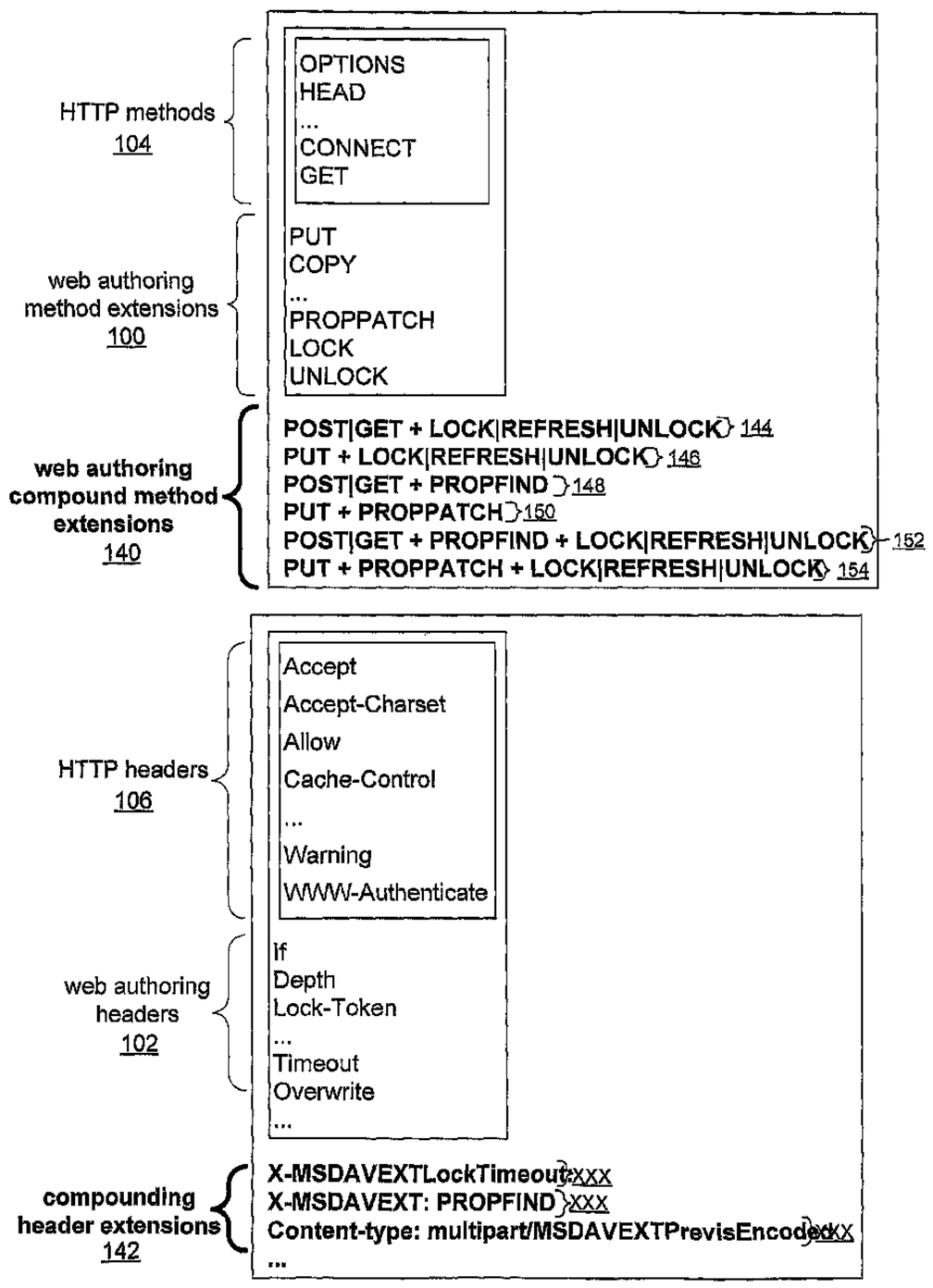
(51) Cl.Int./Int.Cl. *H04L 29/06* (2006.01),
H04L 12/66 (2006.01)

(72) Inventeurs/Inventors:
CHINTALAPATI, V.R. KISHORE, US;
KRUSE, DAVID M., US;
MOHAMED, AHMED H., US;
WATSON, ANDREW SEAN, US;
FRIESENHAHN, DUSTIN G., US;
PAULUS, JAY, US;
SUBBARAYAN, SUNDAR, US;
MCATEER, SEAN, US

(73) Propriétaire/Owner:
MICROSOFT CORPORATION, US

(74) Agent: SMART & BIGGAR

(54) Titre : COMPOSITION DE PROTOCOLE DE CREATION HTTP
(54) Title: COMPOUNDING OF HTTP AUTHORING PROTOCOL



(57) Abrégé/Abstract:
Conventions for extending compounded web authoring methods to a web authoring protocol such as WebDAV. More particularly, a request can be provided with special header information to signify a method compounded with a method indicated by a verb in the request. Techniques for clients and servers to use the web authoring extensions. Extended error handling to allow servers to provider richer web authoring error information to clients.

(12) INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
8 March 2007 (08.03.2007)

PCT

(10) International Publication Number
WO 2007/027426 A1

(51) International Patent Classification:
H04L 12/66 (2006.01) *G06F 17/00* (2006.01)
H04L 29/06 (2006.01)

(21) International Application Number:
PCT/US2006/031705

(22) International Filing Date: 15 August 2006 (15.08.2006)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
11/217,626 31 August 2005 (31.08.2005) US

(71) Applicant *(for all designated States except US)*: **MICROSOFT CORPORATION** [US/US]; One Microsoft Way, Redmond, Washington 98052-6399 (US).

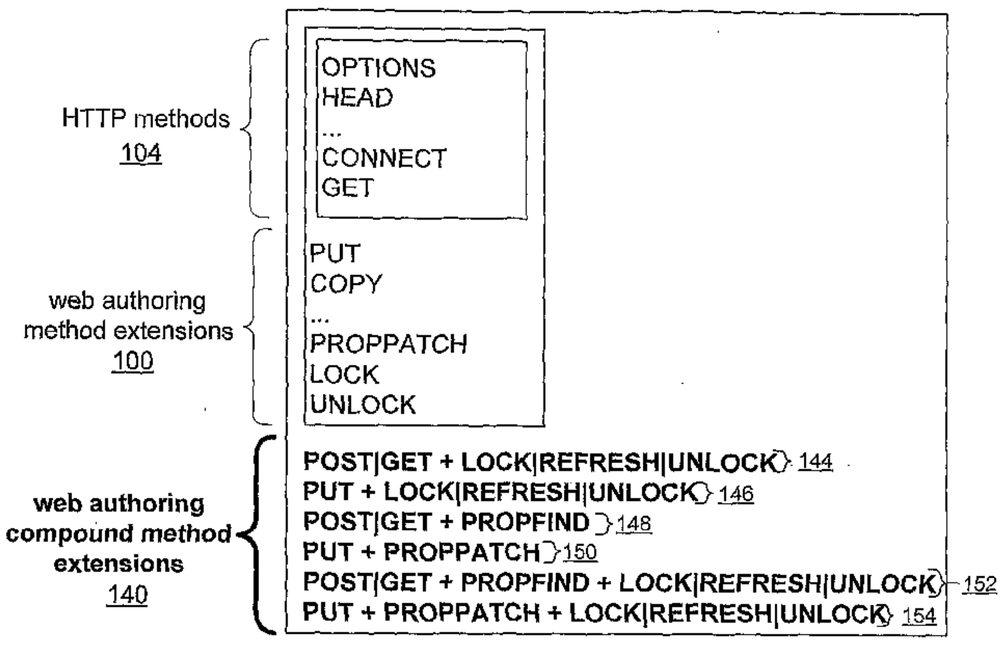
(72) Inventors: **CHINTALAPATI, V.R. Kishore**; One Microsoft Way, Redmond, Washington 98052-6399 (US). **KRUSE, David M.**; One Microsoft Way, Redmond, Washington 98052-6399 (US). **MOHAMED, Ahmed H.**; One Microsoft Way, Redmond, Washington 98052-6399 (US). **WATSON, Andrew Sean**; One Microsoft Way, Redmond, Washington 98052-6399 (US). **FRIESENHAHN, Dustin G.**; One Microsoft Way, Redmond, Washington 98052-6399 (US). **PAULUS, Jay**; One Microsoft Way, Redmond, Washington 98052-6399 (US). **SUBBARAYAN, Sundar**; One Microsoft Way, Redmond, Washington 98052-6399 (US). **MCATEER, Sean**; One Microsoft Way, Redmond, Washington 98052-6399 (US).

