República Federativa do Brasil
Ministério do Desenvolvimento, Indústria e do Comércio Exterior
Instituto Nacional da Propriedade Industrial.

(21) **PI0614275-3 A2**

BRPI0614275A2

(22) Data de Depósito: 15/08/2006
(43) Data da Publicação: 22/03/2011
(RPI 2098)

(51) Int.Cl.:
H04L 12/66
H04L 29/06
G06F 17/00

(54) Título: **COMPOSIÇÃO DE PROTOCOLO DE AUTORIA HTTP**

(30) Prioridade Unionista: 31/08/2005 US 11/217,626

(73) Titular(es): MICROSOFT CORPORATION

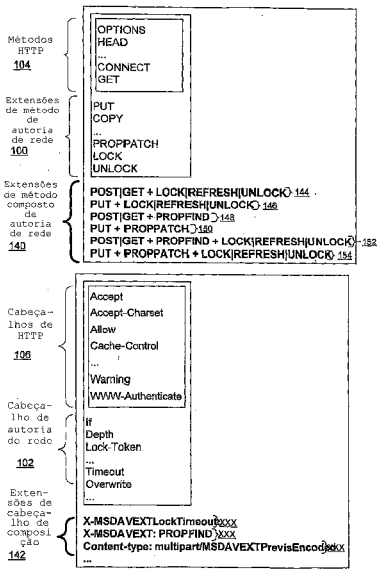
(72) Inventor(es): AHMED H. MOHAMED, ANDREW SEAN WATSON, DAVID M. KRUSE, DUSTIN G. FRIESENHAHN, JAY PAULUS, SEAN MCATTER, SUNDAR SUBBARAYAN, V. R. KISHORE CHINTALAPATI

(74) Procurador(es): Nellie Anne Daniel Shores

(86) Pedido Internacional: PCT US2006031705 de 15/08/2006

(87) Publicação Internacional: WO 2007/027426de 08/03/2007

(57) Resumo: COMPOSIÇÃO DE PROTOCOLO DE AUTORIA HTTP. Convenções para estender métodos de autoria de rede composta para um protocolo de autoria de rede como WebDAV. Mais particularmente, uma solicitação pode ser fornecida com informações especiais de cabeçalho para significar um método composto com um método indicado por um verbo na solicitação. Técnicas para clientes e servidores utilizarem as extensões de autoria de rede. Tratamento de erro estendido para permitir que os servidores forneçam informações mais ricas de erro de autoria da rede para clientes.



"COMPOSIÇÃO DE PROTOCOLO DE AUTORIA HTTP"

ANTECEDENTES

A figura 1 mostra uma mensagem HTTP 50. A troca de mensagens HTTP 50 entre um cliente de HTTP 52 e um servidor de HTTP 54 é bem conhecida na técnica de computação de cliente-servidor. Vários RFCs e outros documentos públicos podem ser consultados para detalhes sobre as diversas versões e variações de HTTP. Por exemplo,RFC 2616 define HTTP versão 1.1. De acordo com RFC 2616, uma mensagem HTTP 50 que é para uma solicitação de HTTP tem uma linha de solicitação 54, como "GET / HTTP/1.1". Uma mensagem de HTTP 50 que é para uma resposta de HTTP em vez disso tem uma linha de estado 56, como "HTTP/1.1 200 OK". Uma linha de solicitação 54 ou linha de estado 56 é normalmente seguida por um ou mais cabeçalhos, cada um consistindo em um nome de campo 60 e, dependendo do cabeçalho específico, zero ou mais corpos de campo 62. Uma mensagem 50 pode terminar com um corpo de mensagem 64, dependendo do tipo de solicitação ou resposta. Os detalhes referentes a delimitadores, cabeçalhos específicos, e outras características de mensagens de HTTP e comunicação de HTTP podem ser encontrados em outra parte.

A figura 2 mostra um exemplo de solicitação de HTTP 80 e um exemplo de resposta de HTTP 82. O cliente de HTTP 52 envia a solicitação 80 através de uma rede de dados 84 para o servidor de HTTP 54, que trata da solicitação e retorna a resposta 82. A solicitação 80 inclui uma linha de solicitação 87 e diversos cabeçalhos 88 (algumas solicitações têm, também, um corpo de mensagem). A resposta

82 inclui uma linha de estado 89, cabeçalhos 90, e um corpo de mensagem 92. Comunicações de HTTP não necessitam se deslocar através de uma rede como a rede 84; a comunicação entre um cliente local e um servidor local é possível, embora normalmente através das pilhas de comunicações do sistema local.

Uma desvantagem com HTTP é que não fornece de autoria através de um canal de HTTP. Isto é, as especificações de HTTP padrão não prevêem especificamente para os clientes gerenciarem recursos em servidores. Não há modo para um cliente executar operações de gerenciamento de recursos como copiar recursos (por exemplo, arquivos, documentos, etc.), mover recursos em um servidor, definir ou obter propriedades de recursos em um servidor, travar recursos, e assim por diante. Em resposta a essa desvantagem, várias extensões públicas e privadas para HTTP foram elaboradas.

A figura 3 mostra algumas extensões de método 100 e extensões de cabeçalho 102 de um protocolo ou extensão de HTTP que acrescentam funcionalidade de autoria remota no topo de HTTP. Essas extensões são de RFC 2518, que define "HTTP Extensions for web de autoria and versioning", ou "WebDAV". WebDAV é um superconjunto de HTTP que é às vezes mencionado como protocolo, e às vezes mencionado como uma extensão de HTTP. O protocolo de WebDAV define convenções, métodos 100, e cabeçalhos 102, para solicitações e respostas que de outro modo estão em conformidade com HTTP. Isto é, as solicitações e respostas de WebDAV seguem o formato

básico de mensagens HTTP (por exemplo, mensagem 50 na figura 1). Tecnicamente, alguns dos verbos nos métodos de autoria de rede 100 são definidos como verbos HTTP válidos, entretanto, sua funcionalidade é estendida por WebDAV. Por exemplo, PUT faz parte de HTTP, porém WebDAV estende sua funcionalidade para coleções, diretórios, pastas, etc. As mesmas regras básicas de comunicação de HTTP são utilizadas, os mesmos delimitadores de linha/campo/corpo são utilizados, os mesmos códigos de erro podem ser utilizados, e métodos de HTTP de base 104 e cabeçalhos de HTTP de base podem aparecer em mensagens de WebDAV. Por exemplo, uma solicitação HTTP OPTIONS comum pode ser respondida por um servidor em conformidade com WebDAV com uma resposta que tem cabeçalhos HTTP padrão bem como um ou mais cabeçalhos HTTP não padrão que indicam a disponibilidade de uma ou mais extensões de HTTP no servidor. Em geral, esse modo de estender HTTP permite que servidores e clientes manipulem tanto comunicações de HTTP de base bem como várias extensões das mesmas, mesmo se um sistema remoto não suportar uma extensão que é suportada localmente; métodos e cabeçalhos não suportados são normalmente ignorados ou manipulados de forma graciosa.

A extensão de WebDAV para HTTP provê funcionalidade para criar, alterar e mover documentos em um servidor remoto (tipicamente um servidor de rede). As implementações de WebDAV são úteis, entre outras coisas, para de autoria remotamente documentos ou recursos servidos por um servidor de rede. Implementações de WebDAV também

podem ser utilizadas para armazenagem de arquivo baseado em rede em qualquer lugar de acesso geral. Muitos sistemas operacionais, como Windows, Linux, e Mac OSX fornecem suporte de servidor e cliente incorporado para WebDAV, desse modo permitindo uso transparente de arquivos em um servidor WebDAV de um certo modo como se eles fossem armazenados em um diretório local.

Os métodos e cabeçalhos de WebDAV são totalmente documentos em outro lugar, entretanto, os principais métodos são: PUT - colocar um recurso ou coleção no servidor; DELETE - deletar um recurso ou coleção a partir do servidor; PROPFIND - recuperar propriedades (como XML) de um recurso; PROPPATCH - alterar e deletar propriedades de um recurso; MKCOL - criar coleções ou diretórios; COPY - copiar um recurso a partir de URI para outro no servidor; MOVE - mover um recurso a partir de um URI para outro no servidor; LOCK - colocar um travamento em um recurso; UNLOCK - remover um travamento de um recurso. Alguns cabeçalhos notáveis (nomes de campo) são: destino - especifica um URI como um recurso de destino para métodos como COPY e MOVE; Lock-token - especifica um token que identifica um travamento específico; e Intervalo - especifica uma duração de um travamento.

