



Industria y Comercio

SUPERINTENDENCIA

*No hay
registros*

PCT

SOLICITUD

PATENTE DE INVENCION

PCT

EXPEDIENTE N°

05-50650

TITULO: SISTEMA Y MÉTODO PARA
ADMINISTRAR Y COMUNICAR
ACTUALIZACIONES DE SOFTWARE

CLASIFICACION INTERNACIONAL: _____

SOLICITANTE: MICROSOFT CORPORATION

DOMICILIO: REDMOND, WASHINGTON, ESTADOS
UNIDOS DE AMERICA

APODERADO: DARIO CARDENAS NAVAS

BOGOTÁ, D.C. _____

PETITORIO

Industria y Comercio
SUPERINTENDENCIA

SUPERINDUSTRIA Y COMERCIO
Radiación : 05050650 00000000 Folios: 169
Fecha (AMD): 2005-05-25 11:06:16
Trámite : 011 PCT-PATENT 376 FASE NACI 411 PRESENTAR
Dependencia: 2020 DIVISION DE NUEVAS CREACIONES

FORMULARIO UNICO DE SOLICITUD DE PATENTE PCT ENTRADA EN FASE NACIONAL

①

SOLICITUD DE:

☒ X

Patente de Invención

☐

Patente de Modelo de Utilidad

☒ X

Capítulo I

☐

Capítulo II

②

SOLICITANTE
(71)

Nombre: **MICROSOFT CORPORATION**

Dirección: One Microsoft Way Redmond, Washington 98052, Estados Unidos de America

Nacionalidad o Domicilio: REDMOND, WASHINGTON, ESTADOS UNIDOS DE AMERICA.

Lugar de Constitución: Estado de Washington, Estados Unidos de América

Teléfono: 3123600 Fax: 3122410

E-mail: general@cardenasycardenas.com

IDENTIFICACION

C.C. ☐

NIT ☐

C.E. ☐

Otro ☒ X

Cual _____
Número _____

③

REPRESENTANTE
O APODERADO

Nombre: **DARIO CARDENAS NAVAS**

Dirección: Carrera 7 No 71-52 Torre B Piso 9

Teléfono: 3123600 Fax: 3122410

E-Mail: general@cardenasycardenas.com

IDENTIFICACION

C.C. ☒ X

NIT ☐

C.E. ☐

Otro ☐

Cual _____
No 17.066.629 TP 1.250
BTA C.S.J.

④

INVENTOR(ES)
(72)

1. Nombre:
1. Michael Edward Meulemans
2. Aaron Averbuch
3. Jason Roberts
4. Ken Showman
5. Mazhar Mohammed
6. Joseph G. Dadzie

2. Dirección:
1. One Microsoft Way Redmond, Washington 98052, Estados Unidos de America
2. One Microsoft Way Redmond, Washington 98052, Estados Unidos de America
3. One Microsoft Way Redmond, Washington 98052, Estados Unidos de America
4. One Microsoft Way Redmond, Washington 98052, Estados Unidos de America
5. One Microsoft Way Redmond, Washington 98052, Estados Unidos de America
6. One Microsoft Way Redmond, Washington 98052, Estados Unidos de America

3. Nacionalidad o Domicilio:
1. REDMOND, WASHINGTON, ESTADOS UNIDOS DE AMERICA
2. REDMOND, WASHINGTON, ESTADOS UNIDOS DE AMERICA
3. REDMOND, WASHINGTON, ESTADOS UNIDOS DE AMERICA
4. REDMOND, WASHINGTON, ESTADOS UNIDOS DE AMERICA
5. REDMOND, WASHINGTON, ESTADOS UNIDOS DE AMERICA
6. REDMOND, WASHINGTON, ESTADOS UNIDOS DE AMERICA

SISTEMA Y MÉTODO PARA ADMINISTRAR Y COMUNICAR ACTUALIZACIONES DE SOFTWARE

5 PCT (86)

Solicitud Internacional No. PCT/US2004/023975 Fecha Julio 23, 2004
Publicación Internacional No. _____ Fecha _____

6 Título(54): SISTEMA Y MÉTODO PARA ADMINISTRAR Y COMUNICAR ACTUALIZACIONES DE SOFTWARE

7 Clasificación Internacional (51):

8 Prioridad

(33)País de Origen

(32)Fecha

(31)N° de Solicitud

U.S.A.

Diciembre 15, 2003

10/737.708

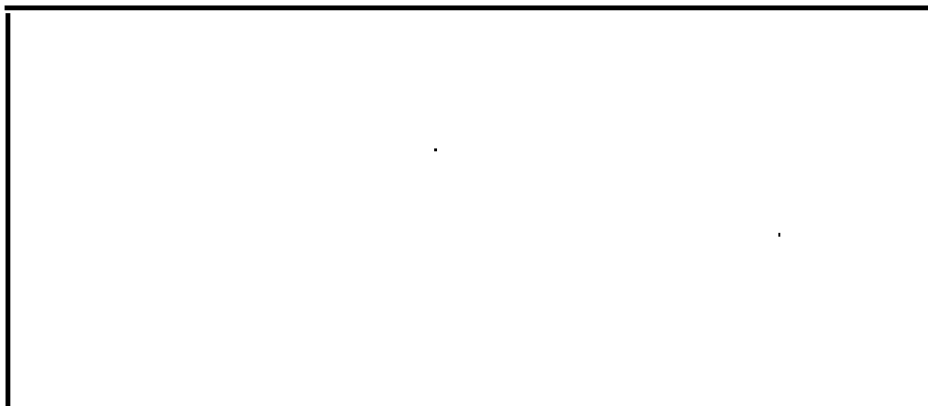
SI ☒ NO ☐

Comprobante de Pago No.: 05-37.299 Fecha Mayo 19, 2005

ANEXOS

- ☒ Comprobante de pago de la tasa de presentación de la solicitud.
- ☒ Documento que acredita la existencia y representación legal cuando el solicitante sea persona jurídica. (Protocolo 16.069)
- ☒ Poderes, si fuere el caso. (Protocolo 16.069)
- ☒ Documento de cesión del inventor al solicitante o a su causante.
- ☐ Declaraciones
- ☒ Copia de la solicitud internacional. (Resumen, descripción, reivindicaciones, dibujos)
- ☒ Traducción de la solicitud internacional al castellano. (Resumen, descripción, reivindicaciones, dibujos)
- ☐ Copia del examen preliminar internacional efectuado por la Autoridad Internacional de Examen Preliminar (IPEA).
- ☐ Traducción del examen preliminar internacional efectuado por la IPEA.
- ☐ De ser el caso, copia del contrato de acceso.
- ☐ De ser el caso, documento que acredite la licencia o autorización de uso de conocimientos tradicionales de las comunidades indígenas.
- ☐ De ser el caso, certificado de depósito del material biológico.
- ☒ Arte final 12x12.
- ☐ De ser el caso, información sobre otras solicitudes de patente o títulos obtenidos en el extranjero por el mismo titular o su causante, relacionadas parcial o totalmente con la invención de esta solicitud.
- ☒ De ser el caso, modificaciones efectuadas a la solicitud internacional.
- ☐ Otros.

11 FIGURA CARACTERISTICA



12 Solicito la Concesión de la patente,

NOMBRE: DARIO GARDENAS NAVAS

FIRMA:

C.C. 17.066.629 BTA TP. 1.250
C.S.J. or

or

SUPERINTENDENCIA DE INDUSTRIA Y COMERCIO

SUPERINDUSTRIA Y COMERCIO
Radicacion : 05050638 00000000 Folios: 169
Fecha (AMD): 2005-05-25 11:06:16
Tramite : 011 PCT-PATENT 370 FASE NACI 411 PRESENTA
Dependencia: 2020 DIVISION DE NUEVAS CREACIONES

NIT.: 800176089-2

RECIBO OFICIAL DE CAJA : 05 - 37,299
FECHA : MAYO 19 DE 2005

***** CONSIGNACION *****

DEPOSITANTE	TIPO PAGO	BANCO	CUENTA	No. PAGO	FECHA PGO	Vr. PAGO
CARDENAS Y CARDENAS P.I.	CONSIGNACION	BANCO POPULAR	050-00110-6	38039997	19/05/2005	3,028,000.00

***** CONCEPTO *****

CANT.	RENTISTICO	CONCEPTO	TOTAL CONCEPTO
1	50005-01-01	SOLICITUDES	
		1 TRAMITES DE SOL. DE PATENTE DE IN	443,000.00
		TOTAL :	\$ 443,000.00

SON: CUATROCIENTOS CUARENTA Y TRES MIL PESOS

RESPONSABLE : _____

RECIBO DE CAJA APLICADO AL EXPEDIENTE No. SISTEMA Y METODO
PARA ADMINISTRAR Y COMUNICAR ACTUA
LIZACIONES DE SOFTWARE

CARDENAS & CARDENAS
ABOGADOS

TRATADO DE COOPERACIÓN DE PATENTES

De la OFICINA RECEPTORA

25-APP
ACCT 306474.26
FISCALIA

PCT

NOTIFICACIÓN DEL NUMERO DE SOLICITUD INTERNACIONAL Y FECHA DE ENTREGA

(PCT REGLA 20.5(C))

Para:

URIBE, Mauricio A
Christensen O'Connor Johnson
Kindness PLLC
1420 Fifth Avenue Suite 2800
Seattle, WASHINGTON 98101-2347

Fecha de envío

(día/mes/año)

18 Agosto 2004

Archivo de referencia del solicitante MSFTF 123281		NOTIFICACIÓN IMPORTANTE	
No. de solicitud internacional PCT/US2004/023975	Fecha de solicitud internacional (día/mes/año) Julio 23 2004	Fecha prioritaria(día/mes/año) Diciembre 15 2003	
Solicitante MICROSOFT CORPORATION			
Título del invento SISTEMA Y MÉTODO PARA ADMINISTRAR Y COMUNICAR ACTUALIZACIONES DE SOFTWARE			

- Por medio de la presente se notifica al solicitante que a la solicitud internacional le ha sido asignado el numero de solicitud Internacional y la fecha de archivo indicados arriba.
- se notifica al solicitante que la copia del récord de la solicitud internacional

☒ Fue transmitida a la Agencia Internacional el 18 de Agosto de 2004

☐ no ha sido transmitida a la Agencia Internacional por las razones que se indican abajo y una copia de esta notificación ha sido enviada a la Agencia Internacional*.

☐ porque el visto bueno de la seguridad nacional no se recibido

☐ porque (especificar razón)

* La Agencia Internacional monitorea el tramite de la copia del récord por parte de la oficina Receptora y notificara al solicitante (con el formato PCT/1B/301) de su recepción. Si la copia del récord no se ha recibido el día de vencimiento 14 meses de la fecha de prioridad, la Agencia Internacional notificara al solicitante (Regla 22.1(c)).

- INFORMACIÓN SOBRE EL TRAMITE DE LA LICENCIA INTERNACIONAL Completada por HS.

☐ Licencia adicional para el tramite Internacional no se requiere. Este asunto ya esta cubierto por una licencia expedida o por una solicitud nacional de EU equivalente. Refiérase a esta licencia para información sobre su alcance.

☐ La licencia para tramite internacional no se requiere. 37 CFR 5.11(e)(i) O 37 cfr 5.11(e)(2). sin embargo una licencia puede ser necesaria para tramites adicionales. Ver CFR 5.15(b).

☒ La licencia de tramite internacional ha sido otorgada. 35 U.S.C. 184; 37 CFR 5.11 en: Agosto 13 de 2004.

☐ 37 CFR 5.15(a)

☒ 37CFR 5.15(b)

Nombre y dirección de correo de ISA/

Mail Stop PCT, Comisionado de Patentes
P.O. Box 1450 Alexandria, VA 22313-1450
Facsimil No. 703-305-3230

Oficial Autorizado

Hallet Saunders
Teléfono No. 703-305-3663

SOLICITUD

PCT

Los abajo firmantes solicitan que la presente solicitud internacional sea procesada de acuerdo con el Tratado de Cooperación de Patentes.