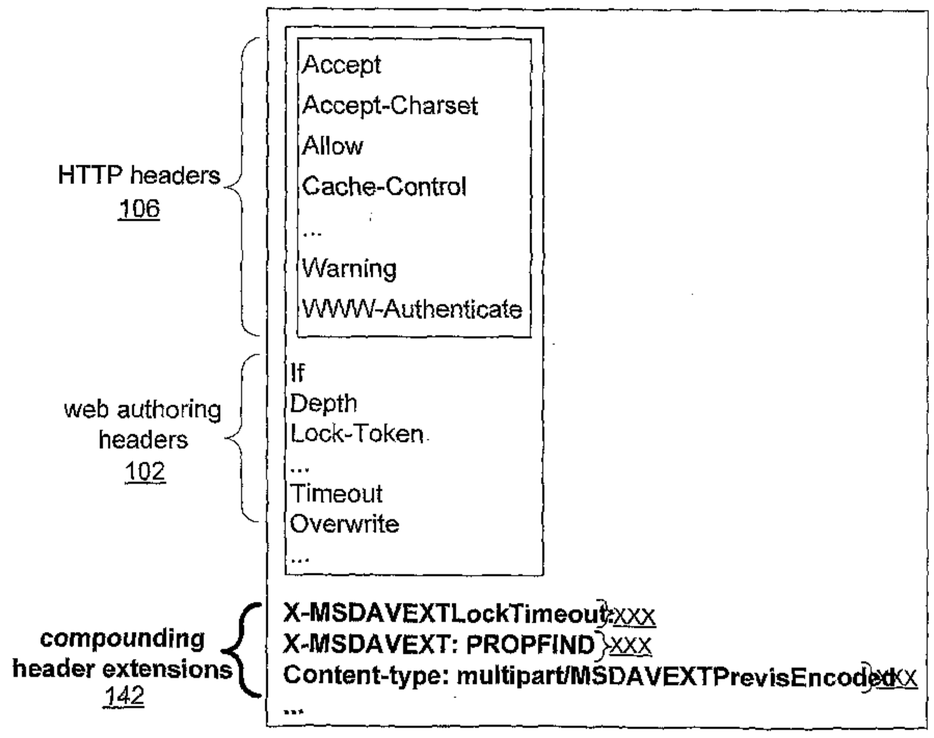
(81) Designated States *(unless otherwise indicated, for every kind of national protection available)*: AE, AG, AL, AM, AT, AU, AZ, BA, BB, BG, BR, BW, BY, BZ, CA, CH, CN, CO, CR, CU, CZ, DE, DK, DM, DZ, EC, EE, EG, ES, FI, GB, GD, GE, GH, GM, HN, HR, HU, ID, IL, IN, IS, JP, KE, KG, KM, KN, KP, KR, KZ, LA, LC, LK, LR, LS, LT, LU, LV, LY, MA, MD, MG, MK, MN, MW, MX, MZ, NA, NG, NI, NO, NZ, OM, PG, PH, PL, PT, RO, RS, RU, SC, SD, SE, SG, SK, SL, SM, SY, TJ, TM, TN, TR, TT, TZ, UA, UG, US, UZ, VC, VN, ZA, ZM, ZW.

(84) Designated States *(unless otherwise indicated, for every kind of regional protection available)*: ARIPO (BW, GH, GM, KE, LS, MW, MZ, NA, SD, SL, SZ, TZ, UG, ZM,

[Continued on next page]

(54) Title: COMPOUNDING OF HTTP AUTHORING PROTOCOL





(57) Abstract: Conventions for extending compounded web authoring methods to a web authoring protocol such as WebDAV. More particularly, a request can be provided with special header information to signify a method compounded with a method indicated by a verb in the request. Techniques for clients and servers to use the web authoring extensions. Extended error handling to allow servers to provider richer web authoring error information to clients.

WO 2007/027426 A1

WO 2007/027426 A1

ZW), Eurasian (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European (AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, HU, IE, IS, IT, LT, LU, LV, MC, NL, PL, PT, RO, SE, SI, SK, TR), OAPI (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, ML, MR, NE, SN, TD, TG).

Declarations under Rule 4.17:

— *as to applicant's entitlement to apply for and be granted a patent (Rule 4.17(ii))*

— *as to the applicant's entitlement to claim the priority of the earlier application (Rule 4.17(iii))*

Published:

— *with international search report*

— *before the expiration of the time limit for amending the claims and to be republished in the event of receipt of amendments*

For two-letter codes and other abbreviations, refer to the "Guidance Notes on Codes and Abbreviations" appearing at the beginning of each regular issue of the PCT Gazette.

COMPOUNDING OF HTTP AUTHORING PROTOCOL

BACKGROUND

[0001] Figure 1 shows an HTTP message 50. The exchange of HTTP messages 50 between an HTTP client 52 and an HTTP server 54 is well known in the art of client-server computing. Various RFCs and other public documents may be consulted for details about the various versions and variations of HTTP. For instance, RFC 2616 defines HTTP version 1.1. According to RFC 2616, an HTTP message 50 that is for an HTTP request has a request line 54, such as "GET / HTTP/1.1". An HTTP message 50 that is for an HTTP response instead has a status line 56, such as "HTTP/1.1 200 OK". A request line 54 or status line 56 is usually followed by one or more headers, each consisting of a field name 60 and, depending on the particular header, zero or more field bodies 62. A message 50 may end with a message body 64, depending on the type of request or response. Details relating to delimiters, particular headers, and other features of HTTP messages and HTTP communication can be found elsewhere.

[0002] Figure 2 shows an example HTTP request 80 and an example HTTP response 82. The HTTP client 52 sends request 80 over a data network 84 to the HTTP server 54, which handles the request and returns the response 82. The request 80 includes a request line 87 and a number of headers 88 (some requests also have a message body). The response 82 includes a status line 89, headers 90, and a message body 92. HTTP communications need not travel over a network such as network 84; communication between a local client and a local server is possible, albeit usually through the local system's communications stacks.

[0003] A shortcoming with HTTP is that it does not provide for authoring through an HTTP channel. That is, the standard HTTP specifications do not specifically provide for clients to manage resources on servers. There is no way for a client to perform resource management operations like copying resources (e.g., files, documents, etc.), moving resources on a server, setting or obtaining properties of

resources on a server, locking resources, and so on. In response to this shortcoming, various public and private extensions to HTTP have been devised.

[0004] Figure 3 shows some method extensions 100 and header extensions 102 of a protocol or extension of HTTP that adds remote authoring functionality on top of HTTP. These extensions are from RFC 2518, which defines "HTTP Extensions for Web Authoring and Versioning", or "WebDAV". WebDAV is a superset of HTTP that is sometimes referred to as a protocol, and sometimes referred to as an extension of HTTP. The WebDAV protocol defines conventions, methods 100, and headers 102, for requests and responses that otherwise comply with HTTP. That is, WebDAV requests and responses follow the basic format of HTTP messages (e.g., message 50 in Figure 1). Technically, some of the verbs in the web authoring methods 100 are defined as valid HTTP verbs, however, their functionality is extended by WebDAV. For example, PUT is part of HTTP, but WebDAV extends its functionality to collections, directories, folders, etc. The same basic HTTP communication rules are used, the same line/field/body delimiters are used, the same error codes may be used, and base HTTP methods 104 and base HTTP headers can appear in WebDAV messages. For example an ordinary HTTP OPTIONS request may be answered by a WebDAV-compliant server with a response that has standard HTTP headers as well as one or more non-standard HTTP headers that indicate the availability of one or more HTTP extensions on the server. In general, this manner of extending HTTP allows servers and clients to handle both base HTTP communications as well as various extensions thereto, even if a remote system does not support an extension that is supported locally; unsupported headers and methods are usually ignored or handled gracefully.

[0005] The WebDAV extension to HTTP provides functionality to create, change and move documents on a remote server (typically a web server). WebDAV implementations are useful, among other things, for remotely authoring documents or resources served by a web server. WebDAV implementations can also be used for general access-anywhere web-based file storage. Many operating systems, such as Windows, Linux, and Mac OSX provide built-in client and server support for WebDAV,

thus allowing transparent use of files on a WebDAV server somewhat as if they were stored in a local directory.

[0006] The methods and headers of WebDAV are fully documented elsewhere, however, the main methods are: PUT - put a resource or collection on the server; DELETE - delete a resource or collection from the server; PROPFIND - retrieve properties (as XML) of a resource; PROPPATCH - change and delete properties of a resource; MKCOL - create collections or directories; COPY - copy a resource from one URI to another on the server; MOVE - move a resource from one URI to another on the server; LOCK - put a lock on a resource; UNLOCK - remove a lock from a resource. Some notable headers (field names) are: destination - specifies a URI as a destination resource for methods such as COPY and MOVE; Lock-Token - specifies a token that identifies a particular lock; and Timeout - specifies a duration of a lock.

[0007] It has not previously been recognized that there are certain inefficiencies and weaknesses built into WebDAV that can become significant in certain circumstances. Figure 4 shows a timeline for a sequence of related authoring requests. Suppose that a user of HTTP client 52 would like to get and lock a resource on HTTP server 54. The user will first direct the client 52 to get a particular resource. The client 52 will generate and transmit a GET request 120 to the server 54. The server 54 handles the GET request 120 and returns an appropriate response 122. The round trip time is the time between client 52's transmission of the GET request 120 and the receipt of response 122. As can be seen in Figure 4, much of the round trip time can be attributed to the time that it takes for the GET request 120 and the response 122 to traverse the network. If the user also needs to lock the resource obtained by GET request 120, another round of communication is needed: client 52 sends discrete LOCK request 124; LOCK request 124 passes through the network; and the server 54 replies with a response 126 that also crosses the network. The second exchange has its own round trip time that may include significant network transmission time. The total time 128 to meet two related needs of the client 52 (the need to both get and lock a resource) includes the time for two round trips or four network transmissions. Furthermore, the two discrete requests

120, 124 require approximately twice the server overhead as a single request, which might cause further delay if the server is heavily loaded.

[0008] Another problem with the example in Figure 4 is that the requested resource could be modified or locked by another client (or the server 54 itself) between the time that client 52 requests the resource and the time the client 52 is able to obtain a lock on the resource. In other words, another request can affect the resource after it is received by the client 52 but before the client 52 is able to obtain a lock on the resource, which could cause an error or unexpected result.

[0009] The atomic nature of WebDAV and the inability of WebDAV clients and servers to use compound or multi-aspect authoring requests with one discrete exchange may have other problems and inconveniences. Without necessity, some embodiments discussed below may alleviate some problems associated with HTTP authoring.

SUMMARY

[0010] The following summary is included only to introduce some concepts discussed in the Detailed Description below. This summary is not comprehensive and is not intended to delineate the scope of protectable subject matter, which is set forth by the claims presented at the end.

[0011] Certain client-server communication conventions extend compounded web authoring methods to a web authoring protocol such as WebDAV. More particularly, a web authoring request can be provided with special header information to signify a first web authoring method compounded with a second web authoring method indicated by a verb in the request. Clients and servers are provided with techniques to use the web authoring extensions. Extended error handling can be used to allow servers to provider richer web authoring error information to clients.