Não foi previamente reconhecido que há certas deficiências e fraquezas embutidas em WebDAV que podem se tornar significativas em certas circunstâncias. A figura 4 mostra uma linha de tempo para uma seqüência de solicitações de autoria relacionadas. Suponha que um usuário de cliente HTTP 52 gostaria de obter e travar um recurso no servidor

HTTP 54. O usuário primeiramente orientará o cliente 52 a obter um recurso específico. O cliente 52 gerará e transmitirá uma solicitação GET 120 para o servidor 54. O servidor 54 trata da solicitação GET 120 e retorna uma resposta apropriada 122. O tempo de ida e volta é o tempo entre a transmissão do cliente 52 da solicitação GET 120 e o recebimento da resposta 122. Como pode ser visto na figura 4, grande parte do tempo de ida e volta pode ser atribuído ao tempo que leva para que a solicitação GET 120 e resposta 122 atravessem a rede. Se o usuário também necessitar de travar o recurso obtido pela solicitação GET 120, outro round de comunicação é necessário: o cliente 52 envia solicitação LOCK discreta 124; a solicitação LOCK 124 passa através da rede; e o servidor 54 responde com uma resposta 126 que também cruza a rede. A segunda troca tem seu próprio tempo de ida e volta que pode incluir tempo significativo de transmissão de rede. O tempo total 128 para atender duas necessidades relacionadas do cliente 52 (a necessidade tanto de obter como travamento um recurso) inclui o tempo para duas idas e voltas ou quatro transmissões de rede. Além disso, as duas solicitações discretas 120, 124 exigem aproximadamente duas vezes o overhead de servidor como uma solicitação única, que poderia causar retardo adicional se o servidor estiver intensamente carregado.

Outro problema com o exemplo na figura 4 é que o recurso solicitado poderia ser modificado ou travado por outro cliente (ou o próprio servidor 54) entre o tempo que o cliente 52 solicita o recurso e o tempo que o cliente 52 é

capaz de obter um travamento no recurso. Em outras palavras, outra solicitação pode afetar o recurso após ser recebido pelo cliente 52 porém antes do cliente 52 ser capaz de obter um travamento no recurso, o que poderia causar um erro ou resultado inesperado.

A natureza atômica de WebDAV e a incapacidade dos clientes e servidores de WebDAV de utilizarem solicitações de autoria de multi-aspectos ou compostas com uma troca discreta podem ter outros problemas e inconveniências. Sem necessidade, algumas modalidades discutidas abaixo podem aliviar alguns problemas associados à autoria HTTP.

SUMÁRIO

O sumário a seguir está incluído somente para introduzir alguns conceitos discutidos na Descrição detalhada abaixo. Esse sumário não é abrangente e não pretende delinear o escopo de matéria de proteção, que é exposta pelas reivindicações apresentadas no final.

Certas convenções de comunicação de cliente-servidor estendem métodos de autoria de rede composta a um protocolo de autoria de rede como WebDAV. Mais particularmente, uma solicitação de autoria de rede pode ser dotada de informações de cabeçalho especiais para significar um primeiro método de autoria de rede composto com um segundo método de autoria de rede indicado por um verbo na solicitação. Clientes e servidores são dotados de técnicas para utilizar as extensões de autoria de rede. O tratamento de erro estendido pode ser utilizado para permitir que os servidores forneçam informações de erro de autoria de rede

mais ricas, para os clientes.

Muitas das características inerentes serão mais prontamente reconhecidas ao se referir à descrição detalhada a seguir considerada em conexão com os desenhos em anexo.

DESCRIÇÃO DOS DESENHOS

Numerais de referência similares são utilizados para designar partes similares nos Desenhos em anexo.

A figura 1 mostra uma mensagem de HTTP.

A figura 2 mostra um exemplo de solicitação HTTP e um exemplo de resposta HTTP.

A figura 3 mostra algumas extensões de método e extensões de cabeçalho de um protocolo ou extensão de HTTP que acrescenta funcionalidade de autoria remota no topo de HTTP.

A figura 4 mostra uma linha de tempo para uma seqüência de solicitações de autoria relacionadas.

A figura 5 mostra algumas extensões de método de autoria de rede e extensões de cabeçalho de composição que podem ser utilizadas para permitir que clientes e servidores componham dois ou mais métodos relacionados à autoria com trocas de servidor-cliente único.

A figura 6 mostra uma troca de autoria não composta e uma troca de autoria composta com uma finalidade similar.

A figura 7 mostra como um cliente pode determinar se a composição está disponível em um servidor.

A figura 8 mostra como as solicitações podem ser formadas para invocar travamento composto.

A figura 9 mostra um mecanismo para compor métodos de propriedade com verbos ou métodos de WebDAV ou HTTP.

A figura 10 mostra métodos compostos adicionais.

A figura 11 mostra um exemplo de solicitação de método POST+PROPFIND+LOCK e uma resposta correspondente.

A figura 12 mostra uma tabela de tratamento de erro e exemplos de uma resposta utilizando tratamento de erro estendido.

DESCRIÇÃO DETALHADA

A figura 5 mostra algumas extensões de método de autoria de rede 140 e extensões de cabeçalho de composição 142 que podem ser utilizadas para permitir que clientes e servidores componham dois ou mais métodos relacionados à autoria com trocas de servidor-cliente único. Embora as extensões de método 140 sejam caracterizadas como "métodos", não necessitam envolver verbos ou linhas de solicitação 54 que são diferentes daquelas definidas por HTTP e WebDAV. Entretanto, de forma conceptual, as extensões de método 140 discutidas abaixo efetuam métodos de autoria composto. Como discutido abaixo, essas extensões de método de autoria composto em vigor 140 podem ser realizadas utilizando várias extensões de cabeçalho compostas 142.

Nas figuras, os símbolos "+" e "|" (barra vertical" representam, respectivamente composição e "ou". Assim, por exemplo, o método "POST|GET + LOCK|REFRESH|UNLOCK" 144 representa um número de métodos de composto discreto: "POST+LOCK", "POST+UNLOCK", "GET+LOCK", etc. Uma explicação sobre as extensões do método 140 podem

ser implementadas utilizando extensões de cabeçalho 142 será feita a seguir. Os métodos 144 e 146 serão discutidos com referência à figura 8. Os métodos 148 e 150 serão discutidos com referência à figura 9. Os métodos 152 e 154 serão discutidos com referência às figuras 10 e 11.

A figura 6 mostra uma troca de autoria não composta e uma troca de autoria composto com uma finalidade similar. O lado esquerdo da figura 6 - uma repetição da figura 4 - mostra o fluxo de uma troca de autoria não composta. O lado direito da figura 6 mostra o fluxo de uma troca de autoria composta. No lado direito da figura 6 (o exemplo composto), uma única mensagem de solicitação 170 é enviada pelo cliente 52. A mensagem de solicitação 170 inclui informações indicando uma operação GET dirigida a um recurso no servidor 174 e informações indicando que o servidor 174 também deve travar o recurso para o cliente 172. No caso composto, há uma troca com o tempo de ida e volta. O tempo total de transação 174 para a solicitação composta 170 é menor do que o tempo total de transação 128 das solicitações não compostas 120, 124. Além disso, como o servidor 174 pode dizer a partir da mensagem de solicitação 170 que uma trava é desejável, o servidor 174 pode travar imediatamente o recurso solicitado, desse modo evitando que uma solicitação intermediária interfira na solicitação do cliente 172.

A figura 7 mostra como o cliente 172 pode determinar se a composição está disponível no servidor 174. Como mencionado anteriormente, é desejável (porém não

necessário) que um cliente seja capaz de utilizar solicitações de autoria tanto compostas como não compostas. Também é desejável que um servidor seja capaz de suportar solicitações de autoria tanto compostas como não compostas. Um mecanismo pode ser fornecido para essa finalidade. Preferivelmente, o mecanismo envolve incluir informações em uma resposta de servidor que indica se a composição é suportada pelo servidor. Embora essa informação possa assumir qualquer forma, o uso de um cabeçalho de resposta novo 190 é conveniente porque os clientes normalmente ignoram cabeçalhos não reconhecidos em uma resposta HTTP. Além disso, algoritmos conhecidos para analisar cabeçalho podem ser prontamente estendidos para processar um nome de campo ou cabeçalho novo. Na alternativa, um indicador de composição pode assumir a forma de um valor novo para um cabeçalho padrão, uma linha de estado especial, etc.

Para estabelecer a disponibilidade de composição, o cliente 172 executa um processo 192 que inicia com o envio de uma solicitação OPTIONS padrão 194 (solicitação 194 é somente um exemplo). Um processo 196 no servidor 174 recebe a solicitação OPTIONS 194 e gera uma resposta como resposta 198 que inclui um indicador de composição, nessa modalidade, cabeçalho de resposta não padrão 190. O nome efetivo do cabeçalho de resposta não padrão 190 não é importante de outra maneira do que ser conhecido antecipadamente pelo cliente 172 de modo que quando o processo 192 do cliente 172 recebe a resposta 198, pode reconhecer a mesma e comunicar com o servidor 174 como apropriado.