PARA USO EXCLUSIVO DE LA OFICINA RECEPTORA
Solicitud Internacional No.
Fecha de Presentación Internacional
Nombre de la Oficina Receptora y "Solicitud Internacional PCT"

Archivo de Referencia del Agente o Solicitante (si se desea)(12 caracteres máximo)
MSFTF123281

Casilla No.1		Título del Invento			SISTEMA Y MÉTODO PARA ADMINISTRAR Y COMUNICAR ACTUALIZACIONES DE SOFTWARE				
Casilla No.2		Solicitante			☐ Esta persona también es inventor				
Nombre y dirección: (apellido seguido por nombre; para una persona jurídica nombre legal completo. La dirección debe incluir código postal y nombre del país. El país de la dirección indicada en esta casilla es el Estado del solicitante (que es, el país) de residencia si no se indica adelante Estado de residencia). MICROSOFT CORPORATION One Microsoft Way Redmond, Washington 98052 Estados Unidos de América					Teléfono no.				
					Fcsimil no.				
					Teleprinter no.				
					No. de registro del solicitante con la oficina				
Estado (que es, el país) de nacionalidad: US					Estado (que es, el país) de residencia: US				
Esta persona es solicitante para los propósitos de		☒ Todos los estados designados		☐ Todos los estados designados excepto los E.U. de América		☐ Solamente los Estados Unidos de América		☐ Los estados indicados en la casilla complementaria	
Casilla No.3		SOLICITANTE(S) Y/O INVENTOR(ES) ADICIONAL(ES)							
Nombre y dirección: (apellido seguido por nombre; para una persona jurídica nombre legal completo. La dirección debe incluir código postal y nombre del país. El país de la dirección indicada en esta casilla es el Estado del solicitante (que es, el país) de residencia si no se indica adelante Estado de residencia). MEULEMANS, Michael Edward 17168 SE 100th Street Renton, WA 98059 Estados Unidos de América					Esta persona es:				
					☐ Solo solicitante				
					☐ Solicitante e inventor				
					☒ Solo inventor (si esta casilla de verificación es marcada, no llene adelante)				
Estado (que es, el país) de nacionalidad: US					Estado (que es, el país) de residencia: US				
Esta persona es solicitante para los propósitos de		☐ Todos los estados designados		☐ Todos los estados designados excepto los E.U. de América		☐ Solamente los Estados Unidos de América		☐ Los estados indicados en la casilla complementaria	
☒ Solicitantes adicionales y/o inventores (adicionales) se indican en una hoja a continuación.									
Casilla No.4		AGENTE O REPRESENTANTE COMÚN; O DIRECCIÓN DE CORRESPONDENCIA							
Por medio de la presente la persona identificada adelante ha sido señalada para actuar en nombre del (los) solicitante (s) ante las Autoridades Internacionales competentes como: ☒ Agente ☐ Representante común									
Nombre y dirección: (apellido seguido por nombre; para una persona jurídica nombre legal completo. La dirección debe incluir código postal y nombre del país. El país de la dirección indicada en esta casilla es el Estado del solicitante (que es, el país) de residencia si no se indica adelante Estado de residencia). URIBE, Mauricio A Christensen O'Connor Johnson Kindness PLLC 1420 Fifth Avenue Suite 2800 Seattle, WA 98101-2347 Estados Unidos de América					Teléfono: 206 682 8100				
					Facsimil: 206 224 0779				
					Teleprinter				
					Registro del Solicitante con la Oficina No. 46.206				
☐ Dirección para correspondencia: marque esta casilla de verificación cuando ningún agente o representante común es/haya sido señalado y el espacio anterior se utiliza en cambio para indicar una dirección especial a la cual la correspondencia se debe enviar.									

Continuación Casilla No. III SOLICITANTE(S) Y/O INVENTOR(ES) ADICIONAL(ES) <i>Si ninguna de las siguientes subcasillas se utiliza, esta hoja no se incluye en la solicitud</i>				
Nombre y dirección: <i>(apellido seguido por nombre; para una persona jurídica nombre legal completo. La dirección debe incluir código postal y nombre del país. El país de la dirección indicada en esta casilla es el Estado del solicitante (que es, el país) de residencia si no se indica adelante Estado de residencia).</i> AVERBUCH, Aaron 2230 Yale Ave. E, #D Seattle, WA 98102 Estados Unidos de América				Esta persona es: ☐ Solo solicitante ☐ Solicitante e inventor ☒ Solo inventor (si esta casilla de verificación es marcada, no llene adelante) No. de Registro del solicitante con la oficina
Estado (que es, el país) de nacionalidad: EU		Estado (que es, el país) de residencia: EU		
Esta persona es solicitante para los propósitos de	☐ Todos los estados designados	☐ Todos los estados designados excepto los E.U. de América	☐ Solamente los Estados Unidos de América	☐ Los estados indicados en la casilla complementaria
Nombre y dirección: <i>(apellido seguido por nombre; para una persona jurídica nombre legal completo. La dirección debe incluir código postal y nombre del país. El país de la dirección indicada en esta casilla es el Estado del solicitante (que es, el país) de residencia si no se indica adelante Estado de residencia).</i> ROBERTS, Jason 103 College Ave. #2 Somerville, MA 02144 Estados Unidos de América				Esta persona es: ☐ Solo solicitante ☐ Solicitante e inventor ☒ Solo inventor (si esta casilla de verificación es marcada, no llene adelante) No. de Registro del solicitante con la oficina
Estado (que es, el país) de nacionalidad: EU		Estado (que es, el país) de residencia: EU:		
Esta persona es solicitante para los propósitos de	☐ Todos los estados designados	☐ Todos los estados designados excepto los E.U. de América	☐ Solamente los Estados Unidos de América	☐ Los estados indicados en la casilla complementaria
Nombre y dirección: <i>(apellido seguido por nombre; para una persona jurídica nombre legal completo. La dirección debe incluir código postal y nombre del país. El país de la dirección indicada en esta casilla es el Estado del solicitante (que es, el país) de residencia si no se indica adelante Estado de residencia).</i> SHOWMAN, Ken 15724 NE 117th Street Redmond, WA 98052 Estados Unidos de América				Esta persona es: ☐ Solo solicitante ☐ Solicitante e inventor ☒ Solo inventor (si esta casilla de verificación es marcada, no llene adelante) No. de Registro del solicitante con la oficina
Estado (que es, el país) de nacionalidad: EU		Estado (que es, el país) de residencia: EU		
Esta persona es solicitante para los propósitos de	☐ Todos los estados designados	☐ Todos los estados designados excepto los E.U. de América	☐ Solamente los Estados Unidos de América	☐ Los estados indicados en la casilla complementaria
Nombre y dirección: <i>(apellido seguido por nombre; para una persona jurídica nombre legal completo. La dirección debe incluir código postal y nombre del país. El país de la dirección indicada en esta casilla es el Estado del solicitante (que es, el país) de residencia si no se indica adelante Estado de residencia).</i> MOHAMMED, Mazhar 127 - 242nd Court SE Sammamish, WA 98074 Estados Unidos de América				Esta persona es: ☐ Solo solicitante ☐ Solicitante e inventor ☒ Solo inventor (si esta casilla de verificación es marcada, no llene adelante) No. de Registro del solicitante con la oficina
Estado (que es, el país) de nacionalidad: EU		Estado (que es, el país) de residencia: EU		
Esta persona es solicitante para los propósitos de	☐ Todos los estados designados	☐ Todos los estados designados excepto los E.U. de América	☐ Solamente los Estados Unidos de América	☐ Los estados indicados en la casilla complementaria
☒ Solicitantes adicionales y/o inventores (adicionales) se indican en una hoja a continuación.				
Formato PCT/RO/101(hoja de continuación)(Enero 2004) Legalstar 2004,Formato PCTREQ				

7

Continuación Casilla No. III SOLICITANTE(S) Y/O INVENTOR(ES) ADICIONAL(ES) Si ninguna de las siguientes subcasillas se utiliza, esta hoja no se incluye en la solicitud				
Nombre y dirección: <i>(apellido seguido por nombre; para una persona jurídica nombre legal completo. La dirección debe incluir código postal y nombre del país. El país de la dirección indicada en esta casilla es el Estado del solicitante (que es, el país) de residencia si no se indica adelante Estado de residencia).</i> DADZIE, Joseph G. 9831-173rd Place NE Redmond,WA 98052 Estados Unidos de América			Esta persona es: ☐ Solo solicitante ☐ Solicitante e inventor ☒ Solo inventor (si esta casilla de verificación es marcada, no llene adelante) No. de Registro del solicitante con la oficina	
Estado (que es, el país) de nacionalidad: EU		Estado (que es, el país) de residencia: EU		
Esta persona es solicitante para los propósitos de	☐ Todos los estados designados	☐ Todos los estados designados excepto los E.U. de América	☐ Solamente los Estados Unidos de América	☐ Los estados indicados en la casilla complementaria
Nombre y dirección: <i>(apellido seguido por nombre; para una persona jurídica nombre legal completo. La dirección debe incluir código postal y nombre del país. El país de la dirección indicada en esta casilla es el Estado del solicitante (que es, el país) de residencia si no se indica adelante Estado de residencia).</i> 			Esta persona es: ☐ Solo solicitante ☐ Solicitante e inventor ☐ Solo inventor (si esta casilla de verificación es marcada, no llene adelante) No. de Registro del solicitante con la oficina	
Estado (que es, el país) de nacionalidad:		Estado (que es, el país) de residencia:		
Esta persona es solicitante para los propósitos de	☐ Todos los estados designados	☐ Todos los estados designados excepto los E.U. de América	☐ Solamente los Estados Unidos de América	☐ Los estados indicados en la casilla complementaria
Nombre y dirección: <i>(apellido seguido por nombre; para una persona jurídica nombre legal completo. La dirección debe incluir código postal y nombre del país. El país de la dirección indicada en esta casilla es el Estado del solicitante (que es, el país) de residencia si no se indica adelante Estado de residencia).</i> 			Esta persona es: ☐ Solo solicitante ☐ Solicitante e inventor ☐ Solo inventor (si esta casilla de verificación es marcada, no llene adelante) No. de Registro del solicitante con la oficina	
Estado (que es, el país) de nacionalidad:		Estado (que es, el país) de residencia:		
Esta persona es solicitante para los propósitos de	☐ Todos los estados designados	☐ Todos los estados designados excepto los E.U. de América	☐ Solamente los Estados Unidos de América	☐ Los estados indicados en la casilla complementaria
Nombre y dirección: <i>(apellido seguido por nombre; para una persona jurídica nombre legal completo. La dirección debe incluir código postal y nombre del país. El país de la dirección indicada en esta casilla es el Estado del solicitante (que es, el país) de residencia si no se indica adelante Estado de residencia).</i> 			Esta persona es: ☐ Solo solicitante ☐ Solicitante e inventor ☐ Solo inventor (si esta casilla de verificación es marcada, no llene adelante) No. de Registro del solicitante con la oficina	
Estado (que es, el país) de nacionalidad:		Estado (que es, el país) de residencia:		
Esta persona es solicitante para los propósitos de	☐ Todos los estados designados	☐ Todos los estados designados excepto los E.U. de América	☐ Solamente los Estados Unidos de América	☐ Los estados indicados en la casilla complementaria
Nombre y dirección: <i>(apellido seguido por nombre; para una persona jurídica nombre legal completo. La dirección debe incluir código postal y nombre del país. El país de la dirección indicada en esta casilla es el Estado del solicitante (que es, el país) de residencia si no se indica adelante Estado de residencia).</i> 			Esta persona es: ☐ Solo solicitante ☐ Solicitante e inventor ☐ Solo inventor (si esta casilla de verificación es marcada, no llene adelante) No. de Registro del solicitante con la oficina	
Estado (que es, el país) de nacionalidad:		Estado (que es, el país) de residencia:		
Esta persona es solicitante para los propósitos de	☐ Todos los estados designados	☐ Todos los estados designados excepto los E.U. de América	☐ Solamente los Estados Unidos de América	☐ Los estados indicados en la casilla complementaria
☐ Solicitantes adicionales y/o inventores (adicionales) se indican en una hoja a continuación.				

Casilla No. V DESIGNACIONES

El archivo de la presente solicitud constituye ante la Norma 4.9(a), de la designación de todos los Estados Contratantes que se rigen por el PCT en la fecha internacional de archivo, para que se otorge toda la protección disponible y, donde sea pertinente, para otorgar tanto la patente regional como la patente nacional.

Sin embargo,

- ☐ DE Alemania no esta designada para ningún tipo de protección
- ☐ KR La republica de Corea no esta designada para ningún tipo de protección
- ☐ RU La Federación Rusa no esta designada para ningún tipo de protección

(Se pueden utilizar las casillas de verificación anteriores para excluir (en forma irrevocable) las designaciones concernientes para evitar la cesación del efecto, bajo la ley nacional, de una solicitud anterior de donde se reivindica la prioridad. Ver las notas a la casilla no. V sobre las consecuencia de los requerimientos de la ley nacional en estos y otros estados.

Casilla No. VI REIVINDICACIONES DE PRIORIDAD

Por medio de la presente se reivindica la prioridad de la(s) siguiente(s) solicitud(es) temprana(s):

Fecha de presentación de la solicitud temprana (día/mes/año)	Número de la solicitud temprana	Donde se encuentra la solicitud temprana:		
		Solicitud nacional: País o miembro del WTO	Solicitud regional:* Oficina regional	Solicitud internacional: Oficina receptora
item(1) Diciembre 15 2003(15.12.03)	10/737.708	EU		
item(2)				
item(3)				

☐ Las reivindicaciones de prioridad se indican en la Casilla Complementaria.

Se solicita a la Oficina receptora preparar y transmitir a la Oficina Internacional una copia certificada de la(s) solicitud(es) temprana(s) (solo si la solicitud temprana fue presentada con la Oficina que para propósitos de esta solicitud internacional es la Oficina receptora) identificada arriba como:

☒ Todos los ítems ☐ ítem (1) ☐ ítem (2) ☐ ítem 3 ☐ ítem (4) ☐ ítem (5) ☐ otros, ver casilla complementaria

* En donde la solicitud temprana es una solicitud ARIPO, indique al menos un estado parte de la Convención de París para la Protección de Propiedad Industrial o un Miembro de la Organización Mundial de Comercio para el cual tal solicitud temprana se presentó (Regla 4.10 (b) (ii)) :.....

Casilla No. VII AUTORIDAD DE BÚSQUEDA INTERNACIONAL

Elección de la Autoridad de Búsqueda Internacional (ISA) (si dos o más Autoridades de Búsqueda Internacionales son competentes para llevar a cabo la búsqueda internacional, indique la autoridad escogida; se puede utilizar el código de dos letras):
ISA/EU.....