[0011a] According to one aspect of the present invention, there is provided a computer-readable storage medium having computer-executable instructions stored thereon for execution by a server computing device, the instructions, when executed, causing the server computing device to perform a process for servicing requests, the process comprising: handling, at the server computing device, standard HTTP get requests, standard HTTP post requests, and standard HTTP options requests; handling, at the server computing device, requests that conform to an HTTP authoring protocol, the requests comprising put requests, copy requests, move requests, propfind requests, and proppatch requests; sending an indicator to a client computing device indicating that the server computing device handles compounded authoring requests; receiving, at the server computing device, the compounded authoring requests, each of the compounded authoring requests including a special content type header value that identifies a request as one of the compounded authoring requests and a special extensions header including a compounded verb of the HTTP authoring protocol; handling, at the server computing device, the compounded authoring requests that do not conform to the HTTP authoring protocol, the compounded authoring requests comprising put+proppatch requests and post+propfind requests; and performing put functionality and proppatch functionality responsive to a put+proppatch request, or performing post functionality and propfind functionality responsive to a post+propfind request.

[0011b] According to another aspect of the present invention, there is provided a computing system for storing information to enable a device to perform a process for servicing requests, the system comprising: a processor; and a computer-readable medium storing instructions that, when executed by the processor, cause the processor to: handle, at a server computing device, standard HTTP get requests, standard HTTP post requests, and standard HTTP options requests; handle, at the server computing device, requests that conform to an HTTP authoring protocol, the requests comprising put requests, copy requests, move requests, propfind requests,

and proppatch requests; send an indicator to a client computing device indicating that the server computing device handles compounded authoring requests; receive, at the server computing device, the compounded authoring requests, each of the compounded authoring requests including a special content type header value that identifies a request as one of the compounded authoring requests and a special extensions header including a compounded verb of the HTTP authoring protocol; handle, at the server computing device, the compounded authoring requests that do not conform to the HTTP authoring protocol, the compounded authoring requests comprising put+proppatch requests and post+propfind requests; and perform put functionality and proppatch functionality responsive to a put+proppatch request, or performing post functionality and propfind functionality responsive to a post+profind request.

[0011c] According to still another aspect of the present invention, there is provided a method for servicing requests, the method comprising: handling, at a server computing device, standard HTTP get requests, standard HTTP post requests, and standard HTTP options requests; handling, at the server computing device, requests that conform to an HTTP authoring protocol, the requests comprising put requests, copy requests, move requests, propfind requests, and proppatch requests; sending an indicator to a client computing device indicating that the server computing device handles compounded authoring requests; receiving, at the server computing device, the compounded authoring requests, each of the compounded authoring requests including a special content type header value that identifies a request as one of the compounded authoring requests and a special extensions header including a compounded verb of the HTTP authoring protocol; handling, at the server computing device, the compounded authoring requests that do not conform to the HTTP authoring protocol, the compounded authoring requests comprising put+proppatch requests and post+propfind requests; and performing put functionality and proppatch functionality responsive to a put+proppatch request, or performing post functionality and propfind functionality responsive to a post+propfind request.

[0012] Many of the attendant features will be more readily appreciated by referring to the following detailed description considered in connection with the accompanying drawings.

DESCRIPTION OF THE DRAWINGS

[0013] Like reference numerals are used to designate like parts in the accompanying Drawings.

[0014] Figure 1 shows an HTTP message.

[0015] Figure 2 shows an example HTTP request and an example HTTP response.

[0016] Figure 3 shows some method extensions and header extensions of a protocol or extension of HTTP that adds remote authoring functionality on top of HTTP.

[0017] Figure 4 shows a timeline for a sequence of related authoring requests.

[0018] Figure 5 shows some web authoring method extensions and compounding header extensions that can be used to allow clients and servers to compound two or more authoring related methods with single client-server exchanges.

[0019] Figure 6 shows an uncompounded authoring exchange and a compounded authoring exchange with a similar purpose.

[0020] Figure 7 shows how a client can determine if compounding is available on a server.

[0021] Figure 8 shows how requests can be formatted to invoke compounded locking.

[0022] Figure 9 shows a mechanism for compounding property methods with HTTP or WebDAV methods or verbs.

[0023] Figure 10 shows further compounded methods.

[0024] Figure 11 shows an example POST+PROPFIND+LOCK method request and a corresponding response.

[0025] Figure 12 shows an error handling table and examples of a response using extended error handling.

DETAILED DESCRIPTION

[0026] Figure 5 shows some web authoring method extensions 140 and compounding header extensions 142 that can be used to allow clients and servers to compound two or more authoring related methods with single client-server exchanges. Although the method extensions 140 are characterized as "methods", they need not involve verbs or request lines 54 that are different than those defined by HTTP and WebDAV. However, conceptually, the method extensions 140 discussed below effectuate compound authoring methods. As discussed below, these in-effect compound authoring method extensions 140 can be accomplished using various compounded header extensions 142.

[0027] In the Figures, the symbols "+" and "|" (vertical bar) respectively represent compounding and "or". So, for example, the "POST|GET + LOCK|REFRESH|UNLOCK" method 144 represents a number of discrete compound methods: "POST+LOCK", "POST+UNLOCK", "GET+LOCK", etc. An explanation of how the method extensions 140 can be implemented using header extensions 142 will follow. Methods 144 and 146 will be discussed with reference to Figure 8. Methods 148 and 150 will be discussed with reference to Figure 9. Methods 152 and 154 will be discussed with reference to Figures 10 and 11.

[0028] Figure 6 shows an uncompounded authoring exchange and a compounded authoring exchange with a similar purpose. The left hand side of Figure 6 – a repetition of Figure 4 – shows the flow of an uncompounded authoring exchange. The right hand side of Figure 6 shows the flow of a compounded authoring exchange. On the right hand side of Figure 6 (the compounded example), a single request message 170 is sent by the client 52. The request message 170 includes information indicating a GET operation directed to a resource on server 174 and information indicating that the server 174 is to also lock the resource for the client 172. In the compounded case, there is one exchange with the time of one round trip. The total transaction time 174 for the compounded request 170 is less than the total transaction time 128 of the uncompounded requests 120, 124. Furthermore, because the server 174 can tell from the request message 170 that a

lock is desired, the server 174 can immediately lock the requested resource, thus preventing an intervening request from interfering with client 172's request.

[0029] Figure 7 shows how client 172 can determine if compounding is available on server 174. As mentioned earlier, it is desirable (but not necessary) for a client to be able to use both compounded and uncompounded authoring requests. It is also desirable for a server to be able to support both compounded and uncompounded authoring requests. A mechanism can be provided for this purpose. Preferably, the mechanism involves including information in a server response that indicates whether compounding is supported by the server. Although this information can take any form, the use of a new response header 190 is convenient because clients usually ignore unrecognized headers in an HTTP response. Furthermore, known algorithms for header parsing can be readily extended to process a new header or field name. In the alternative, a compounding indicator can take the form of a new value for a standard header, a special status line, etc.

[0030] To establish the availability of compounding, the client 172 performs a process 192 that starts with sending a standard OPTIONS request 194 (request 194 is only an example). A process 196 on the server 174 receives the OPTIONS request 194 and generates a response such as response 198 that includes a compounding indicator, in this embodiment, non-standard response header 190. The actual name of the non-standard response header 190 is not important other than it be known in advance by the client 172 so that when the client's 172 process 192 receives the response 198 it can recognize it and communicate with the server 174 as appropriate.

[0031] Figure 8 shows how requests can be formatted to invoke compounded locking. The top part corresponds to the extended methods 144 (see Figure 5) for GET or POST locking, and the lower part corresponds to the extended methods 146 for PUT locking. In one embodiment, ordinary HTTP and WebDAV request verbs 220 GET, POST, and PUT are compounded with locking requests using various combinations of a standard Lock-Token header 222 and a non-standard or extended

lock timeout header 224, for example "X-MSDAVEXTLock-Timeout". The lock timeout header 224 has a value of 0 or more seconds.

[0032] The lock timeout header 224 signifies the creation of a new lock according to the value of the lock timeout header 224. If the Lock-Token header 222 is included then the lock timeout header 224 signals the refresh of an existing lock. If the lock timeout header 224 is set to 0 seconds than an unlock is indicated (in this case, the Lock-Token header 222 and a correct token are required to unlock the file). Furthermore, a Lock-Token header 222 and token are preferably included in the response to any write operation on a locked resource. Example request 228 shows what a typical POST+UNLOCK request might look like. Note the inclusion of a Lock-Token header 222 and a lock timeout header 224.

[0033] Referring to the PUT verb combined with a locking operation, note that the Lock-token header 222 and correct token are needed to modify a locked resource. No token is needed if the resource is not locked. If no token is included but a lock time is specified, then the natural locking logic occurs; a lock is granted if no lock exists, and the PUT and lock are denied if a lock already exists. In sum, if the correct token is included with a PUT request the client can perform any PUT operation or any PUT operation combined with a lock operation. A typical PUT+REFRESH request is shown by request 230. The lock timeout value of 120 seconds indicates a refresh or resetting of the lifetime of the lock to run for another 120 seconds, and the lock token is the key that the server uses to authorize both the PUT operation and the REFRESH operation. In a preferred embodiment a Lock-Token header included in a non-write operation is ignored; i.e., "GET+verify an existing lock" is not supported.

[0034] Figure 9 shows a mechanism for compounding property methods with HTTP or WebDAV methods or verbs. These compound methods correspond to the methods 148, 150 in Figure 5. The compounded property methods use two indicators; a special content type header value 240 (e.g., "multipart/MSDAVEXTPrefixEncoded") and a special extensions header 242 with various possible values such as "PROPFIND" and "PROPPATCH". The combinations in

table 244 are self-explanatory and the resulting methods allow a resource to be accessed or modified while at the same time obtaining or setting one or more properties of the relevant resource. Furthermore, the standard Content-length header will be used and will give the value of the total message body or payload, which may also include properties as well resources (discussed further below).

[0035] In conformance with the rules of table 244, an example GET+PROPFIND request 246 is shown. Note the inclusion of an indication of the PROPFIND portion of the method in the form of the special extensions header 242 with the appropriate value or verb.