A figura 8 mostra como as solicitações podem ser formatadas para invocar travamento composto. A parte superior corresponde aos métodos estendidos 144 (vide a figura 5) para GET ou travamento POST, e a parte inferior corresponde aos métodos estendidos 146 para travamento PUT. Em uma modalidade, verbos de solicitação de WebDAV e HTTP comuns 220 GET, POST e PUT são compostos com solicitações de travamento utilizando várias combinações de um cabeçalho Lock-Token padrão 222 e um cabeçalho de intervalo de travamento estendido ou não padrão 224, por exemplo "X-MSDAVEXTLock-Timeout". O cabeçalho de intervalo de travamento 224 tem um valor de 0 ou mais segundos.

O cabeçalho de intervalo de travamento 224 significa a criação de um novo travamento de acordo com o valor do cabeçalho de intervalo de travamento 224. Se o cabeçalho Lock-Token 222 for incluído então o cabeçalho de intervalo de travamento 224 sinaliza a renovação de um travamento existente. Se o cabeçalho de intervalo de travamento 224 é definido em 0 segundo então um destravamento é indicado (nesse caso, o cabeçalho Lock-Token 222 e um token correto são necessários para destravar o arquivo). Além disso, um cabeçalho de Lock-Token 222 e token são preferivelmente incluídos na resposta a qualquer operação de gravar em um recurso travado. O exemplo de solicitação 228 mostra como se pareceria uma solicitação POST+UNLOCK típica. Observe a inclusão de um cabeçalho de Lock-Token 222 e um cabeçalho de intervalo de travamento 224.

Com referência ao verbo PUT combinado com uma operação de travamento, observe que o cabeçalho Lock-token 222 e token correto são necessários para modificar um recurso travado. Nenhum token é necessário se o recurso não é travado. Se nenhum token for incluído porém um tempo de travamento for especificado, então ocorre a lógica de travamento natural; um travamento é concedido se nenhum travamento existir, e o PUT e travamento são negados se já existir um travamento. Em resumo, se o token correto for incluído com uma solicitação de PUT o cliente pode executar qualquer operação de PUT ou qualquer operação de PUT combinada com uma operação de travamento. Uma solicitação PUT+REFRESH típica é mostrada pela solicitação 230. O valor de intervalo de travamento de 120 segundos indica uma renovação ou redefinição do tempo de vida do travamento para rodar por outro período de 120 segundos, e o lock token é a chave que o servidor utiliza para autorizar tanto a operação PUT como a operação REFRESH. Em uma modalidade preferida um cabeçalho Lock-Token incluída em uma operação não gravar é ignorada; isto é "GET+verify um travamento existente" não é suportado.

A figura 9 mostra um mecanismo para compor métodos de propriedade com verbos ou métodos WebDAV ou HTTP. Esses métodos compostos correspondem aos métodos 148, 150 na figura 5. Os métodos de propriedade compostos utilizam dois indicadores; um valor de cabeçalho do tipo conteúdo especial 240 (por exemplo, "multipart/MSDAVEXTPrefixEncoded") e um cabeçalho de extensões especiais 242 com vários valores

possíveis como "PROPFIND" e "PROPPATCH". As combinações na tabela 244 são auto-explanatórias e os métodos resultantes permitem que um recurso seja acessado ou modificado enquanto ao mesmo tempo obtém ou define uma ou mais propriedades do recurso relevante. Além disso, o cabeçalho de comprimento-Conteúdo padrão será utilizado e fornecerá o valor do corpo de mensagem total ou carga útil, que também pode incluir propriedades bem como recursos (discutidos adicionalmente abaixo).

Em conformidade com as regras da tabela 244, um exemplo de solicitação GET+PROPFIND 246 é mostrado. Observe a inclusão de uma indicação da porção PROPFIND do método na forma de um cabeçalho de extensões especiais 242 com o verbo ou valor apropriado.

Embora em uma modalidade métodos relacionados à propriedade sejam compostos sobre outros métodos utilizando cabeçalhos e uma extensão de corpo de mensagem, outras abordagens também podem ser utilizadas. Por exemplo, os métodos PROPFIND e PROPATCH de WebDAV poderiam ser sobrecarregados utilizando novos cabeçalhos. Além disso, há diferentes modos de combinar um recurso e um conjunto de propriedades e um corpo de mensagem. Todas as propriedades podem ser colocadas em cabeçalhos separados, uma vez que a maioria dos conjuntos de propriedade é de tamanho manejável. As propriedades poderiam ser atribuídas a cabeçalhos diferentes respectivos, embora isso exigiria mais codificação para tratar do transporte de propriedades. Em outra modalidade, todas as propriedades (estrutura XML)

podem ser colocadas em um cabeçalho maior, entretanto, os cabeçalhos poderiam potencialmente se tornar maiores do que os buffers que alguns servidores de rede alocam para tratamento de cabeçalho.

É possível que algumas implementações podem necessitar definir simultaneamente propriedades (PROPATCH) e obter propriedades (PROPFIND) de um recurso. Por exemplo, para determinar se uma propriedade específica foi adequadamente definida, ou determinar a qual propriedade foi definida antes de ser mudada com PROPPATCH. Nesse caso, "PROPPATCH" e "PROPFIND" podem ambos ser incluídos, e uma convenção pode ser estabelecida para o local de propriedades enviadas e retornadas no corpo da mensagem.

Embora o protocolo de WebDAV não especifique propriedades específicas para recursos, algumas propriedades típicas são análogas a propriedades de objetos em um sistema de arquivo, por exemplo, tamanho de conteúdo, data de criação, data da última modificação, usuário de última modificação, tipo de pasta especial, etiqueta de recurso, atributos de arquivo, tempo de criação, último horário de acesso, último horário modificado, e assim por diante.

A figura 9 mostra também um exemplo de resposta GET+PROPFIND 250. Observe que o campo de cabeçalho de tipo de conteúdo 240 sinaliza a presença de um corpo de mensagem de múltiplas partes utilizando a extensão especial "multipart/MSDAVEXTPrefixEncodedheader". As informações relacionadas a travamento não são necessárias para o método PUT+PROPPATCH, porém podem significar a presença de um

travamento do lado do servidor. O campo de cabeçalho do tipo de conteúdo 240 significa a presença do corpo de mensagem de múltiplas partes 248 em um corpo de mensagem 64. Genericamente essas partes múltiplas são divididas por um campo de comprimento seguido por dados correspondentes, em outras palavras, o corpo de mensagem 64 contém um ou mais pedaços de dados discretos, cada um precedido por um indicador de comprimento correspondente. Os tamanhos dos campos de comprimento e os tamanhos de seus dados são somados ao cabeçalho de comprimento-Conteúdo padrão. O exemplo de resposta 250 na figura 9 tem uma seção de propriedades e uma seção de recurso, cada uma precedida por um campo de comprimento respectivo, por exemplo, um número inteiro de 64 bits. Como de autoria de composto é projetado como uma extensão de HTTP, o corpo de mensagem de HTTP padrão 64 é utilizado. Como as propriedades podem necessitar ser trocadas em uma mensagem que pode incluir, também, um recurso como um documento de HTML, os pares de dados-comprimento permitem que tanto propriedades como recursos sejam contidas em um mesmo corpo de mensagem 64. O cabeçalho de comprimento-conteúdo padrão dá o comprimento total do corpo de mensagem 64/248 e pode ser utilizado, em combinação com os indicadores de comprimento, para analisar as peças substantivas de conteúdo no corpo de mensagem 248.

Com referência novamente aos métodos 152, 154 na figura 5 (POST|GET + PROPFIND +LOCK|REFRESH|UNILOCK, PUT + PROPATCH + LOCK|REFRESH|UNLOCK), esses métodos podem ser implementados pela combinação das propriedades e extensões

de travamento discutidas acima. Como a funcionalidade de travamento e funcionalidade de propriedades é logicamente separada, os métodos e cabeçalhos discutidos acima podem co-existir facilmente em uma mensagem. A figura 10 mostra métodos compostos adicionais. A mensagem superior é um exemplo de uma solicitação PUT+PROPPATCH+UNLOCK 270. Observe o tamanho do corpo, incluindo o tamanho dos campos de comprimento, é 114234. A mensagem inferior é um exemplo de uma resposta correspondente 272. Uma resposta que é bem sucedida não necessita diferir da resposta para uma solicitação PUT normal. A ausência de um cabeçalho lock token indica um destravamento bem sucedido. A figura 11 mostra um exemplo de uma solicitação de método POST+PROPFIND+LOCK 290 e uma resposta correspondente 292. A solicitação 290 faz com que o servidor coloque um recurso ou arquivo, defina algumas propriedades e destrave o arquivo ou recurso.