Solicitud para utilizar los resultados de la búsqueda temprana; Investigación de tal búsqueda (si una búsqueda temprana se ha llevado a cabo por o solicitado de la Autoridad de Búsqueda Internacional):

Fecha (día/mes/año) Número País (u Oficina Regional)

Casilla No. VIII DECLARACIONES

Las siguientes declaraciones están contenidas en las Casillas Nos. VIII (i) a (v) (marque las casillas de verificación aplicables adelante e indique en la columna de la derecha el número de cada tipo de declaración):

- ☐ Casilla No. VIII(i) Declaración de la identidad del inventor:
- ☐ Casilla No. VIII(ii) Declaración del derecho del solicitante, como la fecha de presentación internacional, para solicitar y otorgarse una patente:
- ☐ Casilla No. VIII(iii) Declaración del derecho del solicitante, como la fecha de presentación internacional, para reivindicar la prioridad de una solicitud temprana
- ☐ Casilla No. VIII(iv) Declaración de inventor (solo para los propósitos de designación de los Estados Unidos de América) :
- ☐ Casilla No. VIII(v) Declaración de descripción no perjudicial o excepciones para falta de novedad

Casilla No. IX LISTA DE VERIFICACIÓN; IDIOMA DE PRESENTACIÓN	
Esta Solicitud Internacional contiene: en forma de papel, el siguiente número de hojas:	
Solicitud 5	
Descripción (excluye listados de secuencias y/o tablas relacionadas con estas) 35	
Reivindicaciones: 8	
Resumen 1	
Dibujos 14	
Sub total de Hojas: 63	
Parte de la lista de secuencias de las descripciones (el número actual de hojas, si se archivaron en papel, así se hayan llenado en computador, ver (c) más abajo. Total número de hojas 63	
(b) ☐ solo en forma legible por computador (Sección 801 (a)(i))	
(i) ☐ listado de secuencias	
(ii) ☐ tablas relacionadas con estas	
(c) ☐ también en forma legible por computador (Sección 801 (a)(ii))	
(i) ☐ listado de secuencias	
(ii) ☐ tablas relacionadas con estas	
Número y tipo de portadores (diskette, CD-ROM, CD-R u otro) en el cual está contenido el	
☐ listado de secuencias.....	
☐ las tablas relacionadas con esta:.....	
(copias adicionales para ser indicadas de acuerdo con los ítems 9(ii) y/o 10(ii), en la columna derecha)	
Esta solicitud internacional se acompaña por el(los) siguiente(s) ítem(s) (marque las casillas de verificación aplicables adelante e indique en la columna derecha el número de cada ítem):	
Número de ítems:	
1. ☒ Hoja de cálculo de tasas	
2. ☐ Original del Poder de abogado por separado	
3. ☐ Original del Poder general de abogado :1	
4. ☐ copia del Poder general de Abogado; número de referencia si existe:	
5. ☐ Declaración que explica falta de firma:	
6. ☐ Documento(s) de prioridad identificado(s) en casilla No. VI como ítem(s):	
7. ☐ Traducción de la solicitud Internacional en (idioma):	
8. ☐ Indicaciones separadas en relación con microorganismos u otro material biológico	
9. ☐ Lista de secuencias en forma legible por computador	
(i) ☐ copia presentada para los propósitos de búsqueda interna de acuerdo con la Regla 13ter solamente (y no como parte de la solicitud internacional):	
(ii) ☐ (solo cuando la casilla de verificación (b)(i) o (c)(i) se marca en la columna de la izquierda) copias adicionales incluyen, cuando aplica, la copia para los propósitos de búsqueda internacional de acuerdo con la Regla 13ter	
(iii) ☐ junto con las declaraciones importantes para la identidad de la copia, o copias con los listados de secuencias mencionados en la columna izquierda	
10. ☐ Tablas en forma legible por computador relacionadas con la lista de secuencias (indique número y tipo de portadores)	
(i) ☐ copia presentada para los propósitos de búsqueda internacional de acuerdo con la Sección 802(802b-cuarto) solamente (y no como parte de la solicitud internacional):	
(ii) ☐ (solo cuando la casilla de verificación (b)(ii) o (c)(ii) se marca en la columna de la izquierda) copias adicionales incluyen, cuando aplica, la copia para los propósitos de búsqueda internacional de acuerdo con la Sección 802(b-cuarto)	
(iii) ☐ junto con las declaraciones importantes para la identidad de la copia, o copias con las tablas mencionadas en la columna izquierda	
11. ☒ Otra (especifique): Carta de transmisión, cheque y tarjeta	
Figura de los dibujos que puede acompañar al Resumen: 12A	
Idioma de Presentación de la Solicitud Internacional: Inglés	
Casilla No. X FIRMA DEL SOLICITANTE, O AGENTE	
<i>Justo a cada firma, indique el nombre de la persona firmante y la capacidad en la cual la persona firma (si tal capacidad no es obvia de la lectura de la solicitud)</i>	
MAURICIO A, Uribe C.	
Agente para la solicitud	
Reg No. 46.206	
1. Fecha de recibo actual de la solicitud internacional propuesta:	
2. Dibujos:	
3. Fecha corregida de recibo actual debido a posterior pero oportuno recibo de papeles o dibujos que completan la solicitud internacional propuesta	
4. Fecha de recepción oportuna de las correcciones requeridas de acuerdo con el Artículo 11(2) PCT:	
5. Autoridad de Búsqueda Internacional (si dos o más son competentes): ISA/	
7. ☐ Tramite de la copia de búsqueda retrasada hasta que la tasa de búsqueda se pague	
6. ☐ Recibido:	
6. ☐ no recibido:	
Fecha de recibo de la copia de registro por la Oficina Internacional:	

SISTEMA Y MÉTODO PARA ADMINISTRAR Y COMUNICAR ACTUALIZACIONES DE SOFTWARE

EL CAMPO DEL INVENTO

El presente invento se relaciona con el software y con las redes de computador y, en particular, el presente invento se relaciona con un sistema y método para administrar y comunicar actualizaciones de software.

ANTECEDENTES DEL INVENTO

La mayoría de productos de software se someten a un proceso de revisión continua para reparar o actualizar características y/o funciones. Cada revisión de software o sus componentes puede requerir de archivos nuevos y/o el reemplazo de archivos existentes con nuevas versiones. Una vez que un vendedor ha aislado un problema en un producto y creado la solución, quisiera poner esa solución en una actualización y hacerla disponible a los clientes. Los vendedores de software tienen un incentivo de negocio para distribuir actualizaciones de software a los cliente tan pronto y tan fácil como sea posible.

Internet es un canal importante para que los clientes obtengan las ultimas actualizaciones de productos de software. El crecimiento explosivo de la Internet ha creado una expectativa común en los clientes que esperan que los productos de software y las actualizaciones estarán disponibles para descargar todos. También es de interés a los vendedores de software promover el uso de Internet para distribuir las actualizaciones, porque reduce sus costos y les permite a los clientes

obtener la solución para un problema que ha sido identificado tan pronto la solución este disponible para descargar. Los portales de los vendedores en Internet pueden estar diseñados para que sea muy fácil descubrir y localizar los archivos de actualización para una aplicación. Los aspectos técnicos de la descarga de archivos prácticamente ha desaparecido para el usuario, y ahora típicamente los maneja el sistema operativo.

En un enfoque convencional el vendedor de software construye un "paquete" de software para descargar. Este paquete de generalmente es un archivo ejecutable auto-extraíble con el programa de instalación y cada una de las actualizaciones del producto incrustadas y comprimidas para hacer el paquete más pequeño. El tamaño del paquete generalmente es la suma del tamaño comprimido de cada archivo que se va a reemplazar mas el tamaño del código de extracción. Una vez se ejecuta, el paquete extrae cada uno de los archivos contenidos a una localidad temporal, luego inicia el programa de instalación para instalar cada archivo a su dirección apropiada en el directorio del sistema. Los archivos que se envían en forma comprimida se descomprimen mientras se instalan. Cualquier archivo con el mismo nombre y en el mismo lugar simplemente se sobrescribe con el archivo de reemplazo.

Mientras que Internet hace posible una amplia y rápida distribución de actualizaciones de software, el ancho de banda limitado en las redes de transmisión ha causado problemas. El solo tamaño de las aplicaciones de software ha hecho que el tamaño de las actualizaciones que se tienen que descargar sea exageradamente grande. Generalmente se reúnen en una sola actualización una variedad de parches para distintos problemas. Sin un vendedor actualiza sus productos de software con regularidad, el tamaño del paquete de actualización cada vez será mas grande porque el vendedor no puede asumir que el usuario ya tiene los archivos de actualizaciones previas. El paquete de actualización puede ser bastante grande porque combina varios archivos completos, incluso cuando los

archivos están comprimidos. Algunas veces, incluso en con los módems más rápidos, la eficiencia en el ancho de banda de la descarga decrece.

El tiempo que se consume en una actualización convencional es un aspecto, claro esta, indeseable. En algunos casos los clientes tienen que pagar conexiones de larga distancia o tiempos de conexión durante estas descargas. Cualquier reducción en tiempo de conexión reducirá el costo monetario para estos clientes. Los vendedores generalmente también tienen algún tipo de costo relacionado con el tamaño de las actualizaciones que proveen, por lo que reducir el tamaño también puede proporcionarles algún tipo de beneficio. Reducir el tamaño de la descarga aumentara el ancho de banda disponible para atender a más clientes con el servidor disponible.

El largo de tiempo que se necesita para bajar una descarga grande también hace vulnerable el proceso a distintos problema de conexión en la red. Hay varias razones por las cuales una sesión de Internet se puede desconectar en forma prematura, que incluyen ruidos en la línea de teléfono, señales de llamada en espera, y ordenes no intencionales. Algunos proveedores de Internet imponen un limite de tiempo en la conexión, limitando el tiempo que un usuario puede estar en línea en una sola sesión. Si, cuando la conexión se corta, el usuario esta bajando un archivo grande, el o ella pueden tener que comenzar de nuevo. La mayoría de sistemas operativos y protocolos de transferencia no permiten que la transferencia se reanude, así que cualquier progreso que se haya hecho se perderá, y la transferencia tendría que comenzar de nuevo. Las oportunidades de fracaso son tan numerosas que el usuario puede llegar a considerar imposible la actualización en línea. Si el tamaño del paquete es demasiado grande, los usuarios pueden llegar a nunca poder descargarlo en forma completa.

Un intento de reducir el tamaño de las actualizaciones de software he incrementar la eficiencia del ancho de banda se relaciona con el uso de parches delta o parches binarios. Uno diestro en el arte apreciara que un parche delta corresponde a un código de software especializado que modifica un archivo

existente cuando se ejecuta por un dispositivo de computación. Porque los parches delta incluyen un código especializado de software, un parche delta único se necesita para cada versión única de un archivo. En cuanto a actualizaciones de software, un servicio de actualización puede transmitir un parche de actualización delta en vez de transmitir el archivo completo de actualización. El parche delta luego se utiliza para modificar el archivo existente en el archivo de actualización.

Aunque el parche delta de actualización puede potencialmente reducir la cantidad de datos requeridos para actualizar un archivo, las estrategias actuales en la utilización de parches delta son deficientes en administrar la selección de archivos delta aplicables en situaciones donde una gran cantidad de versiones del archivo existen. Porque un parche único delta se necesita para cada versión de un archivo, los sistemas típicos de actualización de software pueden frecuentemente necesitar cientos, si no miles de parches únicos delta que correspondan con la versión única de cada archivo. En un enfoque, algunos servicios de actualización que sustentan parches delta transmiten todos los parches delta posibles al dispositivo de computación del cliente. Si embargo, este enfoque aumenta típicamente la cantidad de datos requeridos para implementar una actualización de software en la medida que el numero de posibles parches de actualización delta aumentan. De acuerdo, el numero de parches delta aplicables potencialmente puede crecer hasta tener el mismo tamaño del archivo completo de actualización. En otro enfoque, un servicio de actualización de software en red investiga la maquina del cliente para seleccionar que parches delta pueden ser apropiados para la maquina del cliente. Aunque esto reduce la cantidad de información de parches delta que se transmite, requiere lógica adicional en el software de actualización para investigar la maquina del cliente y seleccionar el parche delta apropiado. El uso de lógica adicional incrementa los recursos del sistema que el servicio debe aportar. Aun mas, este enfoque típicamente previene la utilización de memoria inmediata de la red, como tradicionalmente se logra a través de servidores de Red.

Adicionalmente a estas deficiencias, los sistemas existentes son incapaces de proveer ciertos tipos de actualizaciones de software, tales como drivers de hardware. Como es bien conocido en el arte, actualizaciones especializadas de software, tales como aquellas que pertenecen a drivers de hardware, son difíciles de proveer a los usuarios en distribuciones a gran escala porque la mayoría de actualizaciones especializadas de software solo funcionan en computadores de clientes con un hardware específico. En la mayoría de casos, por ejemplo, si el computador de un cliente obtiene una actualización de hardware incompatible, la instalación de la actualización del driver puede causar un error fatal, o incluso impedir que el computador funcione.

Como se podrá entender de lo antedicho, habiendo mejorado la comunicación de las actualización de software entre un servidor y un numero de clientes existe la necesidad para un sistema y método. Adicionalmente existe la necesidad para un sistema de actualización de software y un método con mecanismos mejorados para permitir a un servicio de actualización dirigirse a distintos tipos de cliente cuando este entregando una actualización especializada.

RESUMEN DEL INVENTO

El invento presente esta dirigido a un sistema y método para administrar actualización de software. Específicamente, el presente invento esta dirigido a un sistema y método que facilita la selección e implementación de las actualización de software mientras minimiza el ancho de banda y los recursos de procesamiento que se necesitan para seleccionar e implementar las actualización de software. De acuerdo con un aspecto del presente invento se provee un sistema de arquitectura de componentes para procesar las actualizaciones de software.