[0036] Although in one embodiment property related methods are compounded onto other methods using headers and a message body extension, other approaches may also be used. For example, the WebDAV PROPFIND and PROPPATCH methods could be overloaded using new headers. Furthermore, there are different ways for combining a resource and a set of properties in a message body. All of the properties can be put in separate headers, since most property sets are of manageable size. The properties could be assigned to respective different headers, although this would require more coding to handle transport of properties. In another embodiment, all of the properties (XML structure) can be placed in one large header, however, headers could potentially become larger than the buffers that some web servers allocate for header handling.

[0037] It is possible that some implementations may need to simultaneously set properties (PROPATCH) and get properties (PROPFIND) of a resource. For example, to determine whether a particular property was properly set, or to determine what a property was set to before it was changed with a PROPPATCH. In this case, "PROPPATCH" and "PROPFIND" can both be included, and a convention can be established for the location of sent and returned properties in the message body.

[0038] Although the WebDAV protocol does not specify particular properties for resources, some typical properties are analogous to properties of objects in a file system, for example content size, creation date, date of last modification, last

modifying user, special folder type, resource tag, file attributes, creation time, last access time, last modified time, and so on.

[0039] Figure 9 also shows an example GET+PROPFIND response 250. Note that the content type header field 240 signals the presence of a multi-part message body using the special "multipart/MSDAVEXTPrefixEncodedheader" extension. The lock related information is not required for the PUT+PROPPATCH method but may signify the presence of a server-side lock. The content type header field 240 signifies the presence of the multi-part message body 248 within a message body 64. Generally those multiple parts are divided by a length field followed by a corresponding data, in other words, the message body 64 carries one or more pieces of discrete data, each preceded by a corresponding length indicator. The sizes of the length fields and the sizes of their data add up to the standard Content-length header. The example response 250 in Figure 9 happens to have a properties section and a resource section, each preceded by a respective length field, for example, a 64 bit integer. Because compound authoring is designed as an HTTP extension, the standard HTTP message body 64 is used. Because properties may need to be exchanged in a message that may also include a resource such as an HTML document, the length-data pairings allow both properties and resources to be carried in a same message body 64. The standard Content-length header gives the total length of the message body 64/248 and can be used, in conjunction with the length indicators, to parse out the substantive pieces of content in the message body 248.

[0040] Referring back to the methods 152, 154 in Figure 5 (POST|GET + PROPFIND + LOCK|REFRESH|UNLOCK, PUT + PROPPATCH + LOCK|REFRESH|UNLOCK), these methods can be implemented by combining the properties and locking extensions discussed above. Because locking functionality and properties functionality is logically separate, the methods and headers discussed above can readily coexist in a message. Figure 10 shows further compounded methods. The top message is an example of a PUT+PROPPATCH+UNLOCK request 270. Note the size of the body, including the size of the length fields, is 114234. The bottom

message is an example of a corresponding response 272. A response that is successful need not differ from the response to a normal PUT request. The lack of a lock token header denotes a successful unlock. Figure 11 shows an example POST+PROPFIND+LOCK method request 290 and a corresponding response 292. The request 290 causes the server to put a resource or file, set some properties, and unlock the file or resource.

[0041] Figure 12 shows an error handling table and examples of a response 302 using extended error handling. A number of types of errors can occur when creating a resource on a server via an extended HTTP authoring request, for example, insufficient permissions, a resource checked out by another user or not checked at all, a quota violation, or a blocked filename or file type, the presence of a virus, etc. Other errors such as a missing property can occur when attempting to write to a file or resource. When an authoring error occurs on a server, typically the server may have rich system-level information about the error. Previously, when a module or server that implements WebDAV methods would encounters an error it would translate that error to a standard HTTP error code. A client might attempt to provide a useful message about the error code, perhaps using a corresponding hardcoded message string. However, the standard HTTP error codes are not rich enough to support the number and types of errors which users can encounter using extended HTTP authoring. Therefore, an extension is optionally provided to extend the error information fed back to a client, while keeping the existing HTTP error code, which allows for backward compatibility. This extended error handling is accomplished by including in responses information specific to system-level errors on the server.

[0042] As seen in Figure 12, extended error handling can be realized using a new HTTP header, for instance "X-MSDAVEXT_ERROR: Decimal; String". The decimal portion is a code that maps to a system-level error such as a Unix file control error or a Win32 error. Preferably, the String portion is in UTF-8 format.

[0043] Regarding compounding extensions of web authoring protocols in general, it should be noted that some proxy servers may attempt to interpret

requests and send back cached responses. Therefore, it is preferable that clients only use the new extensions or methods with POST rather than GET. Furthermore, when responding to a concatenated method or verb as discussed above, a server should mark a response to indicate that it should not be cached, using, for example, a header like "cache-control: private".

[0044] The server and client processes for using extended compound authoring methods are fairly straight forward given conventions as discussed above. Publicly available source code and documentation can be consulted to determine how to implement servers and clients with the functionality for performing atomic authoring methods and in particular locking and property functionality. This functionality can be performed in serial fashion when a compound method is encountered. For example, whereas previously a server may have had a function to handle a LOCK method and a function to handle a POST method, roughly, those functions can be invoked consecutively when a compounded POST+LOCK method is received.

[0045] Although HTTP and WebDAV have been discussed above, the ideas discussed above are expected to be applicable to any future variations or versions of HTTP and WebDAV. Furthermore, a standard protocol is considered to refer to any future or current standard protocol.

[0046] Further to aspects of extended error passing, there may be provided a volatile or non-volatile machine-readable media storing information to enable a device to perform a process for servicing requests from clients, the process including: handling standard HTTP get requests, standard HTTP post requests, and standard HTTP options requests and sending responses to corresponding clients; handling HTTP standard or non-standard authoring requests that direct the device to lock/unlock resources or direct the device to obtain or set properties of resources; and when errors occur handling the HTTP authoring requests, returning responses comprising error information that is not a HTTP status code. The error information can correspond to a system error of the device that caused the errors to occur. The error information can include an extended error header name and an accompanying

header field comprising identifying and/or describing a corresponding error, and furthermore, in such a case the header field may include a system error code of the device, or the header field can include a string specifically describing the error, or the header field can include a system error code of the device (the error code being or identifying a system error of the device), or the system error comprises a file locking error, or a file or directory read error, or a file or directory write error.

[0047] Further to another aspect of extended error passing, a volatile or non-volatile medium for storing digital data may be provided, the medium storing an HTTP response, the HTTP response including: a standard HTTP status code header and corresponding error code; and an indication of a server-side error, where the indication is not defined by a standard HTTP. The indication can include an HTTP header field specifically defined for conveying extended error information other than standard HTTP error codes, in which case the header field may include a field name not defined by a standard HTTP protocol and a field body that carries information about the server-side error, in which case it is further possible that the field body identifies a particular type of server-side error and that error does not correspond to a standard HTTP error code. The field body can include an operating system error code or a string describing an operating system error, and furthermore, the server-side error can comprise an operating system locking error, or an error reading or writing a server file or server directory.

[0048] In still another extended error handling embodiment, there may be provided a volatile or non-volatile storage for use with a processing device and storing information for enabling the processing device to perform a process, the process comprising: generating an HTTP request and sending the HTTP request to a server; receiving from the server an HTTP response to the HTTP request; and parsing the HTTP response for a non-standard extended error header and extracting from the non-standard extended error header information about an error on the server. The HTTP request can further include a standard HTTP status code header and corresponding error number. The information about the error on the system can comprise detail about a specific type of filesystem or operating system error on the

server. The information about the error on the server can comprise an operating system error number. The information about the error on the server can include a string describing an operating system error. The information about the error on the server may either identify or describe a specific system-level error of the server, further to which the HTTP requests may comprise an HTTP-based authoring request, either compounded or non-compounded, and the error on the server was an error performing a locking-related or properties-related method.

[0049] In conclusion, those skilled in the art will realize that storage devices utilized to store program instructions can be distributed across a network. For example a remote computer may store an example of the process described as software. A local or terminal computer may access the remote computer and download a part or all of the software to run the program. Alternatively the local computer may download pieces of the software as needed, or distributively process by executing some software instructions at the local terminal and some at the remote computer (or computer network). Those skilled in the art will also realize that by utilizing conventional techniques known to those skilled in the art, all or a portion of the software instructions may be carried out by a dedicated circuit, such as a DSP, programmable logic array, or the like.

[0050] All of the embodiments and features discussed above can be realized in the form of information stored in volatile or non-volatile computer or device readable medium. This is deemed to include at least media such as CD-ROM, magnetic media, flash ROM, etc., storing machine executable instructions, or source code, or any other information that can be used to enable a computing device to perform the various embodiments. This is also deemed to include at least volatile memory such as RAM storing information such as CPU instructions during execution of a program carrying out an embodiment.

CLAIMS:

1. A computer-readable storage medium having computer-executable instructions stored thereon for execution by a server computing device, the instructions, when executed, causing the server computing device to perform a process for servicing requests, the process comprising:

handling, at the server computing device, standard HTTP get requests, standard HTTP post requests, and standard HTTP options requests;

handling, at the server computing device, requests that conform to an HTTP authoring protocol, the requests comprising put requests, copy requests, move requests, propfind requests, and proppatch requests;

sending an indicator to a client computing device indicating that the server computing device handles compounded authoring requests;

receiving, at the server computing device, the compounded authoring requests, each of the compounded authoring requests including a special content type header value that identifies a request as one of the compounded authoring requests and a special extensions header including a compounded verb of the HTTP authoring protocol;

handling, at the server computing device, the compounded authoring requests that do not conform to the HTTP authoring protocol, the compounded authoring requests comprising put+proppatch requests and post+propfind requests; and

performing put functionality and proppatch functionality responsive to a put+proppatch request, or performing post functionality and propfind functionality responsive to a post+propfind request.

2. A computer-readable storage medium according to claim 1, the process further comprising performing get functionality and propfind functionality responsive to a get+propfind request.