A figura 12 mostra uma tabela de tratamento de erro e exemplos de uma resposta 302 utilizando tratamento de erro estendido. Diversos tipos de erros podem ocorrer ao criar um recurso em um servidor através de uma solicitação de autoria HTTP estendida, por exemplo, permissões insuficientes, um recurso checado por outro usuário ou não checado, uma violação de cota, ou um tipo de arquivo ou nome de arquivo bloqueado, a presença de um vírus, etc. Outros erros como propriedade ausente podem ocorrer ao tentar gravar em um arquivo ou recurso. Quando um erro de autoria ocorre em um servidor, tipicamente o servidor pode ter

informações ricas em nível de sistema sobre o erro. Previamente, quando um módulo ou servidor que implementa métodos WebDAV encontra um erro traduz aquele erro em um código de erro HTTP padrão. Um cliente poderia tentar fornecer uma mensagem útil sobre o código de erro, talvez utilizando uma stringg de mensagem hardcoded correspondente. Entretanto, os códigos de erro de HTTP padrão não são ricos o bastante para suportar o número e tipos de erros que os usuários podem encontrar utilizando autoria de HTTP estendido. Portanto, uma extensão é opcionalmente fornecida para estender as informações de erro realimentadas para um cliente, enquanto mantém o código de erro de HTTP existente, que permite compatibilidade para trás. Esse tratamento de erro estendido é realizado pela inclusão em respostas informações específicas a erros em nível de sistema no servidor.

Como visto na figura 12, o tratamento de erro estendido pode ser realizado utilizando um novo cabeçalho HTTP, por exemplo "X-MSDAVEXT_ERROR: decimal; string". A porção decimal é um código que mapeia para um erro em nível de sistema como um erro de controle de arquivo Unix ou um erro Win32. Preferivelmente, a porção String está no formato UTF-8.

Com relação a extensões de composição de protocolos de autoria de rede em geral, deve ser observado que alguns servidores proxy podem tentar interpretar solicitações e enviar de volta respostas armazenadas em cache. Portanto, é preferível que clientes utilizem somente

as novas extensões ou métodos com POST em vez de GET. Além disso, ao responder a um método concatenado ou verbo como discutido acima, um servidor deve marcar uma resposta para indicar que não deve ser armazenada em cache, utilizando, por exemplo, um cabeçalho como "cache-control: private".

Os processos de servidor e cliente para utilizar métodos de autoria composta estendidos são relativamente diretos, dadas as convenções como discutido acima. Documentação e código de fonte disponível ao público podem ser consultados para determinar como implementar servidores e clientes com a funcionalidade para executar métodos de autoria atômicos e, em particular, funcionalidade de propriedade e travamento. Essa funcionalidade pode ser executada em modo serial quando um método composto é encontrado. Por exemplo, ao passo que anteriormente um servidor pode ter tido uma função para tratar um método LOCK e uma função para tratar um método POST, aproximadamente, essas funções podem ser invocadas consecutivamente quando o método POST+LOCK composto é recebido.

Embora HTTP e WebDAV tenham sido discutidos acima, espera-se que as idéias discutidas acima sejam aplicáveis a quaisquer futuras variações ou versões de HTTP e WebDAV. Além disso, um protocolo padrão é considerado como se referindo a qualquer protocolo padrão futuro ou atual.

Adicionalmente a aspectos de passagem de erro estendido, pode ser fornecido um meio legível por máquina volátil ou não volátil que armazena informações para permitir que um dispositivo execute um processo para atender

solicitações de clientes, o processo incluindo: tratar de solicitações de get HTTP padrão, solicitações de post HTTP padrão, e solicitações de opções de HTTP padrão e enviar respostas a clientes correspondentes; tratar de solicitações de autoria padrão ou não padrão de HTTP que orientem o dispositivo para recurso de travar/destravar ou orientem o dispositivo a obter ou definir propriedades de recursos; e quando ocorrem erros ao tratar das solicitações de autoria de HTTP,retornar respostas que compreendem informação de erro que não seja um código de estado de http.A informação de erro pode corresponder a um erro de sistema do dispositivo que fez com que os erros ocorressem. A informação de erro pode incluir um nome de cabeçalho de erro estendido e um campo de cabeçalho associado compreendendo identificar e/ou descrever um erro correspondente, e além disso, em tal caso o campo de cabeçalho pode incluir um código de erro de sistema do dispositivo, ou o campo de cabeçalho pode incluir uma cadeia descrevendo especificamente o erro, ou o campo de cabeçalho pode incluir um código de erro de sistema do dispositivo (o código de erro sendo ou identificando um erro de sistema do dispositivo), ou o erro do sistema compreende um erro de travamento de arquivo, ou um erro de leitura de diretório ou arquivo, ou um erro de gravação de diretório ou arquivo.

Adicionalmente a outro aspecto de passagem de erro estendido, um meio volátil ou não volátil para armazenagem de dados digitais pode ser fornecido, o meio armazenado uma resposta de HTTP, a resposta de HTTP

incluindo: um cabeçalho de código de estado de HTTP padrão e código de erro correspondente; e uma indicação de um erro do lado de servidor, onde a indicação não é definida por um HTTP padrão. A indicação pode incluir um campo de cabeçalho de HTTP especificamente definido para transferir informações de erro estendido diferentes de códigos de erro HTTP padrão, em cujo caso o campo de cabeçalho pode incluir um nome de campo não definido pelo protocolo de HTTP padrão e um corpo de campo que contém informações sobre o erro do lado do servidor, em cujo caso é adicionalmente possível que o corpo de campo identifique um tipo específico de erro do lado do servidor e esse erro não corresponda a um código de erro HTTP padrão. O corpo de campo pode incluir um código de erro de sistema operacional ou uma cadeia descrevendo um erro do sistema operacional, e além disso, o erro do lado de servidor pode compreender um erro de travamento de sistema operacional, ou um erro de leitura ou gravação de um arquivo de servidor ou diretório de servidor.

Ainda em outra modalidade de tratamento de erro estendido, pode ser fornecida uma armazenagem volátil ou não volátil para uso com um dispositivo de processamento e armazenagem de informações para permitir que o dispositivo de processamento execute um processo, o processo compreendendo: gerar uma solicitação de HTTP e enviar a solicitação de HTTP para um servidor; receber a partir do servidor uma resposta de HTTP para a solicitação de HTTP; e analisar a resposta de HTTP para um cabeçalho de erro estendido não padrão e extrair a partir da informação de

cabeçalho de erro estendido não padrão sobre um erro no servidor. A solicitação de HTTP pode incluir ainda um cabeçalho de código de estado HTTP padrão e número de erro correspondente. A informação sobre o erro no sistema pode compreender detalhe sobre um tipo específico de erro de sistema operacional ou sistema de arquivo no servidor. A informação sobre o erro no servidor pode compreender um número de erro de sistema operacional. A informação sobre o erro no servidor pode incluir uma cadeia descrevendo um erro de sistema operacional. A informação sobre o erro no servidor pode identificar ou descrever um erro em nível de sistema do servidor, além do que as solicitações de HTTP podem compreender uma solicitação de autoria baseada em HTTP, quer composta ou não composta, e o erro no servidor foi um erro executando um método relacionado a propriedades ou relacionado a tratamento.

Concluindo, aqueles versados na técnica perceberão que dispositivos de armazenagem utilizados para armazenar instruções de programa podem ser distribuídos através de uma rede. Por exemplo um computador remoto pode armazenar um exemplo do processo descrito como software. Um computador terminal ou local pode acessar o computador remoto e fazer download de uma parte ou de todo software para rodar o programa. Alternativamente o computador local pode fazer download de partes do software conforme necessário, ou processar de forma distributiva, pela execução de algumas instruções de software no terminal local e algumas no computador remoto (ou rede de computador). Aqueles versados

na técnica perceberão também que pela utilização de técnicas convencionais conhecidas por aqueles versados na técnica, todas ou uma parte das instruções de software podem ser realizadas por um circuito dedicado, como um DSP, disposição lógica programável, ou similar.

Todas as modalidades e características discutidas acima podem ser realizadas na forma de informações armazenadas em meio legível por dispositivo ou computador volátil ou não volátil. Isso é considerado como incluindo pelo menos meios como CD-ROM, meios magnéticos, ROM flash, etc., que armazenam instruções executáveis por máquina, ou código de fonte, ou qualquer outra informação que pode ser utilizada para permitir que um dispositivo de computação execute as diversas modalidades. Isso também é considerado como incluindo pelo menos memória volátil como RAM que armazena informações com instruções de CPU durante execução de um programa realizando uma modalidade.

REIVINDICAÇÕES

1. Meio legível por máquina volátil ou não volátil, **CARACTERIZADO** por armazenar informações para permitir que um dispositivo execute um processo para atender solicitações, o processo compreendendo:

tratar solicitações get HTTP padrão, solicitações post HTTP padrão, e solicitações options HTTP padrão;

tratar de solicitações que se conformem com um protocolo de autoria HTTP, as solicitações compreendendo solicitações put, solicitações copy, solicitações move, solicitações propfind, e solicitações proppatch; e

tratar de solicitações de autoria compostas que não se conformam com o protocolo de autoria HTTP, as solicitações compostas compreendendo solicitações put+proppatch e solicitações post+propfind.