De acuerdo con un aspecto del presente invento, se provee un método para actualizar los programas de software de un servicio de actualización de software a

un dispositivo de computación de un cliente. El método se puede implementar en un sistema de computación, incluyendo un dispositivo de computación de un cliente que este en comunicación con un servicio de actualización de software. De acuerdo con este método, un servicio de actualización de software obtiene una petición de sincronización del dispositivo de computación del cliente. Si el dispositivo de computo del cliente guarda el software de actualización la petición de sincronización incluye un identificador de una actualización de software. El servicio de actualización de software determina si la condición de sincronización tiene un identificador de por lo menos una actualización del software instalado. Si el servicio de actualización del software determina que la petición de sincronización tiene un identificador de por lo menos una de las actualización de software, el servicio de actualización de software selecciona una actualización adicional para comunicar al dispositivo de computo del cliente. La selección de la actualización de software adicional depende de que se cumplan ciertos prerequisites definidos adicional a la actualización del software donde los prerequisites requieren que la petición de sincronización incluya un identificador para por lo menos una de las actualizaciones de software instaladas. De otra manera, si el servicio de actualización de software determina que la petición de sincronización no tiene un identificador de por lo menos una actualización de software instalada, el servicio de actualización de software selecciona un primer nivel de actualización para comunicación con el dispositivo de computación del cliente, donde el primer nivel de actualización del software no incluye un prerequisite. El servicio de actualización de software luego comunica un componente de instrucciones de actualización del software seleccionado del dispositivo de computación del cliente al servicio de actualización de software. El servicio de actualización de software guarda entonces el componente de instrucciones de la actualización de software seleccionado en el dispositivo de computación del cliente como una actualización de software instalada si el dispositivo de cómputo del cliente contiene por lo menos un componente que llena las condiciones de aplicabilidad guardadas en la actualización de software seleccionada.

De acuerdo a otro aspecto del presente invento, se provee un método para comunicar actualizaciones de software de un servicio de actualización de software al dispositivo de cómputo de un cliente. El método puede implementarse en un sistema de computación que incluye al dispositivo de computación de un cliente en comunicación con un servicio de actualización de software. De acuerdo con el método, un servicio de actualización de software tiene una petición de autorización del dispositivo de cómputo del cliente. La petición de autorización contiene un identificador del módulo de autorización de un cliente en el dispositivo de cómputo del cliente. El servicio de actualización de software determina si el dispositivo de cómputo del cliente está asociado a algún grupo objeto. El servicio de actualización de software determina que el dispositivo de cómputo del cliente está asociado con el grupo objeto si el módulo de autorización del cliente indica que el dispositivo de cómputo del cliente está asociado con el grupo objeto. El servicio de actualización comunica una Cookie del servicio de actualización de software al dispositivo de cómputo del cliente si determina que el dispositivo de cómputo del cliente está asociado con el grupo objeto, en donde la Cookie del servidor identifica al grupo objeto. El servicio de actualización de software adquiere una petición para una actualización de software almacenada en el servicio de actualización de software, en donde la petición contiene la Cookie del servidor. En respuesta a la petición que obtiene, el servicio de actualización de software determina si una actualización de software está asociada con el grupo objeto identificado en la Cookie del servidor. Adicionalmente, si el servicio de actualización de software determina que la actualización de software está asociada con un grupo objeto identificado en la Cookie del servidor, el servicio de actualización de software comunica la actualización de software del servicio de actualización de software al dispositivo de cómputo del cliente.

De acuerdo con otro aspecto del presente invento, se provee un método para comunicar estructuras de datos de un servidor al dispositivo de cómputo del cliente. El método se puede implementar en un sistema de computación incluyendo a un dispositivo de cómputo en comunicación con un servidor. De

acuerdo con el método, un servicio de actualización de software determina si el dispositivo de cómputo del cliente está asociado con un grupo objeto si el dispositivo de cómputo del cliente contiene un módulo de autorización indicando que el dispositivo de cómputo del cliente está asociado con un grupo objeto. El servicio de actualización de software comunica una Cookie del servidor al dispositivo de cómputo del cliente si determina que el dispositivo de cómputo del cliente está asociado con un grupo objeto. La Cookie del servidor identifica el grupo objeto. El servicio de actualización de software obtiene una petición para una estructura de datos almacenada en el servidor. La petición contiene la Cookie del servidor. En respuesta a la petición, el servicio de actualización de software determina si la estructura de datos está asociada con el grupo objeto identificado en la Cookie del servidor. Si el servicio de actualización de software determina que la estructura de datos está asociada con el grupo objeto identificado en la Cookie del servidor, el servicio de actualización de software comunica la estructura de datos del servidor al dispositivo de cómputo del cliente.

DESCRIPCIÓN BREVE DE LOS DIBUJOS

Los aspectos anteriores y muchas de las ventajas que acompañan este invento se podrán apreciar mejor si el mismo se comprende en referencia a la siguiente descripción, y cuando se toma en conjunto con los dibujos que la acompañan, en donde:

la figura 1 es un diagrama de bloque de un sistema de actualización de software, que incluye el computador de un cliente y un servicio de actualización que entrega actualizaciones de software de acuerdo con el presente invento;

la figura 2 es un diagrama de bloque de un sistema de actualización de software de la figura 1 que muestra la autenticación de un dispositivo de cómputo de un cliente con un servicio de actualización de acuerdo con el presente invento;

la figura 3 es un diagrama de bloque de un sistema de actualización de software de la figura 1 que muestra la sincronización de actualizaciones disponibles entre el dispositivo de cómputo de un cliente y el servicio de actualización de acuerdo con el presente invento;

la figura 4 es un diagrama de bloque del sistema de actualización de software de la figura 1 que muestra la transmisión de información de actualización de software al dispositivo de cómputo de un cliente de un servicio de actualización de acuerdo con el presente invento;

la figura 5 es un diagrama de bloque de un sistema de actualización de software de la figura 1 que muestra el procesamiento y la selección de información de actualización por el dispositivo de cómputo de un cliente de acuerdo con el presente invento;

la figura 6 es un diagrama de bloque de un sistema de actualización de software de la figura 1 que muestra la fusión de parches delta y la instalación de archivos de actualización del dispositivo de cómputo de un cliente de acuerdo con el presente invento;

la figura 7 es un diagrama de flujo que ilustra una rutina de actualización de software implementada por el dispositivo de cómputo de un cliente-servicio y actualización que te identifica actualizaciones de software que se puede instalar en el dispositivo de cómputo del cliente, de acuerdo con el presente invento;

la figura 8 es un diagrama de protocolo de una rutina de autorización para acceder a actualizaciones selectivas guardadas en el servicio de actualización, de acuerdo con el presente invento;

la figura 9 es un diagrama de bloque de un conjunto de actualizaciones de software ejemplares que muestran una rutina de autorización, de acuerdo con el presente invento;

la figura 10 es un diagrama de protocolo de una rutina de sincronización para comunicar un grupo selecto de actualizaciones de software de un servicio de

actualización de software al dispositivo de cómputo de un cliente, de acuerdo con el presente invento;

la figura 11 es un diagrama de bloque que muestra una selección de una interfase gráfica de usuario ejemplar para mostrar en pantalla una lista de actualización del peso fue que están disponibles para el dispositivo de cómputo de un cliente individual, de acuerdo con el presente invento;

las figuras 12A y 12B son ilustrativas de una subrutina de procesamiento de actualizaciones 1200 implementada por el dispositivo de cómputo de un cliente 110 para descargar e instalar el software solicitado de acuerdo con el presente invento; y

La figura 13 es un diagrama de flujo ilustrativo de una subrutina implementada por el dispositivo de cómputo de un cliente para actualizar componentes de instalación básicos de acuerdo con el presente invento.

DESCRIPCIÓN DETALLADA DE LA IMPLEMENTACIÓN PREFERIDA

Descrito en términos generales, el presente invento está dirigido a métodos y sistemas para administrar actualizaciones de software. El presente invento está dirigido específicamente a un sistema y método para facilitar la selección e implementación de actualizaciones de software mientras se minimiza el ancho de banda y los recursos de procesamiento requeridos para seleccionar e implementar las actualizaciones del software. Las actualizaciones de software, de acuerdo con el presente invento, pueden corresponder a actualizaciones específicas para aplicaciones de software o sistemas operativos. Actualizaciones de software adicional es poder incluir drivers de software o actualizaciones de firmware, tales como el BIOS del sistema. De acuerdo con un aspecto del presente invento, se provee un sistema y componente de arquitectura para procesar las actualizaciones de software. De acuerdo con otro aspecto del presente invento, se suministra un

protocolo de actualización y una interfase para facilitar la autorización y sincronización de las máquinas de los clientes con un servicio de actualización. De acuerdo con otro aspecto del presente invento, se suministra un método para actualizar un componente de instalación y varios archivos de instalación utilizando parches delta. Una persona diestra en el arte apreciará, sin embargo, que aspectos adicionales del presente invento pueden estar disponibles en la presente aplicación. Aún más, uno diestro en el arte apreciará que cada aspecto identificado se puede considerar en forma individual o como parte de un aspecto inventivo común.

El sistema de actualización de software 100 de la figura 1 es un diagrama de bloque que muestra el sistema de actualización 100 de acuerdo con el presente invento. Descrito en forma general, el servicio de actualización 120 almacena y administra la distribución de actualizaciones de software que se comunican y se instalan en el dispositivo de cómputo del cliente 110. Las actualizaciones de software las puede suministrar el servicio de actualización 120 o un número de proveedores de actualizaciones 130.

El dispositivo de cómputo del cliente 110, el servicio de actualización 120, y el proveedor externo de actualizaciones 130 se comunican electrónicamente a través de una red 101. La red puede ser una red de área local (LAN) o una red más grande, como una red de área amplia (WAN) o Internet. Con el uso de software ampliamente conocido, el sistema de actualización de software 100 puede configurarse para intercambiar documentos, comandos, y otro tipo de información entre el dispositivo de cómputo de un cliente 110 y los servidores 121, 122, 123 y 124 del servicio de actualización 120. Como lo podrán apreciar aquellos diestros en el arte y otros, el sistema de actualización de software que se muestra en la figura 1 es un ejemplo simplificado de un sistema adecuado para implementar el presente invento y que el presente invento no está limitado a este ejemplo.

Como se describirá en más detalle mas adelante, una implementación del servicio de actualización 120 consta de un número de servidores. Como se muestra

en la figura 1, el servicio de actualización 120 incluye un servidor de actualizaciones 121 para administrar todo el proceso del servicio de actualización 120 y coordinar los procesos de los servidores 121, 122, 123 y 124 del servicio de actualización 120. El servidor de autorización 122 genera la Cookie de autorización que ha sido solicitado por el cliente y, a su vez las Cookies de autorización se utilizan para generar Cookies del servidor, que le permiten a los computadores de los clientes acceder a las actualizaciones suministradas por el servicio de actualización 120. El servidor de metadatos 123 da información general concerniente a las actualizaciones suministradas por el servicio de actualización 120. El servidor de metadatos 123 le permite al sistema del presente invento identificar actualizaciones específicas para un tipo de computador de un cliente específico o un grupo específico de computadores de clientes. El servidor de descarga 124 suministra uno o más componentes de software para entregar los archivos asociados con la actualización de software suministrada por el servicio de actualización 120.

El proveedor externo de actualizaciones 130 puede incluir a uno o más servidores que distribuyen actualizaciones de software. El proveedor externo de actualizaciones 130 puede asociarse a una entidad que suministra software, actualizaciones de software, u otros datos que se pueden distribuir a grupos de computadores de clientes. Por ejemplo, el proveedor externo de actualizaciones 130 puede estar asociado a un diseñador de software que desea utilizar el servicio de actualización 120 para distribuir actualizaciones a una o más aplicaciones de software. en otro ejemplo, el proveedor externo de actualizaciones 130 puede estar asociado a un sistema de actualización de software 120. El dispositivo de cómputo del cliente 110 puede ser cualquier dispositivo de cómputo que almacene y ejecute aplicaciones de software 114.

El dispositivo de cómputo del cliente 110 puede estar conformado por uno o más distintos productos de computador incluyendo, pero no limitado, a computadores personales (PCs), asistentes digitales personales (PDAs), teléfonos

móviles, busca-personas de dos vías, etc. Como podrá ser apreciado por aquellos de destreza común en el arte y otros, la arquitectura del dispositivo de computación del cliente 110 puede tener una o más formas adecuadas. Por ejemplo, el dispositivo de cómputo del cliente 110 puede incluir una interfase de red para suministrar comunicación con la red 101. La interfase de redes puede configurarse para ser utilizada con cualquier conexión de red alambrada o inalámbrica, puede ser utilizada con cualquier protocolo de comunicación adecuado, tal como TCP(protocolo IP). Adicionalmente, el dispositivo de cómputo 110 puede incluir una unidad procesadora, una pantalla, y una unidad de memoria. La unidad de memoria puede guardar el código del programa necesario para hacer funcionar el dispositivo de cómputo del cliente 110, tal como un sistema operativo 116. Adicionalmente, la unidad de memoria almacena un componente que administra las actualizaciones 112 para controlar y ejecutar los procesos del procesamiento.