3. A computer-readable storage medium according to claim 1, where the handled requests that do not conform to the HTTP authoring protocol further comprises requests that, in single requests, specify both a post, get, or put operation, and a lock, unlock, or lock refresh operation.

4. A computer-readable storage medium according to claim 1, wherein the process further comprises responding to an HTTP OPTIONS request by generating a response that includes header information indicating support for single requests with compounded authoring requests.

5. A computer-readable storage medium according to claim 1, wherein the compounded authoring requests comprise a method field conforming to the HTTP authoring protocol and a header name or header field not defined by the HTTP authoring protocol.

6. A computer-readable storage medium according to claim 1, the process further comprising including a set of properties and a resource in a body of a response message.

7. A computing system for storing information to enable a device to perform a process for servicing requests, the system comprising:

a processor; and

a computer-readable medium storing instructions that, when executed by the processor, cause the processor to:

handle, at a server computing device, standard HTTP get requests, standard HTTP post requests, and standard HTTP options requests;

handle, at the server computing device, requests that conform to an HTTP authoring protocol, the requests comprising put requests, copy requests, move requests, propfind requests, and proppatch requests;

send an indicator to a client computing device indicating that the server computing device handles compounded authoring requests;

receive, at the server computing device, the compounded authoring requests, each of the compounded authoring requests including a special content type header value that identifies a request as one of the compounded authoring requests and a special extensions header including a compounded verb of the HTTP authoring protocol;

handle, at the server computing device, the compounded authoring requests that do not conform to the HTTP authoring protocol, the compounded authoring requests comprising put+proppatch requests and post+propfind requests; and

perform put functionality and proppatch functionality responsive to a put+proppatch request, or performing post functionality and propfind functionality responsive to a post+profind request.

8. The computing system of claim 7, where the handled requests that do not conform to the HTTP authoring protocol further comprises requests that, in single requests, specify both a post, get, or put operation, and a lock, unlock, or lock refresh operation.

9. A method for servicing requests, the method comprising:

handling, at a server computing device, standard HTTP get requests, standard HTTP post requests, and standard HTTP options requests;

handling, at the server computing device, requests that conform to an HTTP authoring protocol, the requests comprising put requests, copy requests, move requests, propfind requests, and proppatch requests;

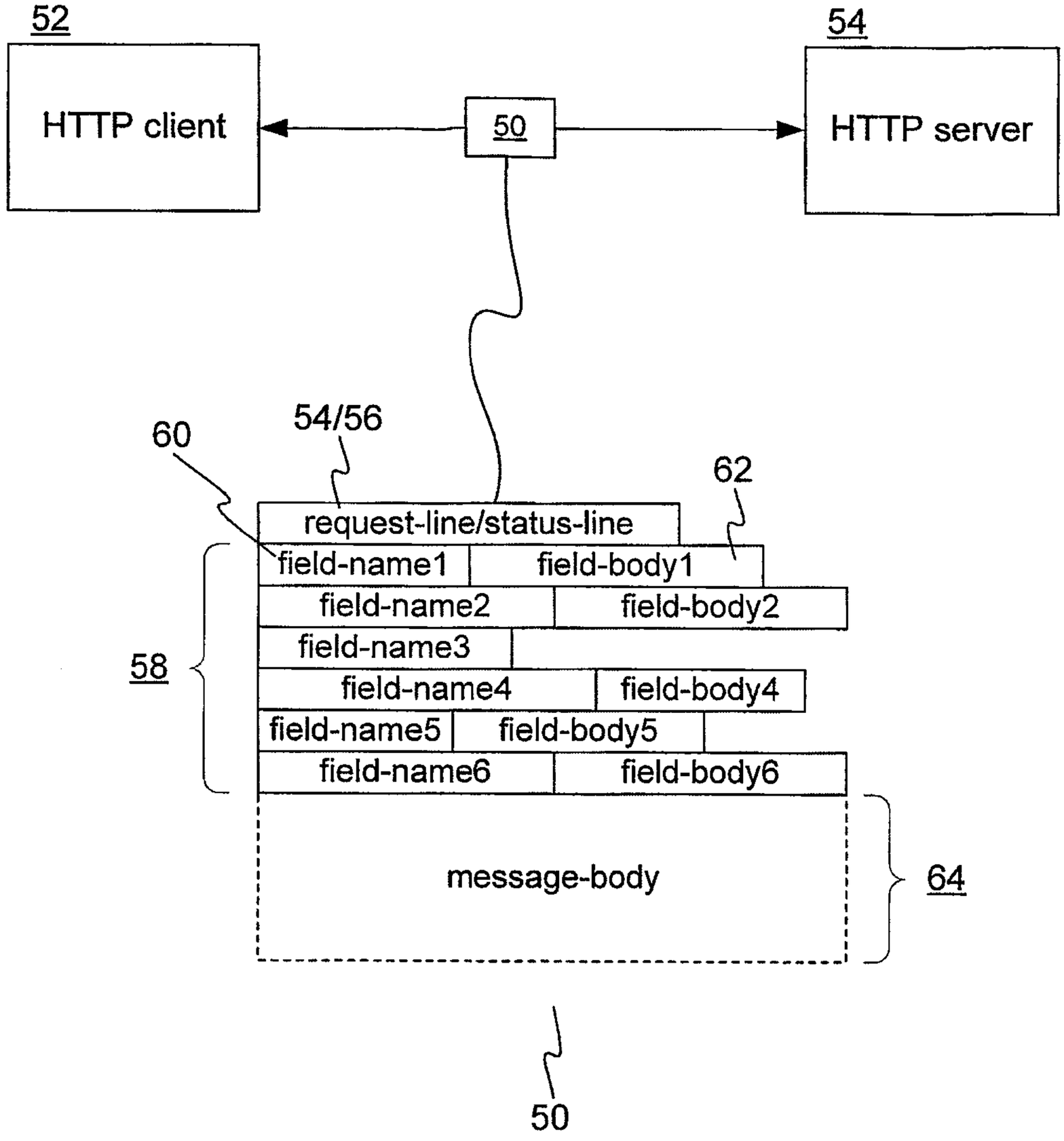
sending an indicator to a client computing device indicating that the server computing device handles compounded authoring requests;

receiving, at the server computing device, the compounded authoring requests, each of the compounded authoring requests including a special content type header value that identifies a request as one of the compounded authoring requests and a special extensions header including a compounded verb of the HTTP authoring protocol;

handling, at the server computing device, the compounded authoring requests that do not conform to the HTTP authoring protocol, the compounded authoring requests comprising put+proppatch requests and post+propfind requests; and

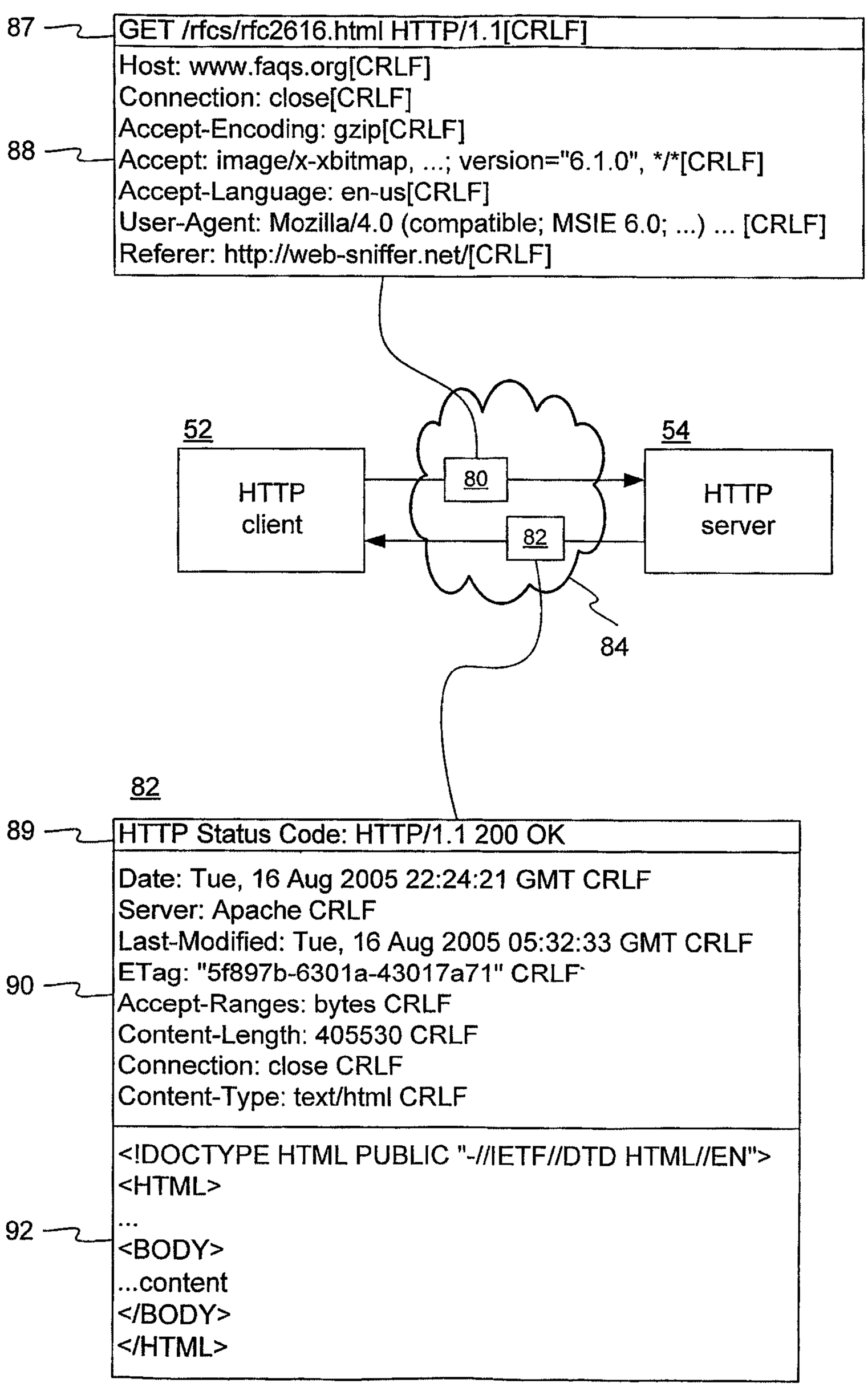
performing put functionality and proppatch functionality responsive to a put+proppatch request, or performing post functionality and propfind functionality responsive to a post+propfind request.