2. Meio, de acordo com a reivindicação 1, o processo sendo **CARACTERIZADO** ainda por compreender executar funcionalidade put e funcionalidade proppatch responsiva a uma solicitação put+proppatch, ou executar funcionalidade get e funcionalidade propfind responsiva a uma solicitação get+propfind, ou executar funcionalidade post e funcionalidade propfind responsiva a uma solicitação post+propfind.

3. Meio, de acordo com a reivindicação 1, **CARACTERIZADO** pelo fato de que as solicitações tratadas que não se conformam com o protocolo de autoria HTTP compreendem ainda solicitações que, em solicitações únicas, especificam tanto uma operação post, get, ou put, como uma operação

travar, destravar, ou travar renovação.

4. Meio, de acordo com a reivindicação 1, **CARACTERIZADO** pelo fato de que o processo compreende ainda responder a uma solicitação options HTTP pela geração de uma resposta que inclui informações de cabeçalho indicando suporte para solicitações únicas com solicitações de autoria compostas.

5. Meio, de acordo com a reivindicação 1, **CARACTERIZADO** pelo fato de que as solicitações de autoria compostas compreendem um campo de método que se conforma ao protocolo de autoria HTTP e um campo de cabeçalho ou nome de cabeçalho não definido pelo protocolo de autoria HTTP.

6. Meios, de acordo com a reivindicação 1, **CARACTERIZADO** pelo processo compreender ainda a inclusão de um conjunto de propriedades e um recurso em um corpo de uma mensagem de resposta.

7. Meio volátil ou não volátil para armazenagem de dados digitais, **CARACTERIZADO** pelo fato de que o meio armazena uma solicitação de HTTP indicando métodos de um protocolo público padrão que estende um HTTP, onde o protocolo público padrão define métodos e cabeçalhos para manipular remotamente recursos em um servidor de HTTP, a solicitação de HTTP armazenada compreendendo:

um primeiro indicador de método indicando um primeiro método, onde o primeiro método é especificado pelo HTTP ou o protocolo público padrão estendendo o HTTP, e onde o primeiro indicador de método se conforma ao HTTP ou a extensão pública padrão do HTTP; e

um segundo indicador de método indicando um segundo método, onde o segundo método é especificado pela extensão pública padrão do HTTP.

8. Meio, de acordo com a reivindicação 7, **CARACTERIZADO** pelo fato de que o segundo indicador de método compreende um cabeçalho na solicitação de HTTP.

9. Meio, de acordo com a reivindicação 7, **CARACTERIZADO** pelo fato de que a extensão pública padrão do HTTP não especifica operações de autoria composta em uma única mensagem de solicitação.

10. Meio, de acordo com a reivindicação 7, **CARACTERIZADO** pelo fato de que o primeiro indicador de método compreende um verbo get, post ou put e o segundo método compreende um método de propriedades ou travamento dirigido a um recurso de servidor.

11. Meio, de acordo com a reivindicação 10, **CARACTERIZADO** pelo fato de que o protocolo público padrão especifica somente métodos de autoria atômicos.

12. Aparelho de servidor, **CARACTERIZADO** por compreender o meio, do tipo definido na reivindicação 7 e configurado para receber a solicitação de HTTP e executar o primeiro método e o segundo método na solicitação de HTTP.

13. Aparelho de cliente, **CARACTERIZADO** por compreender o meio do tipo definido na reivindicação 7 e configurado para gerar a solicitação de HTTP.

14. Meio de armazenagem volátil ou não volátil, **CARACTERIZADO** por ser para uso com um dispositivo de processamento e armazenar informações para permitir que o

dispositivo de processamento execute um processo, o processo compreendendo:

gerar uma solicitação de HTTP, onde a solicitação de HTTP compreende um campo de método, uma pluralidade de campos de cabeçalhos, e informações de cabeçalho; onde

o campo de método compreende um verbo que indica um método definido por um protocolo de autoria distribuído que estende um HTTP, e o verbo se conforma ao protocolo de autoria distribuído; onde

a pluralidade de campos de cabeçalho compreende nomes de campo de cabeçalho e valores de campo de cabeçalho, e os campos de cabeçalho se conformam ao HTTP e/ou o protocolo de autoria distribuído; onde

a informação de cabeçalho compreende um nome de campo de cabeçalho ou um valor de campo de cabeçalho, qualquer um deles indica outro método definido pelo protocolo de autoria distribuído; e onde

ter a informação de cabeçalho e o campo de método na mesma solicitação de HTTP não se conforma ao protocolo de autoria distribuído.

15. Meio, de acordo com a reivindicação 14, **CARACTERIZADO** pelo fato de que o protocolo de autoria distribuído compreende um protocolo público padrão.

16. Meio, de acordo com a reivindicação 14, **CARACTERIZADO** pelo fato de que o verbo compreende "get", "post" ou "put", e em que o outro método compreende um método de travar, ou um método de destravar, ou um método de travar renovação, ou um método de definir propriedade, ou um

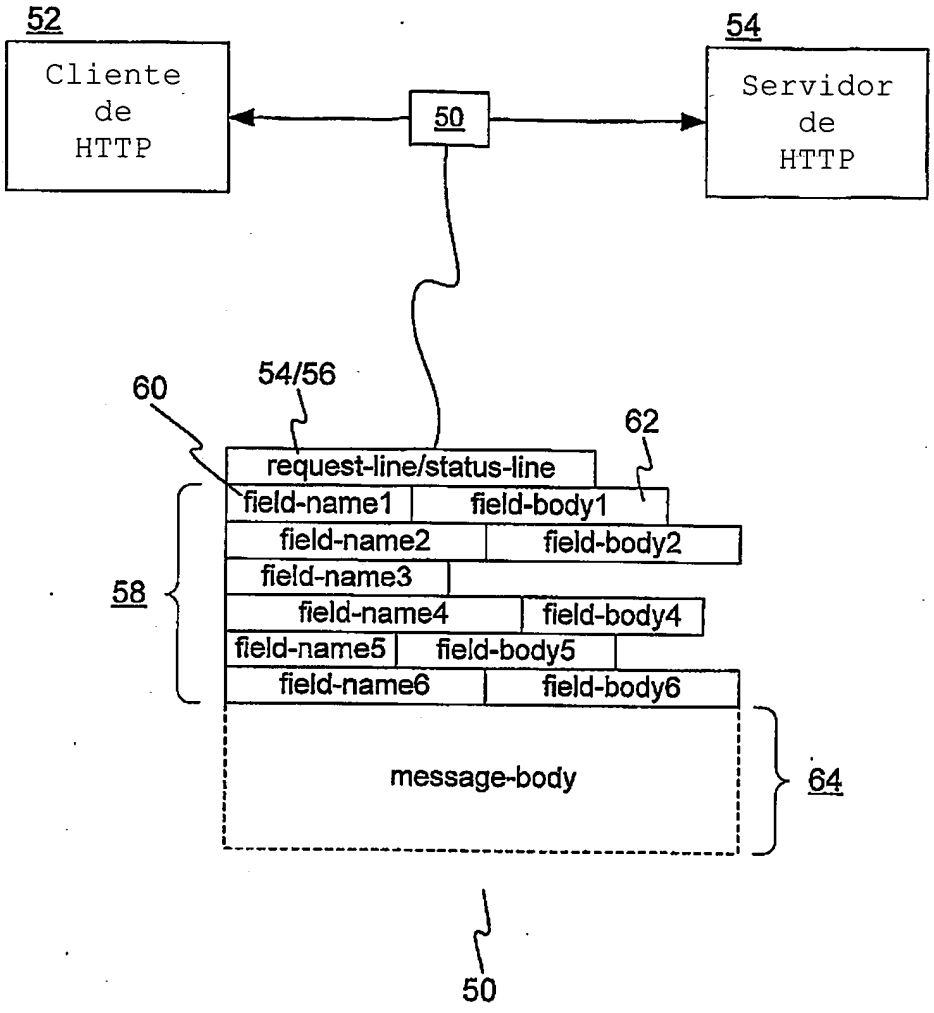
método de obter propriedade.

17. Meio, de acordo com a reivindicação 14, **CARACTERIZADO** pelo fato de que a informação de cabeçalho compreende um nome de campo de cabeçalho não definido pelo HTTP ou pelo protocolo de autoria distribuído.