El sistema de actualización 100 almacena programas de software que, cuando se ejecutan, implementan el presente invento. Cuando se ejecutan, el sistema de actualización de software 100 almacena, administra y comunica en forma selectiva las actualizaciones de software. Como se describe a continuación en más detalle, entre otros beneficios del presente invento, esta suministrar un mecanismo para definir y seleccionar grupos objeto del dispositivo de computación del cliente que son elegibles para recibir las actualizaciones de software. El presente invento también suministra un mecanismo para descargar archivos de datos asociados con una actualización de software. Con el propósito de ilustrar el presente invento, se suministra una descripción detallada de un ejemplo funcional del presente invento. Describiendo el ejemplo funcional, se hace referencia a actualizaciones de software que se pueden referir a una actualización de una aplicación de software específica, por ejemplo, una actualización de media player versión 6. 0. a media player versión 7.0. Como podrán apreciar aquellos con destreza común en el arte, esta actualización de software puede incluir la comunicación e instalación de un número de archivos de datos asociados con actualización del software. Así, con el propósito

de ilustrar el presente invento, se hace una distinción entre una actualización de software y un archivo individual conteniendo actualizaciones de software.

Ahora, en referencia a la figura 2-6, se describirá una interacción ilustrativa entre los componentes del sistema de actualización de software 100 para actualizar uno o más archivos en el dispositivo de cómputo del cliente 110. En referencia a la figura 2, el servicio de actualización de software se inicia con la transmisión de información de actualización de software por uno o más proveedores externos 130. Como se describió más arriba, el proveedor externo de actualizaciones 130 puede estar asociado con un sistema de actualización de software 100. En forma alternativa, la información de actualización de software puede ser transmitidas por un tercer proveedor de actualizaciones 130.. En una implementación ilustrativa del presente invento, la información de actualización de software puede incluir: Un código de software utilizado para actualizar el archivo, un código de software utilizado para reemplazar el archivo, varias reglas para determinar la aplicabilidad de las actualizaciones del software, y/o mostrar la información que describe la actualización del software. La transmisión de la información de la actualización se puede completar en cualquier momento y no tiene que ser contemporánea con el inicio de la otra actualización de componentes de software ilustrada.

Tan pronto se recibe la información de actualización del proveedor externo 130, el servicio de actualización 120 genera uno o más elementos de datos para facilitar la transmisión de la información de actualización. Los datos pueden incluir un archivo parche de almacenamiento que corresponde al conjunto de software de parches Delta para actualizar distintas versiones del archivo. Los datos también pueden incluir un parche manifiesto de almacenaje que corresponde a una mapeo indexado de una versión particular de un archivo que corresponde a uno Delta encontrado en el archivo parche de almacenamiento. Los datos pueden incluir un archivo auto extraíble que corresponde a información que el agente de actualización utilizará para solicitar e instalar datos de actualización específicos de

software, y como se describirá en más detalle a continuación. Uno diestro en el arte podrá apreciar que la generación del parche de archivo de almacenamiento, almacenamiento de parche manifiesto, y el archivo auto extraíble puede completarse en cualquier momento y no tiene que ser contemporáneo con otras interacciones ilustradas.

Para iniciar la transmisión de información de actualización para el cliente, el dispositivo de cómputo del cliente 110 iniciara una solicitud de autenticación al servicio de actualización 120. En una implementación ilustrativa del presente invento, la solicitud de autenticación corresponde a una interacción del protocolo de actualización entre el dispositivo de cómputo del cliente 110 y el servicio de actualización 120, que será descrito en más detalle mas adelante. Una vez, que se termina la de autenticación, el servicio de actualización 120 trasmite unas Cookies de autenticación al dispositivo de cómputo del cliente 120. En referencia a la figura 3, el dispositivo de cómputo del cliente autenticado 120 inicia la sincronización de actualizaciones disponibles con el servidor de actualizaciones 120. En una implementación ilustrativa del presente invento, la solicitud de sincronización también corresponde a la interacción del protocolo de actualización entre el dispositivo de cómputo del cliente 110 y una servicio de actualización 120, que se describirá en más detalle mas adelante. Tan pronto se termina la sincronización, el dispositivo de cómputo del cliente 110 recibe la información de todo el software de actualización aplicable y la información que describe la actualización. Sin embargo, en una implementación ilustrativa del presente invento, no se ha descargado ningún código de software para iniciar la actualización.

En referencia a la figura 3, en algún momento del proceso de actualización, se recibe una selección de actualizaciones que se van a instalar. En una implementación ilustrativa del presente invento, a un usuario se le puede presentar la información de actualización recibida durante la sincronización y pedirsele que seleccione la actualización apropiada. En forma, alternativa el dispositivo de cómputo del cliente 110 se puede configurar de tal manera que seleccione

automáticamente el software de actualización aplicable. Aún más, el dispositivo de cómputo 110 puede tener reglas que le permiten seleccionar automáticamente una subconjunto de actualizaciones de software disponibles. Aún más, un usuario puede iniciar una selección de actualizaciones comunican 12 con el servicio de actualización 120, a través de una vía como una página web.

En referencia a la figura 4, el componente de administración de actualizaciones 112 da inicio al agente de actualización 118 en el dispositivo de cómputo del cliente, si un agente de actualización no está presente. El agente de actualización 118 solicita la transmisión del paquete de información de actualización de software, como un archivo auto extraíble. El agente de actualización 118 recibe el archivo auto extraíble y ejecuta las actualizaciones para el instalador, como se describirá más abajo. Aún más, el agente de actualización 118 puede solicitar información faltante o corrupta del servicio de actualización 120.

En referencia a la figura 5, una vez que el agente de actualización 118 recibe la información de actualización de software, el agente de actualización 118 hace un inventario de los archivos que están instalados en el dispositivo de cómputo del cliente 110. Basándose en una comparación del inventario del paquete de información de actualización de software, el agente de actualización 118 determina que parche delta, o que otra información de actualización, se requiere para completar la actualización seleccionada. El agente de actualización 118 luego transmite una petición para actualizaciones delta específica. En una implementación del presente invento, la solicitud para actualizaciones de software puede corresponder a una petición directa transmitida a través de una conexión de red, a la que se le referirá como actualización manual. En otra implementación del presente invento, una solicitud de actualización de software puede ser una solicitud de fondo que se transmite sin requerir una acción ostensible por parte del usuario. Nos referiremos a esta implementación como actualización automática.

En una implementación del presente invento, si la actualización de software corresponde a un parche delta, el agente de actualización 118 trasmite la petición al servicio de actualización 120 que identifica el parche delta particular señalado por el manifiesto parche de almacenamiento. En forma alterna, en el caso de que un parche delta no esté disponible o varios parche delta hayan fracasado, el agente de actualización 118 puede iniciar un procedimiento de recaída. El procedimiento de recaída puede incluir la transmisión de una petición de una copia completa de todo el archivo de actualización del archivo parche de almacenamiento. El procedimiento de recaída también puede incluir una petición para la transmisión de una copia completa de todo el archivo de actualización de un paquete auto-contenido.

En una implementación ilustrativa del presente invento, el servidor de descargas 124 del servicio de actualización 120 puede procesar directamente el software de actualización solicitado por el agente de actualización 118. En forma alternativa, la petición también puede ser procesada por cualquier servidor de descargas externo adicional tal como los servidores web que han recibido la solicitud de actualización del parche delta del servicio de actualización 120. Por ejemplo, una corporación puede utilizar un servidor interno para actualizar las máquinas de los clientes. De acuerdo, en esta implementación, la descarga se puede distribuir a un número de servidores de descargas adicionales capaces de servir solicitudes de datos de protocolos de transferencia de hiper texto(HTTP).

En referencia a la figura 6, una vez que se recibe la información de actualización, el agente de actualización 118 une el parche delta con el archivo instalado para generar un archivo actualizado. En forma adicional, el agente de actualización 118 puede validar si la unión actualizó exitosamente el archivo apropiado. Como se describió anteriormente, si un parche del no puede ser validado, el agente de actualización 118 puede solicitar un parche Delta de nuevo o solicitar un archivo de actualización completo luego de varios intentos fallidos.

Una vez que el agente de actualización 118 obtiene el archivo de actualización validado, el archivo se instala en el dispositivo de cómputo del cliente 110.

La figura 7 es un diagrama de flujo de una rutina de procesamiento de software de actualización 700 que ilustra la interacción entre el dispositivo de cómputo del cliente 110 y un servicio de actualización de software 120 de acuerdo con el presente invento. En el bloque 702, el servicio de actualización de software 120 autoriza el acceso al computador del cliente 110. En una implementación ilustrativa del presente invento, la autorización de acceso al computador del cliente puede incluir la generación de una Cookie emitida por el servidor para permitir el acceso al software de actualización que está asociado con un grupo particular de computadores. Una explicación más detallada del proceso de autorización será descrita en referencia a la figura 8.

En el bloque 704, el computador del cliente 110 y el servicio de actualización de software 120 sincronizan la información de actualización. En una implementación ilustrativa del presente invento, el servicio de actualización de software 120 transmite meta datos que describen actualizaciones específicas de software al dispositivo de cómputo del cliente 110. Los metadatos contienen información que describe las actualizaciones de software disponibles para que el usuario seleccione una o más actualizaciones a instalar. Una descripción más detallada del proceso de sincronización se describirá a continuación en referencia a las figuras 9 y 10. En el bloque 706, el dispositivo de cómputo del cliente 110 obtiene una selección de actualizaciones aplicables para descargar. En una implementación ilustrativa del presente invento, la selección de actualizaciones aplicables puede corresponder a la utilización de un número de interfaces únicas para facilitar las selecciones. La selección de interfaces de usuario se describirá el más detalle en referencia a la figura 11.

En el bloque 708, el dispositivo de cómputo del cliente 110 procesa la selección del usuario de actualizaciones de software aplicables e interactúa con el servicio de actualización de software 120. En una implementación ilustrativa del

presente invento, el dispositivo de cómputo del cliente 110 selecciona y solicita uno o más parches de actualización delta aplicables. El agente de actualización 118 en el dispositivo de cómputo del cliente 110 puede entonces procesar los datos solicitados para implementar la actualización de software seleccionado. En el bloque 710, las rutinas 700 terminan.

En referencia a la figura 8, se describirá a continuación un diagrama de protocolo 800 para autorizar el acceso al dispositivo de cómputo del cliente 110 y correspondiente al bloque 702 (figura 7). En una implementación ilustrativa del presente invento, el servicio de actualización de software 120 usa un mecanismo extensible de búsqueda de un objeto para controlar el acceso al dispositivo de cómputo del cliente 110 a actualizaciones y otros softwares. El servicio de actualización 120 incorpora un mecanismo que asocia actualizaciones de software específicas con uno o más grupos objeto del dispositivo de cómputo del cliente 110. En este ejemplo, el servicio de actualización 120 puede definir un grupo objeto del dispositivo de cómputo del cliente 110 que tienen una marca particular y dispositivos de hardware específicos y que limitan la transmisión de software de descarga al grupo objeto.

En una implementación ilustrativa del presente invento, el mecanismo de búsqueda extensible se facilita con el uso de componentes de software ("plug-ins de autorización") que definen la membresía del dispositivo de cómputo del cliente con uno o más grupos objeto. La presencia de una plug-in de autorización en el dispositivo de cómputo del cliente 110 describe si el dispositivo de cómputo del cliente pertenece a un grupo objeto específico presente en el plug-in del cliente. Un grupo objeto, por ejemplo, puede incluir a todos los computadores que tengan un número de identificación del producto válida ("PID") para una aplicación de software particular. En este ejemplo, como se describirá en más detalle abajo con respecto la figura 8, un plug-in 826 se puede instalar en el cliente para leer el PID de un módulo de memoria del dispositivo de cómputo del cliente y entregar el PID al correspondiente plug-in de servidor PID 829. Del correspondiente plug-in ND,

llamado aquí validador PID 829. Una vez que se determina que el PID almacenado en el dispositivo de cómputo del cliente 110 es válido, el servidor genera una Cookie de servidor que indica que el dispositivo de cómputo del cliente 110 es un miembro del grupo objeto que tiene un PID válido. En otro ejemplo, el grupo objeto puede incluir el dispositivo de cómputo de un cliente que está designado como computador de prueba delta.

En una implementación ilustrativa del presente invento, la autorización del servidor 120 y contienen un número de plug-ins de la autorización del servidor que definen un grupo objeto del dispositivo de cómputo del cliente que el servidor de autorización reconocerá. Cada plug-in de autorización del servidor contiene componentes para comunicar datos con el plug-in de autorización del cliente almacenado en el dispositivo de cómputo del cliente 110. En forma similar, cada dispositivo de cómputo de un cliente 110 incluye uno o más plug-ins de autorización que identifican el grupo objeto al cual el cliente pertenece. En una implementación ilustrativa del presente invento, el plug-in de autorización del cliente se puede instalar en cada uno de los dispositivos de cómputo de los clientes durante una instalación o actualización de aplicaciones de software, tales como la instalación y actualización de sistemas operativos. Adicionalmente, el plug-in de autorización del servidor puede instalarse en forma dinámica corre moverse por una administradora que desea controlar el acceso a actualizaciones de software. El plug-in de autorización almacenado en el dispositivo de cómputo del cliente 110 y el servidor de autorización 122 puede ser un plug-in de software, o el el plug-in de autorización puede ser una librería asociada dinámicamente en un código duro.