10. The method of claim 9, where the handled requests that do not conform to the HTTP authoring protocol further comprises requests that, in single requests, specify both a post, get, or put operation, and a lock, unlock, or lock refresh operation.



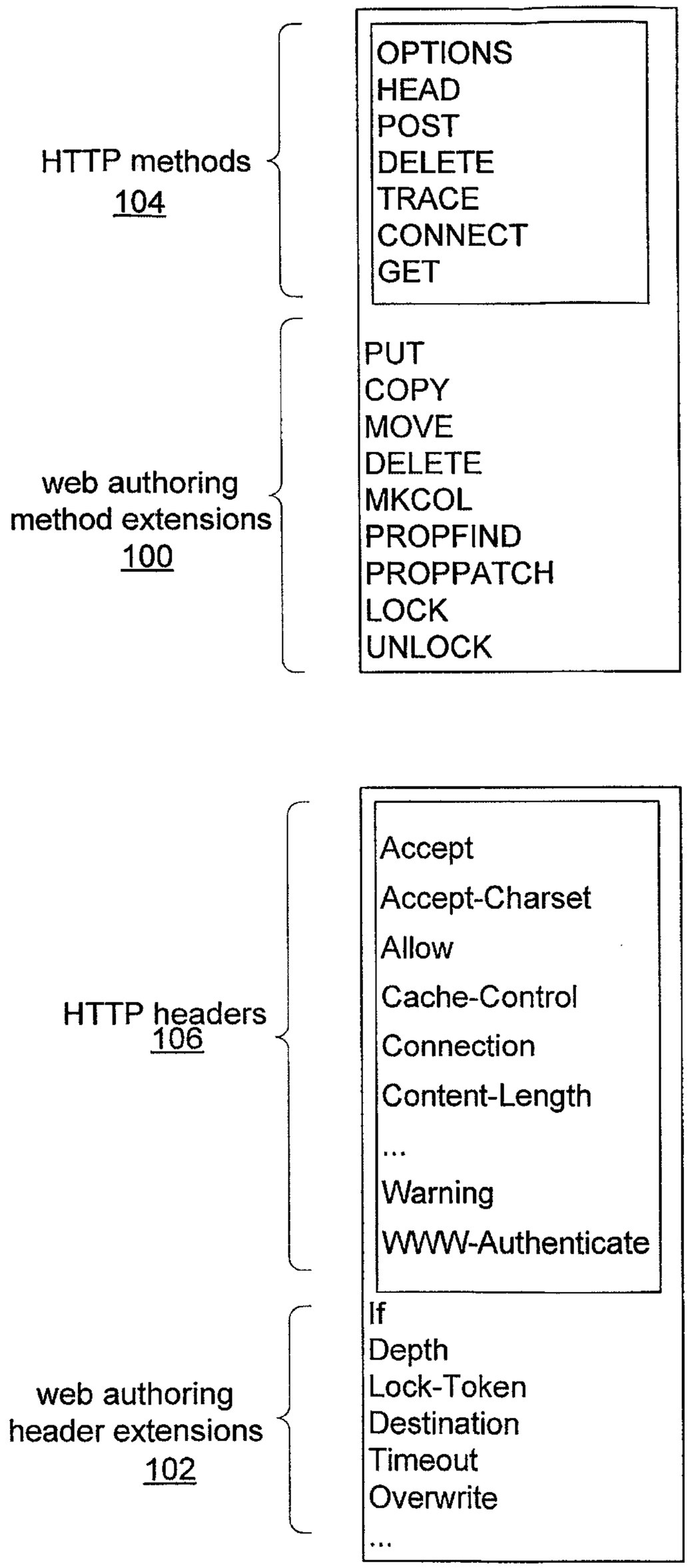
PRIOR ART

FIG. 1



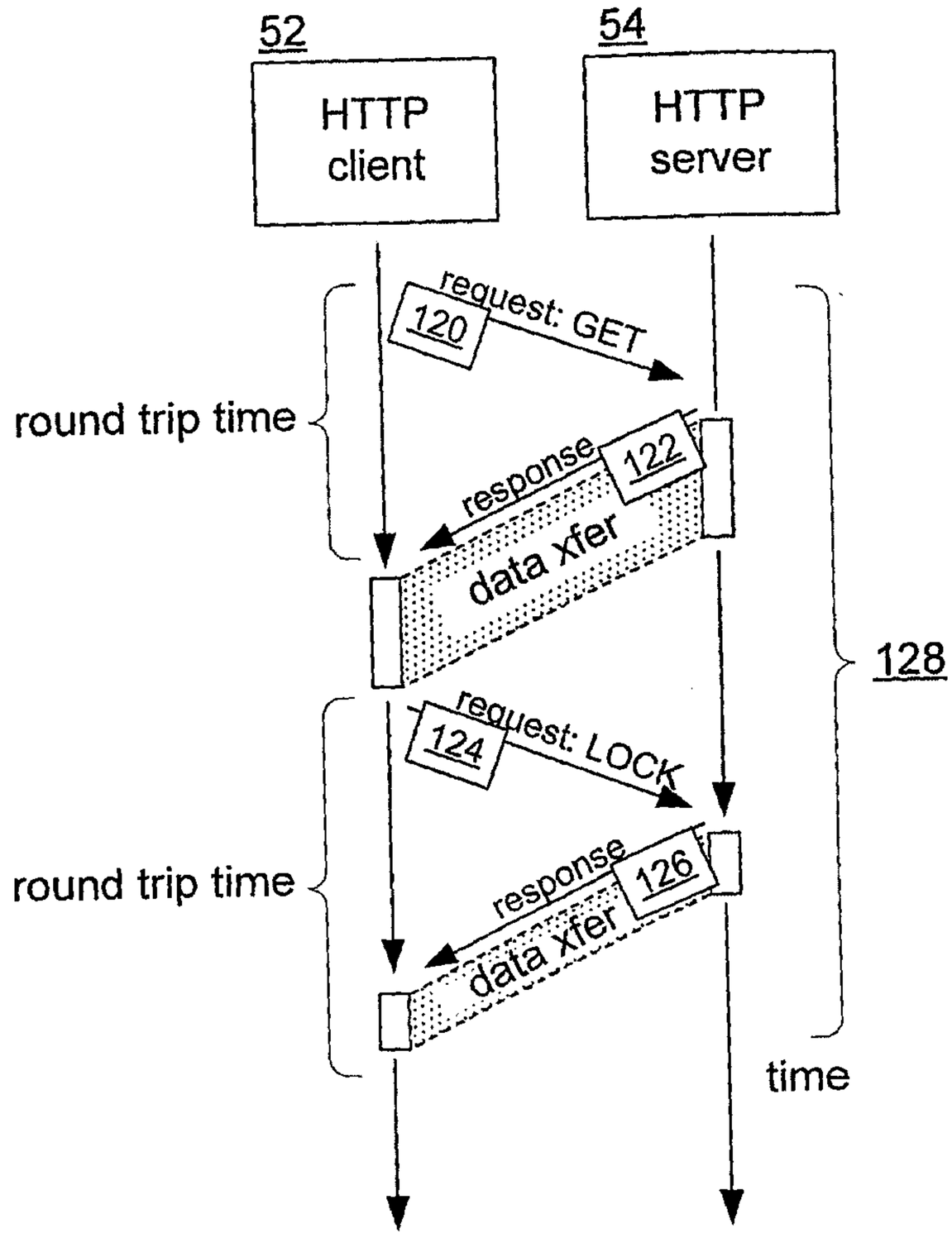
PRIOR ART

FIG. 2



PRIOR ART

FIG. 3



PRIOR ART

FIG. 4

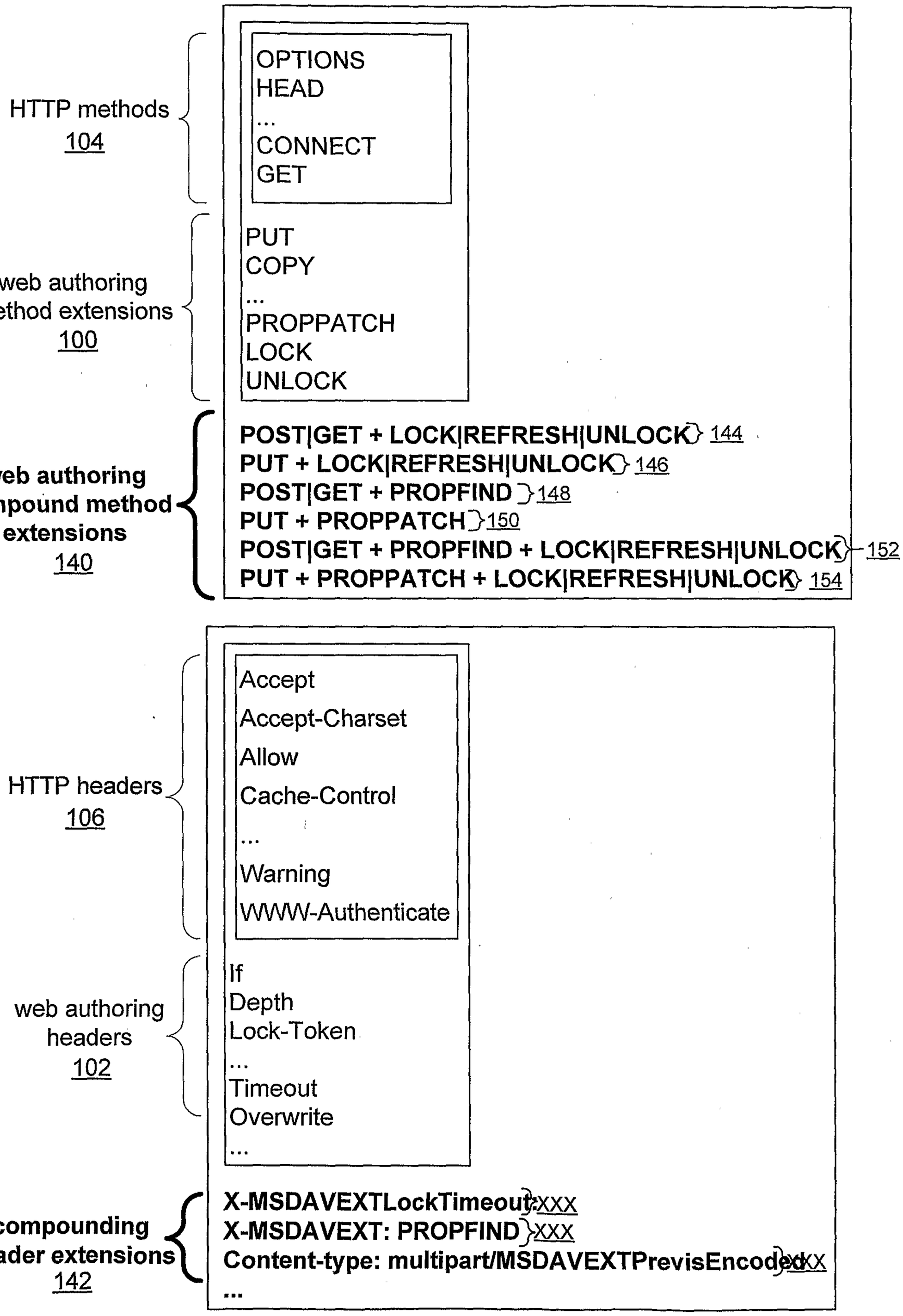

HTTP methods
104
OPTIONS
HEAD
...
CONNECT
GET
web authoring
method extensions
100
PUT
COPY
...
PROPPATCH
LOCK
UNLOCK
web authoring
compound method
extensions
140
POST|GET + LOCK|REFRESH|UNLOCK 144
PUT + LOCK|REFRESH|UNLOCK 146
POST|GET + PROPFIND 148