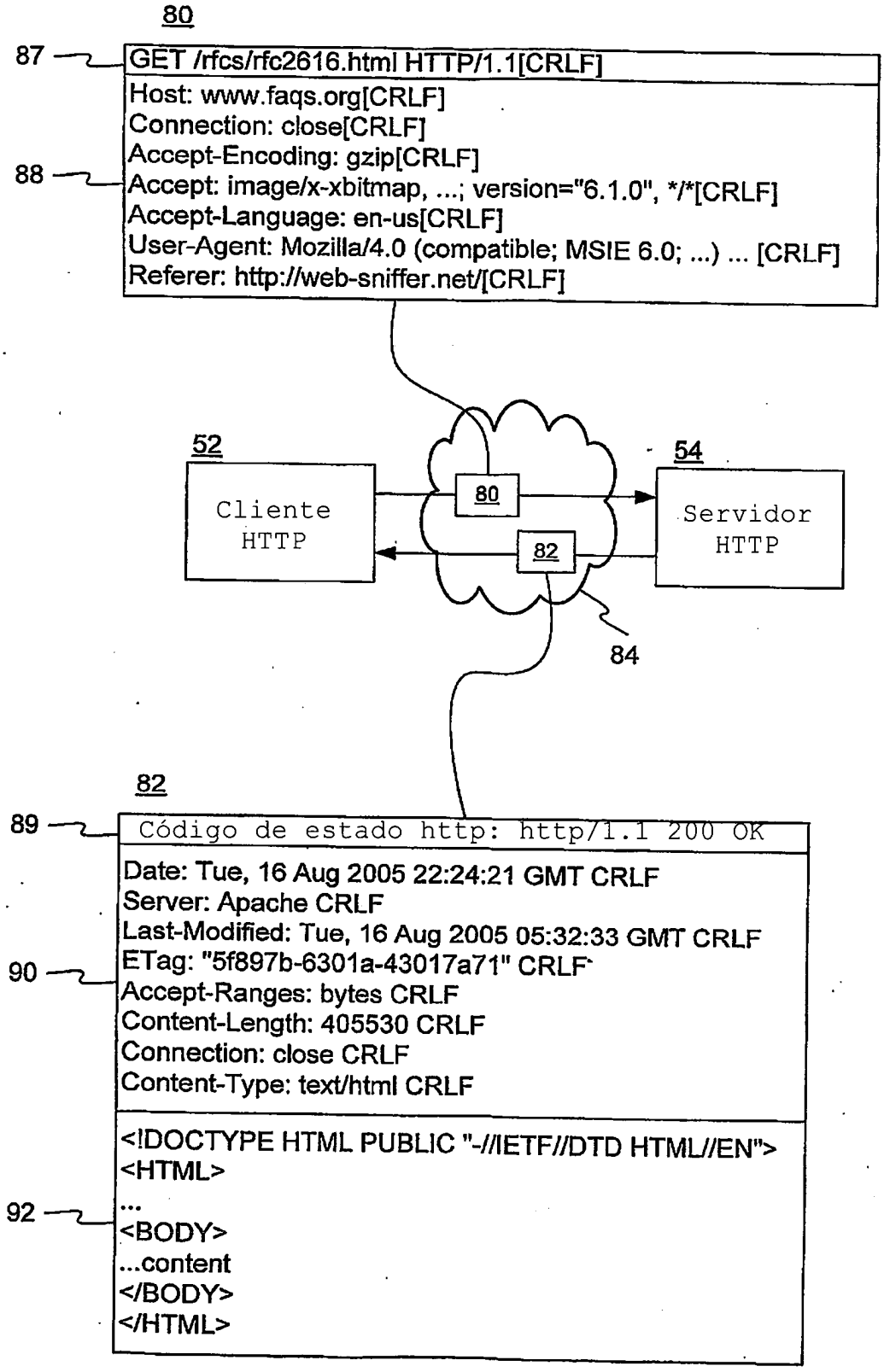
18. Meio, de acordo com a reivindicação 14, **CARACTERIZADO** pelo fato de que a informação de cabeçalho compreende um valor em um corpo de campo de cabeçalho.

19. Meio, de acordo com a reivindicação 14, **CARACTERIZADO** pelo fato de que o protocolo de autoria distribuído define métodos para criar, alterar, e mover documentos ou pastas em um servidor HTTP.

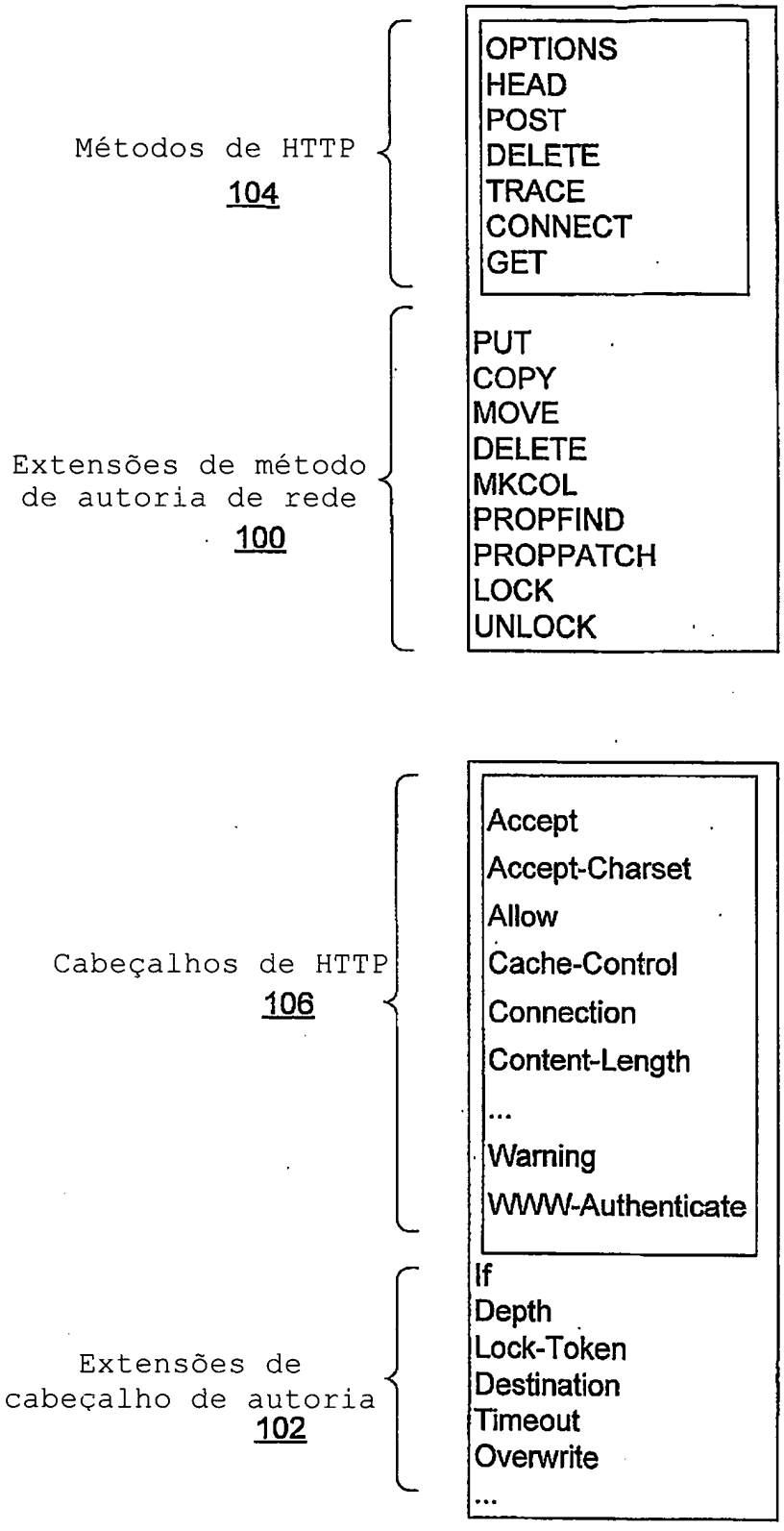
20. Meio, de acordo com a reivindicação 14, **CARACTERIZADO** pelo fato de que o protocolo de autoria distribuído compreende uma versão do protocolo WebDAV.