Como se muestra en la figura 8, el servidor de autorización 122 tienen tres plug-in de autorización de servidor: (1) un primer plug-in de autorización de servidor 828 que define un grupo objeto que incluye a todos los computadores (llamados acá "grupo de Todos los Computadores Objeto"); (2) o el segundo plug-in de autorización de servidor 829 que definen un grupo objeto que incluye los computadores que tienen PID válido (llamado acá "grupo objeto PID"); y (3) un

plug-in de autorización de un servidor 830 que define un grupo objeto que incluye computadores de prueba delta (llamados acá "grupo objeto beta"). Como se muestra también en la figura 8, el dispositivo de cómputo del cliente 110 tiene dos plug-ins de autorización: (1) un primer plug-in de autorización 825 que indica que el dispositivo de cómputo del cliente 110 es un miembro de el grupo objeto de todos los computadores; y (2) un segundo plug-in de autorización del cliente 826 que indica que el dispositivo de cómputo del cliente 110 es un miembro del grupo objeto PID. En este ejemplo, el dispositivo de cómputo del cliente no tiene plug-in de autorización indicando que es miembro del grupo objeto beta. Como podrán apreciarse aquellos de destreza como un en el arte, cada plug-in de autorización para el cliente 825 y 826 se puede configurar para ejecutar una o más funciones en el dispositivo de cómputo del cliente 110 para ayudar en el proceso de validación. Por ejemplo, el segundo plug-in de autorización del cliente 826 puede configurarse para que examine la memoria del dispositivo de cómputo del cliente 110 para verificar y obtener un PID para una aplicación de software instalada.

Como se muestra en la figura 8, la subrutina de autorización 702 comienza cuando el dispositivo de cómputo del cliente 110 comunica una solicitud de configuración 813 al servidor de autorización 122. En una implementación ilustrativa del presente invento, la petición de configuración 803 está formada por un componente de software adecuado y configurado para obtener información que describe el plug-in de autorización almacenado en el servidor de autorización 122. Como podrán apreciar aquellos diestros en el arte, la petición de configuración 803 puede utilizar un método conocido como "GetConfig". En respuesta a haber recibido la petición de configuración 803, el servidor de autorización 122 responde con una comunicación de configuración 804, e incluye la información que identifica a todos los plug-ins de autorización almacenados en el servidor de autorización 122. En una implementación, la respuesta de configuración 804 incluye una matriz de cuerdas que identifica y describe a todos los plug-in de autorización almacenados en el servidor de autorización 122. En el presente ejemplo, la respuesta de configuración 804 incluye información que identifica el primer plug-in

de autorización del servidor 828, el segundo plug-in de autorización del servidor 829, y un tercer plug-in de autorización del servidor 830.

En el bloque 805, el dispositivo de cómputo del cliente 110 genera una o más Cookies de autorización en respuesta a haber recibido la respuesta de configuración 804. En el proceso del bloque 805, el dispositivo de cómputo del cliente 110 genera una Cookie de autorización para cada par de plug-ins que concuerdan en el cliente y el servidor. Por lo tanto, en el ejemplo presente, el primer plug-in de autorización del cliente 825 genera una primera Cookies de autorización asociada con el grupo objeto Todos los Computadores porque el primer plug-in de autorización del cliente 825 y el primer plug-in de autorización del servidor 828 están ambos asociados con el grupo objeto Todos los Computadores. Adicionalmente, el segundo plug-in de autorización del cliente 826 genera una segunda Cookies de autorización asociada con el PID del grupo objeto por qué el segundo plug-in de autorización del cliente 826 y el segundo plug-in de autorización del servidor 829 están ambos asociados con un grupo objeto PID. Una tercera Cookies de autorización no se genera puesto que el dispositivo de cómputo del cliente 110 no tiene un plug-in de autorización indicando que es miembro del grupo objeto beta.

Como también lo podrán apreciar aquellos diestros en el arte, una implementación del bloque de proceso 805 puede incluir el uso del método de software conocido en el arte como "GetAuthCookie". También se podrá apreciar que la generación de cada Cookie de autorización puede involucrar procesos adicionales. Por ejemplo, el segundo plug-in de autorización del cliente 826 puede estar configurado para examinar la información almacenada en el registro del sistema del cliente para obtener un PID e incluir el PID en la Cookie de autorización. En otros ejemplos, el proceso del bloque 805 puede incluir un proceso para comunicarse con otros computadores o dispositivos. Por ejemplo el plug-in de autorización del cliente puede comunicarse con un dispositivo, tal como una tarjeta de sonido, un escáner, una tarjeta de video, etc. para obtener el

modelo y la marca del dispositivo. En otros ejemplos no limitantes, el plug-in de autorización del cliente se puede comunicar con un dispositivo de seguridad, como un lector de huellas digitales, para obtener información que describe al usuario.

En general, el plug-in de autorización del cliente puede leer información de configuración de cualquier componente del dispositivo de cómputo del cliente 110 o de cualquier otro dispositivo conectado con el dispositivo de cómputo del cliente 110. En otro ejemplo, un plug-in de autorización del cliente se puede configurar para ser utilizado por una o más interfases de aplicación de programas públicos o privados (API) para reunir y codificar la información del cliente que será validada por el plug-in del servidor correspondiente. En este ejemplo, el plug-in que valida el PID 826 usa un API privado para codificar el PID del cliente para pasarle el PID codificado al servidor para validación y decodificación. En otras implementaciones, otros plug-ins de autorización del cliente utilizan medidas biométricas, tales como lectores de huellas digitales o archivos de voz, para emitir unas Cookies de autorización que se le pasan al servidor para validación. En otro ejemplo, el plug-in de autorización del cliente puede llamar a un servicio Web o cualquier otro servicio para comunicar credenciales de autorización o cualquier otro tipo de datos al servidor de autorización 122.

En una implementación ilustrativa del presente invento, cada Cookies de autorización incluye una cuerda que identifica el grupo u objeto asociado. Por ejemplo, una cuerda puede indicar que una Cookie de autorización particular está asociada con el PID del grupo objeto. Cada Cookies de autorización también incluye una sección de datos para comunicar datos entre un cliente y un servidor. Por ejemplo, una Cookies de autorización asociada con el PID del grupo objeto puede tener una sección de datos que contienen un PID actual. Como lo podrán apreciar aquellos de destreza común en el arte, la sección de datos puede contener cualquier tipo de datos que estén almacenados en cualquier formato, como una matriz de bits. Por ejemplo, si el plug-in del cliente y del servidor

requieren, pueden transmitir llaves públicas o privadas, tales como secciones de datos codificados de una o más Cookies de autorización.

Una vez que el dispositivo de cómputo del cliente 110 genera una Cookies de autorización para cada par de plug-ins de autorización correspondientes al cliente y al servidor, el dispositivo de cómputo del cliente le transmite la Cookies de autorización generada al servidor de autorización 122. Como se muestra en la figura 8, el dispositivo de cómputo del cliente 110 transmite una Cookies de autorización en la petición de Cookie 806. La petición de Cookie 806 incluye cualquier formato adecuado para comunicar una matriz de Cookies de autorización generada en el proceso del bloque 805. Una implementación de ésta parte del método de autorización 702 puede incluir el uso del método de software generalmente conocido en el arte como "GetCookie".

En una implementación, la petición de Cookie 806 también incluye otras Cookies de autorización del servidor guardadas en la memoria del dispositivo de cómputo del cliente 110. Como se podrá entender de la siguiente descripción, la memoria del dispositivo de cómputo del cliente 110 puede guardar Cookies de autorización del servidor antiguas que se crearon en ejecuciones previas de la rutina de autorización 700. Al proveer la Cookie de autorización del servidor en la petición de Cookie 806, el dispositivo de cómputo del cliente 110 podrá mantener el acceso privilegiado que le fue concedido en ejecuciones previas de la subrutina de autorización 702. En el ejemplo presente, como no hay Cookies de autorización del servidor guardadas por el cliente, la petición de Cookie 806 incluye una Cookie de autorización asociada con el grupo objeto Todos los Computadores y la segunda Cookie de autorización asociada con el PID del grupo objeto.

Luego, como se muestra en el bloque 807, en respuesta a haber recibido la petición de Cookie 806, el servidor de autorización 122 genera una Cookie de servidor. En una implementación, para cada una de las Cookies de autorización recibidas, una llamada se hace al plug-in del servidor de autorización apropiado para generar datos de Cookie del servidor. La Cookie del servidor generada para

0 cada plug-in de autorización del servidor incluye un identificador para cada grupo de objetos identificado en la Cookie de autorización recibida. En el ejemplo presente, puesto que la petición de Cookie 806 incluye la primera Cookie de autorización asociada con el grupo objeto Todos los Computadores y la segunda Cookie de autorización asociada con el PID del grupo objeto, el servidor de autorización 122 genera una Cookie de servidor que tiene un identificador para estos grupos objeto respectivos. En el servidor de autorización 122, la Cookie de servidor se combina con los datos de las Cookies del servidor antiguas, si la Cookie del servidor antiguas se recibe en la petición de Cookie 806, para generar una nueva Cookie de servidor. En una implementación, la nueva Cookie del servidor se codifica con el uso de un método de codificación público disponible, tal como el Triple DES.

0 En una implementación ilustrativa del presente invento, la Cookies del servidor puede incluir información codificada que identifica uno o más de los grupos objeto asociados. Adicionalmente, las Cookies del servidor puede incluir datos de vencimiento, que se almacenan en un formato claro de texto y en un formato codificado. Los datos de vencimiento almacenados en el formato de texto claro son utilizados por el dispositivo de cómputo del cliente 110 para monitorear el vencimiento de las Cookies del servidor. Los datos de vencimiento almacenados en forma codificada son utilizados por el servicio de actualización 120 para determinar si el dispositivo de cómputo del cliente 110 está autorizado para recibir actualizaciones asociadas con un grupo objeto particular. En una implementación, los datos de vencimiento de las Cookies del servidor aplican a todo el grupo objeto identificado en la Cookie del servidor. En forma alternativa, adicional al tiempo de vencimiento que se le aplica a todas las Cookies del servidor, las Cookies del servidor pueden incluir una variedad de datos de vencimiento, cada uno de los cuales aplica a un grupo objeto individual. Como podrán apreciar aquellos de destreza común en el arte, cada Cookie del servidor puede incluir datos adicionales. Por ejemplo, una Cookies de servidor puede estar configurada para

almacenar la información del estado de un cliente, tal como una marca de tiempo de la última ejecución de la subrutina de autorización 702.

Una vez que se ha generado la Cookies de autorización del servidor 809 el servidor de autorización 122 se comunica con el dispositivo de cómputo del cliente 110. Luego, como se muestra en el bloque 811, la Cookies del servidor se almacenan en la memoria del dispositivo de cómputo del cliente 110. Cuando el dispositivo de cómputo del cliente 110 determinar que por lo menos uno de los componentes de la Cookies del servidor a vencido, el dispositivo de cómputo del cliente puede volver a ejecutar el método de autorización 702 para recibir Cookies nuevas del servidor. Como se mencionó anteriormente, en cada ejecución subsiguiente del método de autorización 702, el dispositivo de cómputo del cliente 110 le puede comunicar sus Cookies de servidor almacenadas al servidor de autorización 122 en la petición de Cookie 806. En una implementación, el cliente no tiene que enviar la petición 803 al menos de que el servidor le informe al cliente que la configuración del servidor ha cambiado, por ejemplo, un nuevo plug-in de autorización se ha añadido.

De acuerdo con otro aspecto del presente invento, el servicio de actualización de software 120 puede proveer una subrutina de sincronización para sincronizar información de actualización entre el servidor de metadatos 123 y el dispositivo de cómputo del cliente 110. Con el uso de una jerarquía de actualización de software única, la subrutina de sincronización puede efectivamente identificar actualizaciones específicas que aplican a un dispositivo de cómputo de un cliente particular. Adicionalmente, con el uso de Cookies generadas por el servidor en una subrutina 702, la subrutina de sincronización puede permitir el acceso selectivo a actualizaciones asociadas con grupos objetos específicos.

De acuerdo con una implementación ilustrativa del presente invento, cada actualización de software tiene tres componentes: (1) un componente de instrucciones, (2) un componente de datos localizado; y (3) un componente de datos. Como lo podrá apreciar una persona con destreza común en el arte, cada

actualización puede tener uno o más de los componentes descritos anteriormente. Por ejemplo una actualización puede contener un componente de instrucciones, un componente de datos localizados, y un componente de flujo de datos. En otro ejemplo, una actualización puede contener únicamente un componente de instrucciones para aprobar una o más condiciones en el dispositivo de cómputo del cliente. Los componentes varios de una actualización de software se describen en más detalle a continuación.

Descrito en forma general. El componente de instrucciones contiene dos sub-componentes: (1) una regla de aplicabilidad que define una o más condiciones que se han de poner a prueba en el dispositivo de cómputo del cliente 110; y (2) una serie de requisitos que identifica una o más de las actualizaciones que se requieren para una instalación apropiada de una actualización individual. Como se describe más abajo, la regla de aplicabilidad puede definir un número de condiciones relacionadas con un computador, y cada una de estas condiciones puede estar asociada a otras condiciones con el uso de cualquier operador lógico. Por ejemplo, el componente de instrucciones puede incluir una regla de aplicabilidad para determinar si un computador tiene una versión particular de Windows instalada. También, se describe más abajo, el conjunto de prerrequisitos que pueden identificar una o más actualizaciones instaladas previamente. Por ejemplo, como se describe a continuación en más detalle, en referencia a la figura 9, una actualización individual puede tener un prerrequisito que enumera las otras actualizaciones requeridas para una instalación apropiada de una actualización individual. En otros ejemplos, como se muestra en la figura 9, el conjunto de prerrequisitos puede incluir el uso de operadores lógicos para definir reglas más complejas de prerrequisitos .

El componente de instrucciones también tiene un código, como puede ser una bandera Booleana, que indica si hay otras actualizaciones que dependen de una actualización particular. Por razones ilustrativas, una actualización se considera una actualización LEAF si no hay otras actualizaciones que dependen de una

actualización particular. La bandera Booleana que se usa para indicar si una actualización es LEAF se actualiza dinámicamente por el servidor de metadatos 123 en la medida que actualizaciones relacionadas se añadan o se remuevan.