PUT + PROPPATCH 150
POST|GET + PROPFIND + LOCK|REFRESH|UNLOCK 152
PUT + PROPPATCH + LOCK|REFRESH|UNLOCK 154
HTTP headers
106
Accept
Accept-Charset
Allow
Cache-Control
...
Warning
WWW-Authenticate
web authoring
headers
102
If
Depth
Lock-Token
...
Timeout
Overwrite
...
compounding
header extensions
142
X-MSDAVEXTLockTimeout:XXX
X-MSDAVEXT: PROPFIND XXX
Content-type: multipart/MSDAVEXTPrevisEncoded XXX
...

FIG. 5

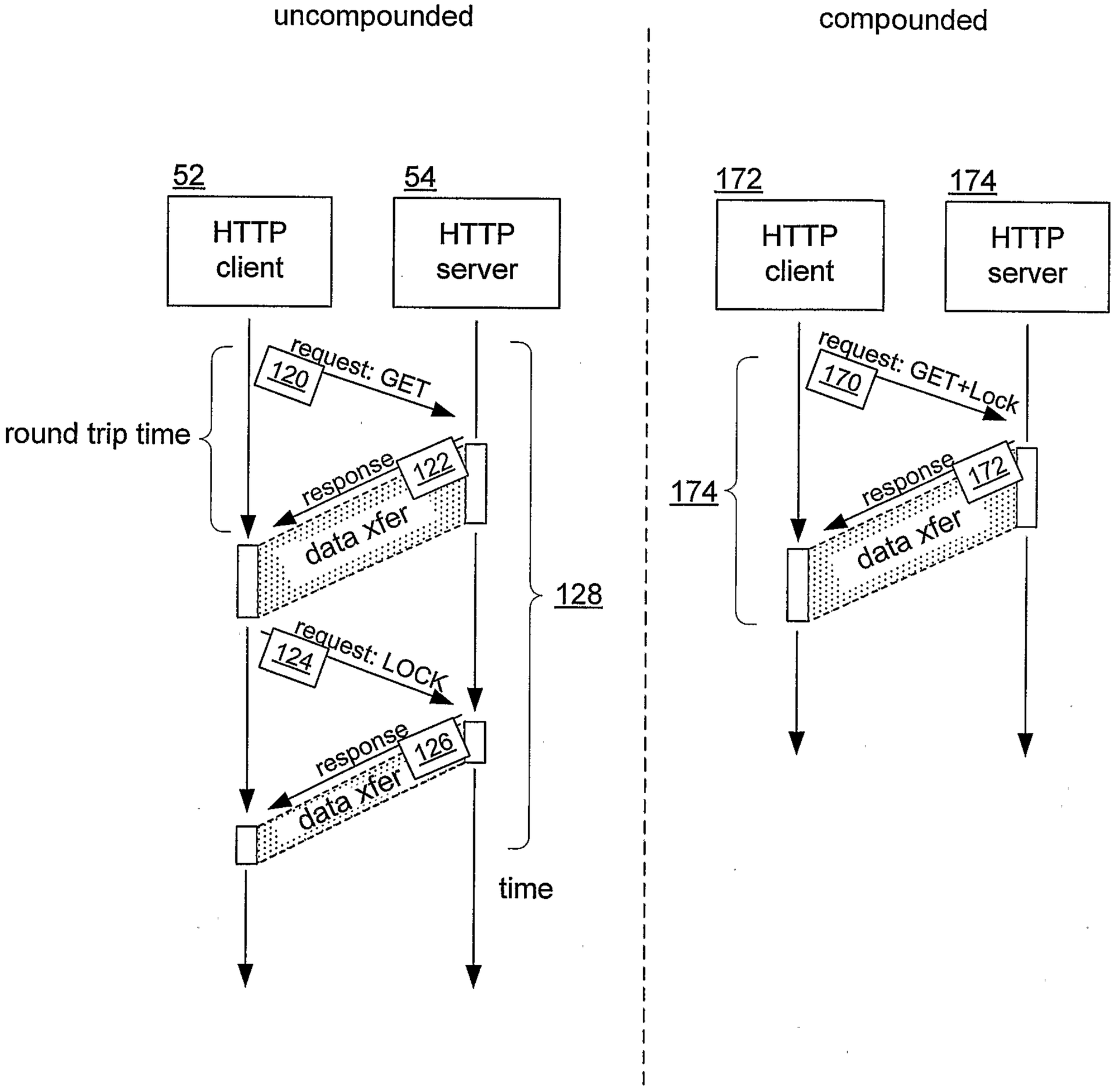

uncompounded
compounded
52
54
HTTP client
HTTP server
172
174
HTTP client
HTTP server
request: GET
120
round trip time
response
122
data xfer
128
request: LOCK
124
response
126
data xfer
time
request: GET+Lock
170
174
response
172
data xfer