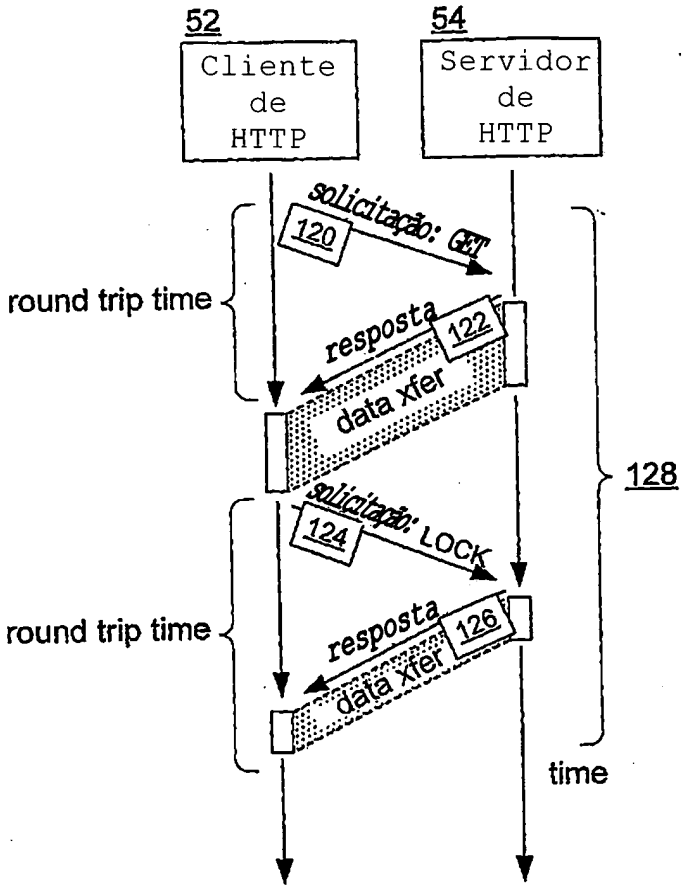
Técnica Relacionada
Fig. 1



Técnica Relacionada
Fig. 2



Técnica Relacionada
Fig. 3



Técnica Relacionada
Fig. 4

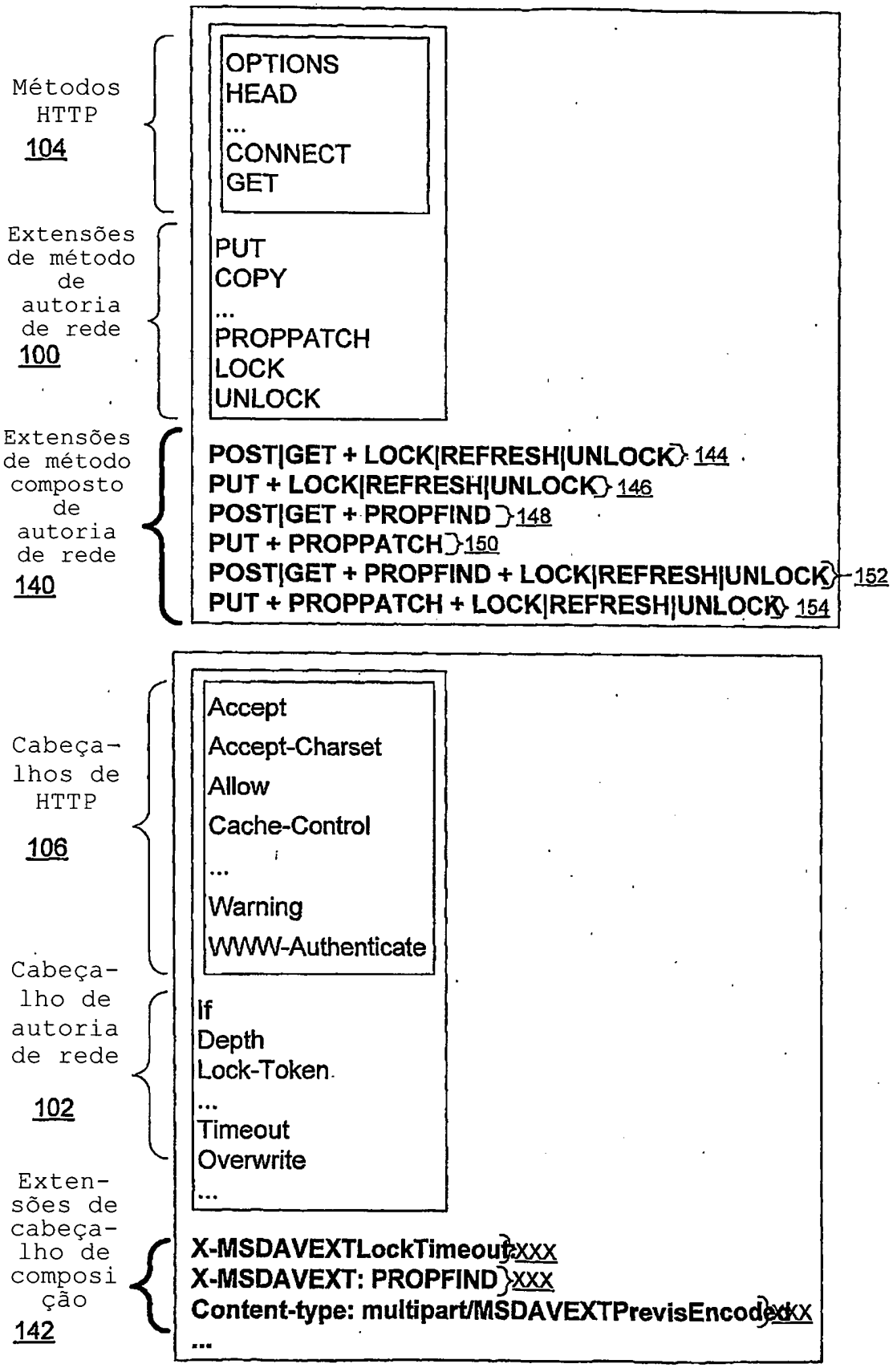

Métodos HTTP 104
OPTIONS
HEAD
...
CONNECT
GET
Extensões de método de autoria de rede 100
PUT
COPY
...
PROPPATCH
LOCK
UNLOCK
Extensões de método composto de autoria de rede 140
POST|GET + LOCK|REFRESH|UNLOCK 144
PUT + LOCK|REFRESH|UNLOCK 146
POST|GET + PROPFIND 148
PUT + PROPPATCH 150
POST|GET + PROPFIND + LOCK|REFRESH|UNLOCK 152
PUT + PROPPATCH + LOCK|REFRESH|UNLOCK 154
Cabeça-lhos de HTTP 106
Accept
Accept-Charset
Allow
Cache-Control
...
Warning
WWW-Authenticate
Cabeça-lho de autoria de rede 102
If
Depth
Lock-Token
...
Timeout
Overwrite
...
Exten-sões de cabeça-lho de composi ção 142
X-MSDAVEXTLockTimeout: XXX
X-MSDAVEXT: PROPFIND XXX
Content-type: multipart/MSDAVEXTPrevisEncoded XXX
...