El componente de datos localizado de cada actualización incluye información general que describe la actualización. Por ejemplo, el componente de datos localizado puede incluir información que describe las características y beneficios de la actualización. El componente de datos localizado puede también incluir una descripción de texto sobre los procedimientos a seguir en la instalación de la actualización. Adicionalmente, el componentes de datos localizado puede incluir más datos o información relacionada con la actualización. Por ejemplo, los datos localizados pueden indicar que una actualización es de alta prioridad. En otro ejemplo, los datos localizados pueden entregar mensajes de instalación especiales como un mensaje que indica si una actualización no se puede instalar con otras actualizaciones de software. La información localizada puede estar en un formato que permite mostrar en pantalla a un usuario la información que contiene .

El componente de datos para cada actualización incluye uno o más flujo de datos binarios de la actualización. En una implementación, el componente de datos de cada actualización puede estar asociado con uno o más archivo de datos, tales como, archivos ejecutables, un documento, una biblioteca vinculada, etc. como se describe en más detalle más abajo, cada actualización puede estar asociada con una combinación de archivos de datos, cada uno de los cuales facilita la actualización actual, instalación, o modificación de software usado por el cliente. Por ejemplo, la instalación de una actualización puede facilitarse con el uso de un solo archivo CAB que incluye toda la información necesaria para completar la actualización seleccionada. En forma alternativa, la instalación de la actualización se puede facilitar con el uso de un número de actualizaciones individuales utilizadas para actualizar uno o más archivos almacenados en el dispositivo de cómputo del cliente.

En una implementación ilustrativa del presente invento, las actualizaciones de software pueden estar organizadas en una jerarquía que permite la distribución controlada de actualizaciones de software. Como se describe en forma general, la jerarquía de actualizaciones define las relaciones entre las actualizaciones y específicamente indica que actualizaciones dependen de otras actualizaciones. Con propósito ilustrativo, un ejemplo de un conjunto de actualizaciones se muestra en la figura 9. Como se ilustra, un conjunto de actualizaciones de prueba 900 incluye un conjunto básico de actualizaciones 901, un segundo conjunto de actualizaciones 902, y un tercer conjunto de actualizaciones 903. En general, cada actualización en el conjunto básico de actualizaciones 901 no tiene un prerequisite que requiera la instalación de otras actualizaciones. Sin embargo, la sexta actualización 921 contiene un prerequisite que requiere la instalación de la primera actualización 911, la segunda actualización 912, y la tercera actualización 913. La séptima actualización 912 contiene un prerequisite que requiere la instalación de la cuarta actualización 914. La octava actualización 931 contiene un prerequisite que requiere la instalación de la sexta actualización 921 y de la quinta actualización 915. Por lo tanto, la octava actualización 931 también requiere la instalación de la primera actualización 911, la segunda actualización 912, y la tercera actualización 913. Para ilustrar el presente invento, se da en la actualización de un conjunto de actualizaciones 900 están asociadas con el grupo objeto todos los computadores y el grupo objeto PID.

También se muestra en la figura 9, y como se describe en más detalle, cada actualización contiene una regla de aplicabilidad que especifica las condiciones para la instalación del actualización. Por ejemplo, la primera actualización 911 requiere la instalación de una versión en inglés del sistema operativo. La segunda actualización 912 requiere la instalación de Windows® XP. versión SP1. En otro ejemplo, la sexta actualización 921 requiere la instalación de un parche de software referido como parche 1 XP, de esta manera el dispositivo de cómputo del cliente 110 no instalará las actualizaciones si las reglas de aplicabilidad no se han cumplido.

La figura 9 también nos muestra una sección del componente de instrucciones que indica si además actualizaciones que dependen de una actualización particular. A esta sección se le refiere como LEAF, que puede estar en la forma de un valor Booleano que indica que la actualización de software es la última actualización de una serie de actualizaciones relacionadas. Con motivo de ilustrar el invento, una actualización particular es una actualización LEAF sin ninguna otra actualización enumera esa actualización particular como prerequisite. Como se muestra una figura 9, la séptima actualización 922 y la octava actualización 931 son las únicas actualizaciones LEAF.

La figura 10. nos muestra un diagrama de protocolo de una subrutina de sincronización 704 (figura 7) formada de acuerdo con el presente invento. Descrita en forma general, la subrutina de sincronización 704 comunica entre el dispositivo de cómputo del cliente 110 y el servidor, el componente de instrucciones en forma selectiva de ciertas actualizaciones, tal como, el servidor de metadatos 123, para identificar actualizaciones que se puedan aplicar al dispositivo de cómputo del cliente 110. Como se muestra en la figura 10, para iniciar una actualización, el dispositivo de cómputo del cliente 110 primero procesa las actualizaciones instaladas y comunica una petición de sincronización 1051 al servidor de meta datos 123 solicitando una o más actualizaciones disponibles para el cliente. En respuesta a haber recibido la petición de sincronización 1051, el servidor de meta datos 123 devuelve un número de actualizaciones al dispositivo de cómputo del cliente 110. Como se entenderá de la siguiente descripción, el dispositivo de cómputo del cliente 110 procesa datos almacenados localmente antes de comunicar la solicitud de sincronización 1051.

El dispositivo de cómputo del cliente 110 procesa el componente de instrucciones para cada actualización recibida para determinar si la condición definida en la regla de aplicabilidad se pueden satisfacer. Si la condición definida en una actualización individual se satisface, con motivo de ilustrar la subrutina de sincronización 1050, la actualización individual se "instalará", y la actualización

instalada se guarda en un primer componente de la memoria intermedia de la actualización del cliente. Por otro lado, si las condiciones definidas en una actualización individual no se satisfacen, la actualización individual se considera "fracasada" y la actualización fracasada se guarda en un segundo componente de la memoria intermedia de la actualización del cliente. En esta descripción de la subrutina de sincronización 1050, si una actualización se instala, se puede asumir que el prerequisite y las condiciones de aplicabilidad se han cumplido. La instalación de una actualización con el propósito de escribir estas subrutinas no necesariamente significa que los archivos de datos asociados con la actualización estén actualmente instalados en el dispositivo de cómputo del cliente 110.

En una implementación, los dos componentes de la memoria intermedia de la actualización del cliente se pueden usar para categorizar las actualizaciones recibidas. El primer componente se utiliza para guardar actualizaciones LEAF instaladas, y un segundo componente se utiliza para guardar las otras actualizaciones recibidas por el cliente, por ejemplo, las actualizaciones que no se instalaron. El segundo componente en la memoria intermedia de la actualización también incluye el almacenamiento de todas las actualizaciones LEAF. Como se describe en más detalle a continuación, actualizaciones guardadas en la memoria intermedia pueden ser transmitidas y procesadas por el servidor de metadatos 123 para identificar otras actualizaciones relacionadas que están disponibles para instalar en el dispositivo de cómputo del cliente 110.

Regresando a la figura 10, ahora se describirá en detalle la petición de sincronización, que se ilustra como los items 1051, 1055, y 1060. Como lo podrán apreciar aquellos de destreza común en el arte. La petición de sincronización puede iniciarse por un número de dispositivos diferentes, procesos, aplicaciones, comandos iniciados por el usuario, solicitando una actualización. La solicitud de sincronización la puede iniciar un usuario solicitando una lista de actualizaciones, una actualización automática iniciada por el agente del cliente, o cualquier otro componente de software que solicita información del servidor de metadatos 123 o

del servicio de actualizaciones 120. En una implementación, la solicitud de sincronización incluye una Cookie de autorización del servidor, tal como, las Cookies de autorización del servidor generadas por la rutina de autorización 702. La utilización de Cookies del servidor permite que el servidor determine si el cliente es un miembro de uno o más de los grupos objeto.

Cada solicitud de sincronización puede también incluir identificadores para cada actualización guardada en la memoria intermedia de la actualización del cliente. Específicamente, si una o más actualizaciones están guardadas en la memoria intermedia de la actualización, la solicitud de sincronización incluye un primer componente que tiene identificadores para actualizaciones no-LEAF; y un segundo componente que tiene identificadores para todas las demás actualizaciones, tales como actualizaciones LEAF, actualizaciones fracasadas, y otras actualizaciones que no están instaladas. Los identificadores de actualizaciones pueden estar en cualquier formato incluyendo, pero no limitado a una matriz de integrales. De otra manera, si no hay actualizaciones guardadas en la memoria intermedia de la actualización del cliente, la solicitud de sincronización no se configura con un identificador de actualizaciones. Cuando una solicitud de sincronización no está configurada con un identificador de actualizaciones, la solicitud de sincronización da una indicación de que el sistema de cómputo del cliente 110 no tiene ninguna actualización en la memoria intermedia.

Como se muestra en la figura 10, una primera solicitud de sincronización 1051 se comunica del dispositivo de cómputo del cliente 110 a un servidor de metadatos 123. En el ejemplo presente, la actualización en la memoria intermedia del cliente no contiene ninguna actualización, dado que esta es la primera ejecución del método. Así, la primera solicitud de sincronización 1051 no contiene un identificador para una actualización que está en la memoria intermedia. En respuesta a haber recibido la solicitud de sincronización, como se muestra en el bloque 1052, el servidor de meta datos 122 determina si la solicitud de sincronización contiene por lo menos un identificador de actualización. Si

determina que la solicitud de sincronización no incluye un identificador de actualizaciones, el servidor de meta datos 123 responde seleccionando el primer nivel de actualizaciones para comunicar al dispositivo de cómputo del cliente 110. Como se describió anteriormente, el primer nivel de actualizaciones puede incluir cualquier actualización que no tenga un prerequisite identificando otras actualizaciones.

De otra manera, si se determina que la solicitud de sincronización contiene por lo menos un identificador de actualizaciones, el servidor de metadatos 123 examina los prerequisites de las actualizaciones guardadas en el servidor para seleccionar actualizaciones adicionales para ser enviadas al cliente. En una implementación, el servidor de metadatos 123 selecciona las actualizaciones que cumplieron los prerequisites. El servidor utiliza las actualizaciones del primer componente para examinar los prerequisites de la solicitud de sincronización, que incluye identificadores de actualizaciones no-LEAF que están instaladas en el cliente.

Adicionalmente a seleccionar actualizaciones que hayan cumplido los prerequisites, el servidor también usa las actualizaciones identificadas en el segundo componente de la solicitud de sincronización para filtrar las actualizaciones seleccionadas. Específicamente, actualizaciones desinstaladas, actualizaciones LEAF y actualizaciones identificadas como fracasadas en el segundo componente de la solicitud de sincronización son usadas para filtrar una o más de las actualizaciones seleccionadas. Este rasgo del invento presente permite al sistema y método del presente invento evitar transmisiones múltiples de actualizaciones guardadas en el servidor de metadatos 123.

Regresando al presente ejemplo, como la primera solicitud de sincronización 1051 no incluye un identificador de actualizaciones, el servidor de meta datos 123 selecciona el nivel básico de actualizaciones 901 para comunicar al dispositivo de cómputo del cliente 110. En referencia al conjunto de actualizaciones que se utilizan como ejemplo y que se muestran en la figura 9, el nivel básico de

actualizaciones 901 incluye las actualizaciones referenciadas en 911, 912, 913, 914, y 915.

En el bloque de procesamiento 1052, el servidor de metadatos 123 examina también a la Cookie de autorización del servidor contenida en la petición de sincronización 1051 para identificar los grupos objeto que están asociados con el dispositivo de cómputo del cliente 110. El servidor de metadatos 123 examina, también, los grupos objeto de las actualizaciones seleccionadas en el proceso del bloque 1052. El proceso del bloque 1052 luego filtra todas las actualizaciones seleccionadas que no están asociados con un grupo objeto identificado en la Cookie de autorización recibida del servidor. En el ejemplo presente, puesto que todas las actualizaciones seleccionadas 911, 912, 913, 914, y 915 están asociadas con el PID y con todos los grupos objeto de Todos los Computadores, todas las actualizaciones seleccionadas se envían al dispositivo de cómputo del cliente 110.

El servidor de metadatos 123 comunica entonces la actualización seleccionada en una respuesta de sincronización 1053 al dispositivo de cómputo del cliente 110. En general, cada respuesta de sincronización incluye un componente de instrucciones para cada actualización enviada por el servidor 120. Así, en el ejemplo presente, la primera respuesta de sincronización 1053 incluye un componente de instrucciones para las actualizaciones referenciadas como 911, 912, 913, 914, y 915. En una implementación, cada respuesta de sincronización no incluye el componente de datos localizado ni el componente de datos de cada actualización.