FIG. 6

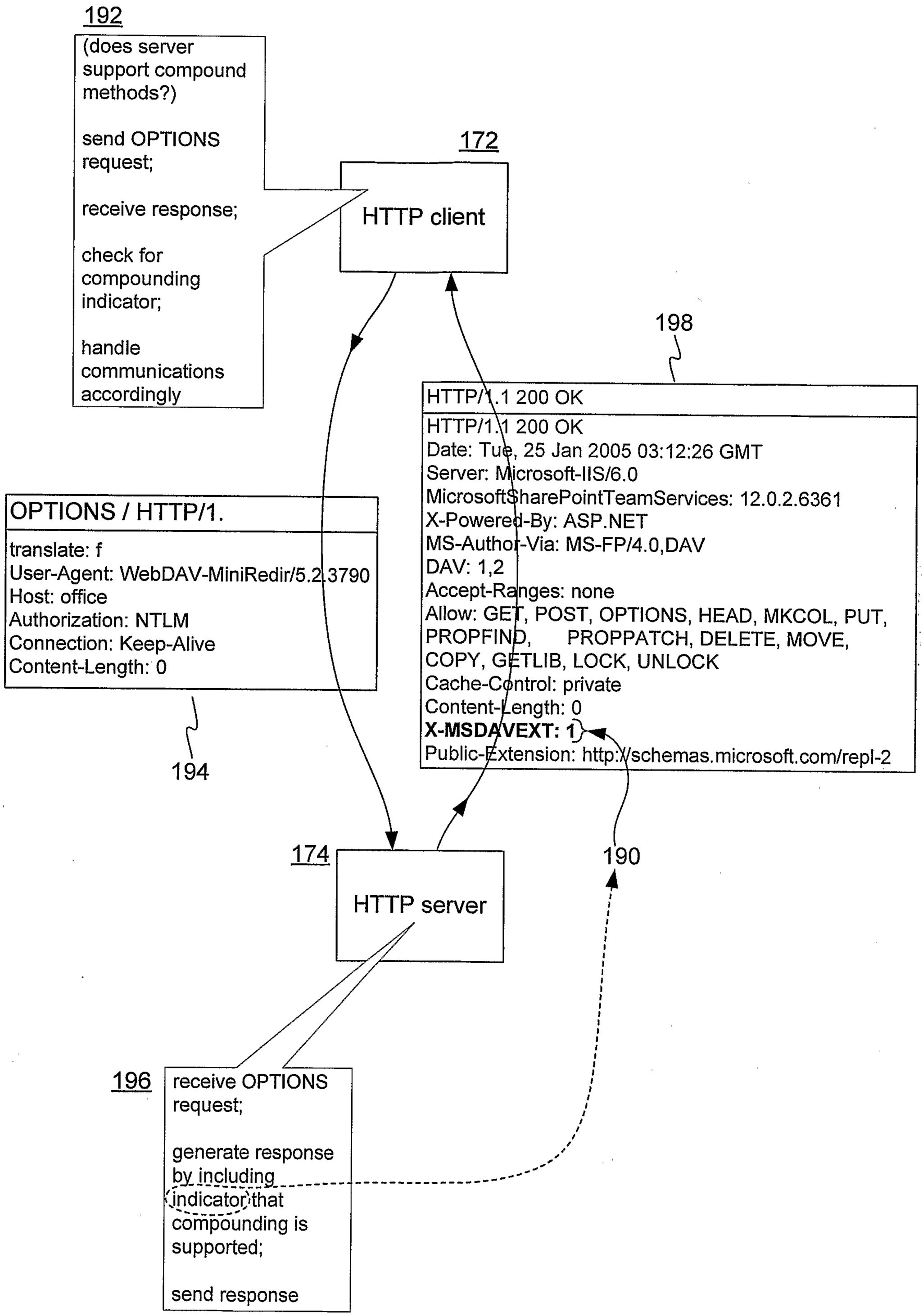

192
(does server support compound methods?)
send OPTIONS request;
receive response;
check for compounding indicator;
handle communications accordingly
172
HTTP client
198
HTTP/1.1 200 OK
HTTP/1.1 200 OK
Date: Tue, 25 Jan 2005 03:12:26 GMT
Server: Microsoft-IIS/6.0
MicrosoftSharePointTeamServices: 12.0.2.6361
X-Powered-By: ASP.NET
MS-Author-Via: MS-FP/4.0,DAV
DAV: 1,2
Accept-Ranges: none
Allow: GET, POST, OPTIONS, HEAD, MKCOL, PUT, PROPFIND, PROPPATCH, DELETE, MOVE, COPY, GETLIB, LOCK, UNLOCK
Cache-Control: private
Content-Length: 0
X-MSDAVEXT: 1
Public-Extension: http://schemas.microsoft.com/repl-2
OPTIONS / HTTP/1.
translate: f
User-Agent: WebDAV-MiniRedir/5.2.3790
Host: office
Authorization: NTLM
Connection: Keep-Alive
Content-Length: 0
194
174
HTTP server
190
196
receive OPTIONS request;
generate response by including indicator that compounding is supported;
send response

FIG. 7

example POST+UNLOCK request:

228

POST http://www.example.com/name_of_file.htm

...
Content-Type: application/x-www-form-urlencoded; charset=utf-8; ...
222 **Lock-Token: opaquelocktoken:{3932E32A-8825-494E-A19E-E714A7A741A8}20041130T183232Z**
224 **X-MSDAVEXTLockTimeout: second-0**
...

<content>

220 222 224 226

144

HTTP verb	headers: Lock-Token:	headers: X-MSDAVEXT LockTimeout:	result
GET|POST	Y	N	FAIL
GET|POST	Y	Y	FAIL - if token doesn't match LOCK - if no existing lock at server REFRESH - if X-MSDAVEXTLockTimeout ≠ 0 UNLOCK - if X-MSDAVEXTLockTimeout = 0
GET|POST	N	Y	sent lock token: FAIL - if file is already locked FAIL - if file is locked & X-...Lock-Timeout=0 didn't send lock token: LOCK - if no existing lock at server
GET|POST	N	N	return file

146

PUT	Y	N	success - if token matched FAIL - if token mismatch or file not locked
PUT	Y	Y	FAIL - if token doesn't match LOCK - if no existing lock at server REFRESH - if X-MSDAVEXTLock-Timeout ≠ 0 UNLOCK - if X-MSDAVEXTLock-Timeout = 0
PUT	N	Y	sent lock token: FAIL - if file is already locked FAIL - if file is locked & X-...Lock-Timeout=0 didn't send lock token: LOCK - if no existing lock at server
PUT	N	N	return file

example PUT+REFRESH request:

230

PUT http://www.example.com/somefile/name_of_file.htm

...
Content-Type: application/x-www-form-urlencoded; charset=utf-8; ...
224 **X-MSDAVEXTLockTimeout: second-120**
Lock-Token: opaquelocktoken:{3932E32A-8825-494E-A19E-E714A7A741A8}20041130T183232Z
222
...

<content>

FIG. 8

244

HTTP verb	header		result (effective method)
	Content-type: (240)	X-MSDAVEXT: (242)	
GET	multipart/MSDAVEXTPrefixEncoded	PROPFIND	GET+PROPFIND
POST	multipart/MSDAVEXTPrefixEncoded	PROPFIND	POST+PROPFIND
PUT	multipart/MSDAVEXTPrefixEncoded	PROPPATCH	PUT+PROPPATCH
PUT	multipart/MSDAVEXTPrefixEncoded	PROPFIND	PUT+PROPFIND

246

example GET+PROPFIND request:

```
POST /shared%20documents//Copy%20of%20Folder/test.rtf HTTP/1.1
translate: f
User-Agent: Microsoft-WebDAV-MiniRedir/5.2.3790
Host: office
Connection: Keep-Alive
Pragma: no-cache
X-MSDAVEXT: PROPFIND
Content-type: ...
...
```

242

250

example GET+PROPFIND response:

```
HTTP/1.1 200 OK
Date: Tue, 25 Jan 2005 03:12:31 GMT
Server: Microsoft-IIS/6.0
MicrosoftSharePointTeamServices: 6.0.2.6361
X-Powered-By: ASP.NET
Last-Modified: Thu, 12 Feb 2004 02:33:01 GMT
ETag: "{3E94207C-8E12-4491-B0FF-A903E2C9610E},7"
ResourceTag: rt:3E94207C-8E12-4491-B0FF-A903E2C9610E@00000000007
Content-type: multipart/MSDAVEXTPrefixEncoded ; ...
Cache-Control: private
X-MSDAVEXTLockTimeout: Second-####
Lock-Token: opaquelocktoken:{4A7A741A8}20041130T183232Z
Content-Length: #####
Public-Extension: http://schemas.microsoft.com/repl-2
```

240

(some number of seconds)

(size of body 248)

248

```
<length of properties section>
<properties>
<length of resource>
<resource>
```

FIG. 9

270 example PUT+PROPPATCH+UNLOCK request:

PUT /shared%20documents/Copy%20of%20Folder/test.rtf HTTP/1.1
... translate: f User-Agent: Microsoft-WebDAV-MiniRedir/5.2.3790 Host: dustinfrserver Content-Length: 114234 Connection: Keep-Alive X-MSDAVEXT: PROPPATCH X-MSDAVEXTLockTimeout: Second-0 Lock-Token: opaquelocktoken:{4A7A741A8}20041130T183232Z Content-type: multipart/MSDAVEXTPrefixEncoded Pragma: no-cache Authorization: NTLM ...
<length of properties section><some pushed properties> <length of resource><some PUT resource subject to the pushed properties>

272 example PUT+PROPPATCH+UNLOCK response:

HTTP/1.1 200 OK
HTTP/1.1 200 OK Date: Tue, 30 Nov 2004 18:32:34 GMT Server: Microsoft-IIS/6.0 X-Powered-By: ASP.NET MicrosoftSharePointTeamServices: 6.0.2.5530 Cache-Control: private Content-Length: 0 Public-Extension: http://schemas.microsoft.com/repl-2

FIG. 10

290 example POST+PROPFIND+LOCK request:

POST /shared%20documents/Copy%20of%20Folder/test.rtf HTTP/1.1
translate: f User-Agent: Microsoft-WebDAV-MiniRedir/5.2.3790 Host: dustinfrserver Connection: Keep-Alive X-MSDAVEXT: PROPFIND X-MSDAVEXTLockTimeout: Second-60 Content-Length: 0 Pragma: no-cache Authorization: NTLM ...

292 example POST+PROPFIND+LOCK response:

HTTP/1.1 200 OK
Date: Tue, 30 Nov 2004 18:32:34 GMT Server: Microsoft-IIS/6.0 X-Powered-By: ASP.NET MicrosoftSharePointTeamServices: 6.0.2.5530 Cache-Control: private Content-Length: 1056 Content-type: **multipart/MSDAVEXTPrefixEncoded ; ...** X-MSDAVEXTLockTimeout: Second-60 Lock-Token: opaquelocktoken:{4A7A741A8}20041130T183232Z Public-Extension: http://schemas.microsoft.com/repl-2 ...
00000000000001F4 <500 bytes of some returned properties> 000000000000020C <524 bytes of some posted resource>

FIG. 11

300

Extended Error code (Decimal)	an error code which the client can map to a system-level error code
String (URL Encoded)	a string with extended information for clients to use

302

HTTP/1.1 401 Unauthorized
Content-Length: 1656 Content-Type: text/html X-MSDAVEXT_ERROR: 2342; The file is checked out to "Redmond\ dustinfr" Server: Microsoft-IIS/6.0 WWW-Authenticate: NTLM X-Powered-By: ASP.NET MicrosoftSharePointTeamServices: 12.0.0.000 Date: Tue, 25 Jan 2005 03:11:51 GMT ...

FIG. 12

HTTP methods
104

OPTIONS
HEAD
...
CONNECT
GET

web authoring
method extensions
100

PUT
COPY
...
PROPPATCH
LOCK
UNLOCK

web authoring compound method extensions
140

POST|GET + LOCK|REFRESH|UNLOCK 144
PUT + LOCK|REFRESH|UNLOCK 146
POST|GET + PROPFIND 148
PUT + PROPPATCH 150
POST|GET + PROPFIND + LOCK|REFRESH|UNLOCK 152
PUT + PROPPATCH + LOCK|REFRESH|UNLOCK 154

HTTP headers
106

Accept
Accept-Charset
Allow
Cache-Control
...
Warning
WWW-Authenticate

web authoring
headers
102

If
Depth
Lock-Token
...
Timeout
Overwrite
...

compounding header extensions
142

X-MSDAVEXTLockTimeout:XXX
X-MSDAVEXT: PROPFIND}XXX
Content-type: multipart/MSDAVEXTPrevisEncoded}XXX
...