FIG. 5

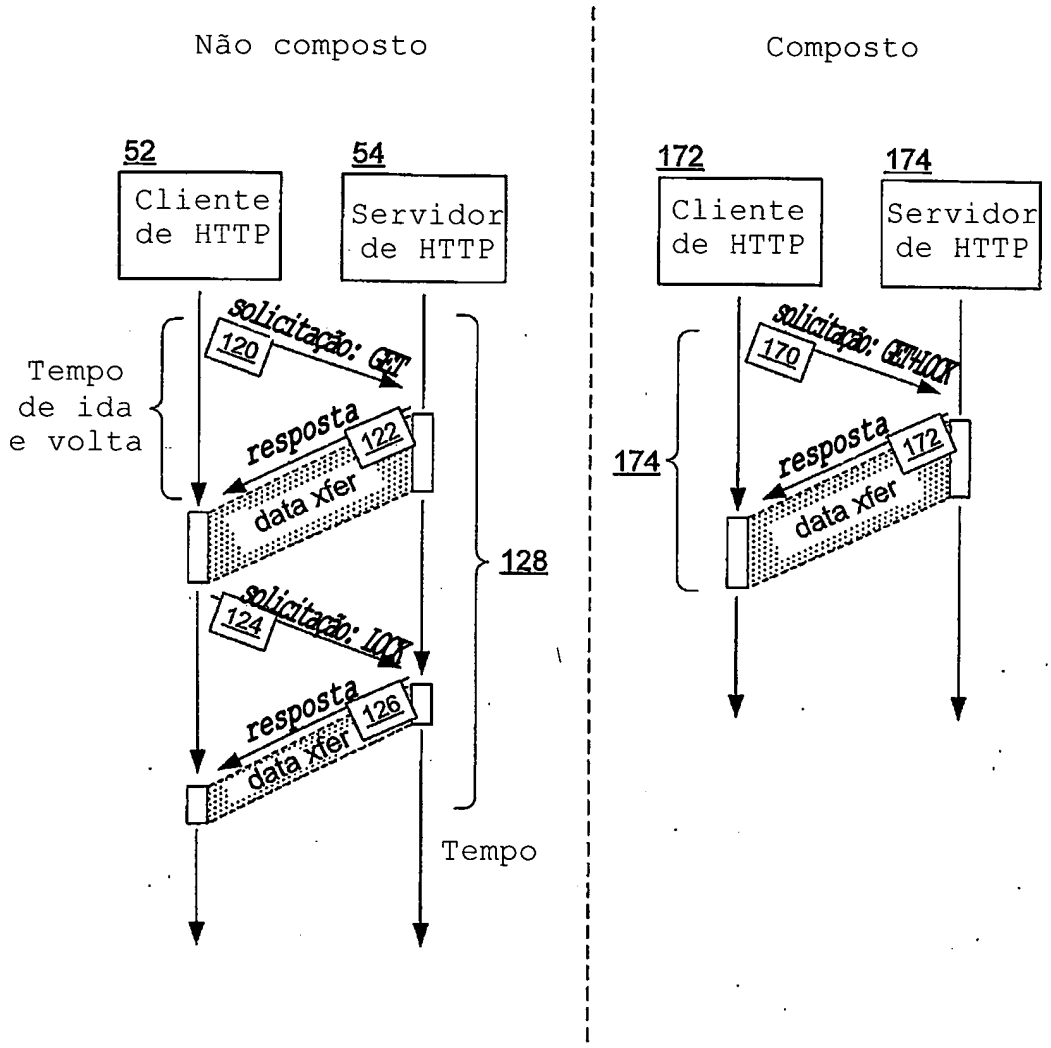

Não composto
Composto
52
Cliente de HTTP
54
Servidor de HTTP
172
Cliente de HTTP
174
Servidor de HTTP
Tempo de ida e volta
solicitação: GET
120
resposta
122
data xfer
128
solicitação: 100
124
resposta
126
data xfer
Tempo
174
solicitação: GETHOOK
170
resposta
172
data xfer

FIG. 6

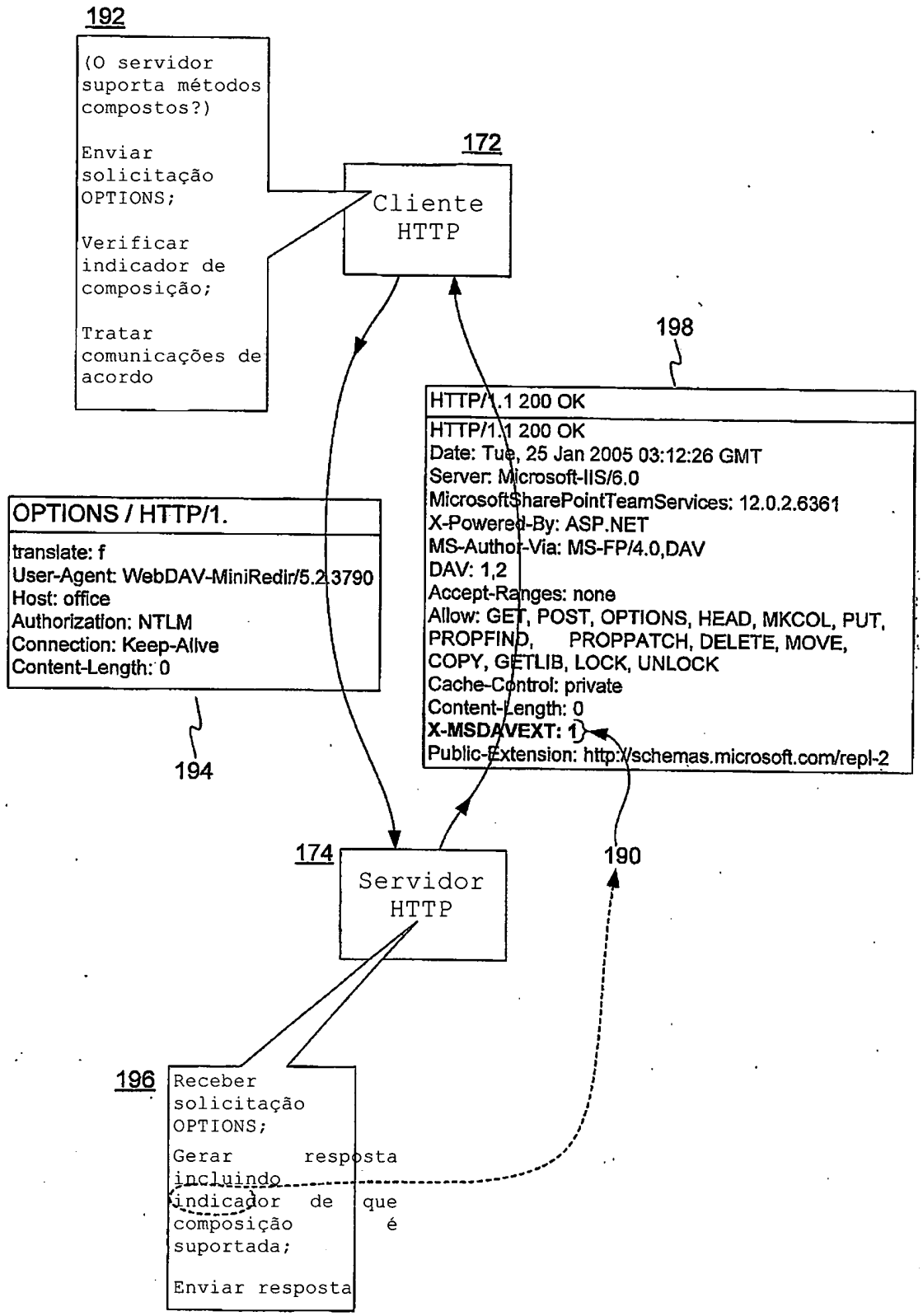

192
(O servidor suporta métodos compostos?)
Enviar solicitação OPTIONS;
Verificar indicador de composição;
Tratar comunicações de acordo
172
Cliente HTTP
198
HTTP/1.1 200 OK
HTTP/1.1 200 OK
Date: Tue, 25 Jan 2005 03:12:26 GMT
Server: Microsoft-IIS/6.0
MicrosoftSharePointTeamServices: 12.0.2.6361
X-Powered-By: 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
Servidor HTTP
190
196
Receber solicitação OPTIONS;
Gerar resposta incluindo indicador de que composição é suportada;
Enviar resposta

FIG. 7

Exemplo de solicitação POST+UNLOCK:

228

```
POST http://www.example.com/name_of_file.htm
...
Content-Type: application/x-www-form-urlencoded; charset=utf-8; ...
Lock-Token: opaquelocktoken:{3932E32A-8825-494E-A19E-E714A7A741A8}20041130T183232Z
X-MSDAVEXTLockTimeout: second-0
...
<content>
```

222 — Lock-Token: opaquelocktoken:{3932E32A-8825-494E-A19E-E714A7A741A8}20041130T183232Z

224 — X-MSDAVEXTLockTimeout: second-0

220	222 Cabeçalhos	224	226
HTTP verb	Lock-Token:	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
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

144 — GET|POST rows

146 — PUT rows

Exemplo de solicitação PUT+REFRESH:

230

```
PUT http://www.example.com/somefile/name_of_file.htm
...
Content-Type: application/x-www-form-urlencoded; charset=utf-8; ...
X-MSDAVEXTLockTimeout: second-120
Lock-Token: opaquelocktoken:{3932E32A-8825-494E-A19E-E714A7A741A8}20041130T183232Z
...
<content>
```

224 — X-MSDAVEXTLockTimeout: second-120

222 — Lock-Token: opaquelocktoken:{3932E32A-8825-494E-A19E-E714A7A741A8}20041130T183232Z

FIG. 8

244

Verbo HTTP	240 Cabeçalhos		242 Resultado (método eficaz)
	Content-type:	X-MSDAVEXT:	
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 Exemplo de solicitação de GET+PROPFIND:

```
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 Exemplo de resposta GET+PROPFIND:

```
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-#### (some number of seconds)
Lock-Token: opaquelocktoken:{4A7A741A8}20041130T183232Z
Content-Length: ##### (size of body 248)
Public-Extension: http://schemas.microsoft.com/repl-2
```

240

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

248

FIG. 9

270 Exemplo de solicitação de PUT+PROPPATCH+UNLOCK:

```
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 Exemplo de resposta PUT+PROPPATCH+UNLOCK:

```
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 Exemplo de solicitação POST+PROPFIND+LOCK:

```
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 Exemplo de resposta POST+PROPFIND+LOCK:

```
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

Código de erro estendido (decimal)	Na error code which the client can map to a system-level error code
Stringg (codificado URL)	Uma cadeia com informação estendida para clientes utilizarem

302

HTTP/1.1 401 Não autorizado
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

PI 0614275-3

RESUMO

"COMPOSIÇÃO DE PROTOCOLO DE AUTORIA HTTP"

Convenções para estender métodos de autoria de rede composta para um protocolo de autoria de rede como WebDAV. Mais particularmente, uma solicitação pode ser fornecida com informações especiais de cabeçalho para significar um método composto com um método indicado por um verbo na solicitação. Técnicas para clientes e servidores utilizarem as extensões de autoria de rede. Tratamento de erro estendido para permitir que os servidores forneçam informações mais ricas de erro de autoria da rede para clientes.