Luego, como se muestra en el bloque 1054, el dispositivo de cómputo del cliente 110 procesa el componente de instrucciones para cada una de las actualizaciones recibidas para determinar si las condiciones definidas en la regla de aplicabilidad se pueden satisfacer. En referencia, de nuevo, a la figura 9, el dispositivo de cómputo del cliente 110 procesa el componente de instrucciones de la actualización recibida 911-915. Con el propósito de ilustrar el presente invento, se da por hecho que el sistema operativo del dispositivo de cómputo del cliente

110 es una instalación de Windows® versión XP SPI en inglés. Se da por hecho también que el dispositivo de cómputo del cliente 110 es un PC DELL corriendo a 32 bits en un procesador XY 86. Así, en el procesamiento del componente de instrucciones del conjunto de actualizaciones ejemplares, el dispositivo de cómputo del cliente 110 determino que las condiciones definidas en la primera actualización 911 se cumplirían porque el computador contienen un OS en inglés. Las condiciones definidas en la segunda actualización 912 también se cumplirían porque el sistema operativo es Windows® versión XP SPI. Las condiciones definidas en la tercera actualización 913 se cumplirían porque el dispositivo de cómputo del cliente 110 corre en un procesador X 86. Las condiciones definidas en la quinta actualización 915 se cumplirían porque el dispositivo de cómputo del cliente 110 es un PC Dell. Como resultado la primera actualización 911, la segunda actualización 912, la tercera actualización 913, y la quinta actualización 915 se guardan en el primer componente de la memoria intermedia de la actualización del cliente. La condición definida en la cuarta actualización 914 no se cumpliría porque el dispositivo de cómputo del cliente 110 no corre en un procesador de 64 bits X 86. Por lo tanto la cuarta actualización 914 se considera una actualización fracasada y se guarda en el segundo componente de la memoria intermedia de la actualización del cliente.

Regresando a la figura 10, en el procesamiento del bloque 1054, el dispositivo de cómputo del cliente 110 determina también si se requiere una petición de sincronización subsiguiente. En una implementación, se determina que una petición de sincronización subsiguiente se requiere si por lo menos una de las actualizaciones recibidas indica que no es una actualización LEAF. En el presente ejemplo, se determina que una petición de sincronización subsiguiente se requiere porque todas las actualizaciones recibidas no son actualizaciones LEAF. Por lo tanto, el dispositivo de cómputo del cliente 110 envía una petición de sincronización subsiguiente 1055 al servidor de meta datos 123.

6 Como se resumió anteriormente, una petición de sincronización incluye identificadores para cada actualización guardada en la memoria intermedia de la actualización del cliente. Por lo tanto, en el ejemplo presente, la petición de sincronización subsiguiente 855 incluye un primer componente de datos indicando que la primera actualización 711, la segunda actualización 712, la tercera actualización 713, y la quinta actualización 715 están instaladas en el cliente. Adicionalmente, la petición de sincronización subsiguiente 855 incluye un segundo componente de datos indicando que la cuarta actualización 711 no se instaló exitosamente en el cliente.

En respuesta a haber recibido la petición de sincronización subsiguiente 1055, como se resumió anteriormente, el servidor de metadatos 123 determina si la petición de sincronización subsiguiente 1055 contienen por lo menos un identificador de actualizaciones, el servidor de metadatos 123 examina los prerequisites de todas las actualizaciones almacenadas para seleccionar actualizaciones adicionales que se le han de enviar al cliente.

6 Con referencia de nuevo al ejemplo presente, en un bloque de procesamiento 856, el servidor de metadatos 123 seleccionaría la sexta actualización 921 porque sus prerequisites se cumplen. Específicamente, como se muestra en la figura 9, la sexta actualización 921 se selecciona para ser transmitida al dispositivo de cómputo del cliente 110 porque sus prerequisites, que requieren la instalación de la primera actualización 711, la segunda actualización 912, la tercera actualización 913, se cumplen. La séptima actualización 912 y la octava actualización 931 no se seleccionarían para transmitir al cliente porque sus prerequisites no se cumplen. Específicamente, la petición de sincronización 1055 no contenía un identificador para la cuarta actualización 914, que es prerequisite para la séptima actualización 922. Adicionalmente, la petición de sincronización 855 no contenía un identificador para la sexta actualización 921, que es un prerequisite para la octava actualización 931.

Regresando a la figura 10, la subrutina de sincronización 1050 continúa transmitiendo la actualización seleccionada en una respuesta subsiguiente 1057 del servidor de metadatos 123 al dispositivo de cómputo del cliente 110. En referencia al ejemplo presente, se le comunica al cliente 110 la respuesta subsiguiente 1057 que incluye información relacionada con la sexta actualización 721 en una respuesta subsiguiente 1057.

Una vez que el dispositivo de cómputo del cliente 110 recibe la respuesta subsiguiente 1057, procesa el componente de instrucciones de la respuesta subsiguiente 857. Similar al proceso del bloque 854, el dispositivo de cómputo del cliente 110 procesa el componente de instrucciones para cada una de las actualizaciones recibidas para determinar si las condiciones definidas en las reglas de aplicabilidad se cumplen. En el ejemplo presente, si se da que el PARCHE1 XP está instalado en el dispositivo de cómputo del cliente, la sexta actualización 921 es considerada como instalada y la actualización se escribe en la memoria inmediata de la actualización del dispositivo de cómputo del cliente 110. Puesto que la sexta actualización 921 no es una actualización LEAF, el dispositivo de cómputo del cliente 110 envía otra petición de sincronización 1060 que incluye todas las actualizaciones guardadas en el primero y segundo componente de la memoria inmediata del cliente. La petición de sincronización 1060 también incluye una Cookie de autorización del servidor.

En el ejemplo presente, con la utilización del procesamiento de metadatos 123 descrito anteriormente, la petición de sincronización 1060 se procesa en el bloque 1061 donde el servidor selecciona la octava actualización 731. La octava actualización 931 se selecciona porque la petición de sincronización 1060 indica que la quinta y la sexta actualización 915 y 921 están instaladas en el dispositivo de cómputo del cliente 110. Dado que la octava actualización 931 está asociada con el mismo grupo objeto identificado en la Cookie de autorización del servidor, el componente de instrucciones de la octava actualización 931 se comunica con el dispositivo de cómputo del cliente 110 en otra respuesta 1062. La octava

actualización 931 se procesa luego en el bloque 1063 en una forma similar a los procesos de los bloques 1054 y 1059. Puesto que todas las actualizaciones recibidas en la respuesta 1062 son actualizaciones LEAF, la petición de sincronización subsiguiente no se envía devuelta al servidor de metadatos 123.

En el dispositivo de cómputo del cliente 110, luego de que se determina que todas las actualizaciones recibidas son actualizaciones LEAF, o si no se recibieron actualizaciones en la respuesta 1062, la subrutina de sincronización 1050 le comunica un driver de petición de sincronización 1064 del dispositivo de cómputo del cliente 110 al servidor de metadatos 123. Como lo podrán apreciar uno de destreza común en el arte, el driver de petición de sincronización 1064 puede incluir información que describe todo el hardware instalado en el dispositivo de cómputo del cliente 110 e información describiendo el software instalado. Similar al software de petición de sincronización antedicho (1051, 1055 y 1060), el driver de petición de sincronización 1064 puede comunicarle al servidor las actualizaciones instaladas. Adicionalmente, todas las demás actualización que se encuentran actualmente en la memoria inmediata del cliente, si hay alguna, se comunican al servidor.

En respuesta a haber recibido el driver de petición de sincronización 1064, el servidor de metadatos 123 responde enviando todos los drivers de actualización que aplican al dispositivo de cómputo del cliente 110 que no se encuentran en la memoria inmediata del cliente. Un driver de actualización se envía al dispositivo de cómputo del cliente 110 en una respuesta 1065 si los prerequisites y condiciones se cumple. La respuesta 1065 comunica preferiblemente el componente de instrucciones para cada actualización. El driver de actualizaciones luego se escribe en la memoria inmediata de la actualización del dispositivo de cómputo del cliente.

Luego de recibir la respuesta 1065 que contienen el driver de actualización, la subrutina de sincronización 1050 envía una petición 1066 para los datos localizados de cada una de las actualizaciones de software y hardware. Como se resumió anteriormente, el componente de datos localizado de cada actualización

incluye información general que describe la actualización. Por ejemplo, el componente de datos localizado puede incluir información que describe los rasgos y los beneficios de una actualización. El componente de datos localizado puede incluir una descripción de texto de los procedimientos de instalación y de actualización. Adicionalmente el componente de datos localizado puede incluir más datos o información relacionada con la actualización.

Por lo tanto, tan pronto el servidor de metadatos 123 recibe la petición 1066 para los datos localizados de cada una de las actualizaciones de software y hardware recibidas, responde enviando todos los datos localizados para todas las actualizaciones de software y hardware recibidas y guardadas en la memoria inmediata de la actualización del cliente. Una vez se recibe, los datos localizados pueden ser procesados por una aplicación de software para determinar cuál de las actualizaciones necesita instalarse. De otra manera, los datos localizados recibidos se pueden mostrar a un usuario para informarle de todas las actualizaciones que están disponibles para el dispositivo de cómputo del cliente 110. En una implementación, los datos localizados recibidos se pueden mostrar en una página web. En el ejemplo presente, los datos localizados los puede recibir un cliente para la sexta y octava actualización 921 y 931. Si los datos localizados se guardan en la actualización básica 911, 912, 913 y 915, los datos localizados para esas actualizaciones también los recibiría el cliente.

La figura 11 ilustra un ejemplo de una página web 1100 que muestra un ejemplo de datos localizados asociados con las actualizaciones que están disponibles al cliente. Con motivo ilustrativo, la página web 1100 comprende primero una descripción detallada 1105 de una actualización y una segunda descripción detallada 1106 de otra actualización. Como también se muestra, cada actualización está asociada respectivamente con un mecanismo de selección 1103 y 1104 para recibir la selección de actualizaciones del usuario. También se muestra, que la página web 1100 está configurada con un botón de control 1101 para permitirle al usuario controlar la comunicación de la actualización

seleccionada a un servidor, tal como el servidor de meta datos 123 o el servidor de descargas 124.

En un aspecto del presente invento, el cliente ejecuta un número de procesos para mejorar la apariencia en pantalla de la página web 1100. Por ejemplo, el dispositivo de cómputo del cliente 1100 examinar los datos localizados para cada actualización para determinar si una actualización particular es de alta prioridad. Éste rasgo se puede facilitar localizando texto en los datos localizados, o en otros componentes de la actualización particular, que indican que esa actualización particular es de alta prioridad o es una actualización de emergencia. Si el dispositivo de cómputo del cliente 110 detecta una actualización de alta prioridad o de emergencia, el cliente despliega la actualización de alta prioridad en una sección visible de la página web 1100, tal como la sección superior de la página. Adicionalmente, el cliente puede generar un indicador visual, tal como un mensaje de texto especializado 1120, que indica que la actualización es una actualización de alta prioridad.

El dispositivo de cómputo del cliente 110 puede examinar también los datos localizados de cada actualización para determinar si una actualización particular requiere de una instalación exclusiva, por ejemplo, una actualización que tiene un archivo de instalación que no se puede instalar en forma simultánea con un archivo de instalación de otra actualización. Éste rasgo se puede facilitar localizando texto en los datos localizados, o en otro componente de esa actualización particular, que indique que esa actualización particular requiere de una instalación exclusiva. Si el dispositivo de cómputo del cliente 110 detecta tal actualización, el cliente despliega un indicador visual, como un mensaje de texto 1122 que se muestra en la figura 11, con una descripción de las actualizaciones que requieren instalación exclusiva.

Regresando a la figura 7, la rutina del software de actualización 700 continúa en el bloque 708 donde el dispositivo de cómputo del cliente 110 recibe una selección de actualizaciones. Como se anotó arriba, en respuesta a la actuación del

botón de control 1101, la selección de una o más actualizaciones se puede obtener del servidor de metadatos 123 o del servidor de descargas 124. Una vez que la selección de una o más actualizaciones se recibe de la rutina del software de actualización 700 continúa en el bloque 708 donde el software de actualizaciones seleccionado se procesa.

De acuerdo con otro aspecto del presente invento, el servicio de actualización de software 120 puede proveer un método para seleccionar y transmitir información entre el servicio de actualización de software y el dispositivo de cómputo del cliente 110. La figura 12A y 12B ilustran la subrutina de procesamiento de actualizaciones de software 1200 implementadas por el dispositivo de cómputo del cliente 110 para descargar e instalar el software solicitado de acuerdo con el presente invento, tal como se describió más arriba, la subrutina de procesamiento de actualizaciones de software 1200 puede ser implementada una vez que una selección de actualizaciones de software ha sido generada o recibida. Con referencia a la figura 12 A, en el bloque 1200, el componente de administración de actualizaciones 111 inicia al agente de actualización 118. En una implementación ilustrativa del presente invento, el agente de actualizaciones 118 es un componente de software especializado para determinar que actualizaciones de información de software se requieren para completar la actualización de software solicitada, para generar la versión requerida de un componente de instalación del agente de actualización, para generar el archivo de actualización uniendo archivos existentes con parches delta, y/o para iniciar la instalación de archivos de actualización. En el caso de que el agente de actualización 118 ya esté iniciado, el bloque 1202 se puede omitir.

En el bloque 1204, el agente de actualización 118 obtiene información de software de actualización del servicio de actualización 120. En una implementación ilustrativa del presente invento, la información del software de actualización transmitida por el servicio de actualización 120 está en la forma de un paquete, tal como un archivo a auto extraíble, que incluye una variedad de datos que el agente

de actualización puede utilizar. En un aspecto, el paquete puede incluir una lista de todos los archivos que corresponden a una actualización de software particular. Adicionalmente, el paquete puede incluir una copia de por lo menos una porción del parche de almacenaje manifiesto que mapea versiones específicas de archivos que se van a actualizar al correspondiente software de actualización parche delta almacenado en el parche de almacenamiento en el servicio de actualización 120. El paquete también puede incluir información de instalación para cada archivo que se va instalar que puede incluir la identificación de una versión de un componente de instalación requerido para completar la instalación. Aún más, el paquete también puede incluir un componente de instalación para el agente de actualización 118 o un parche delta para actualizar una versión de un componente que ya está instalado o que ya estaba almacenado en el dispositivo de cómputo del cliente 110. También, el paquete puede incluir información de verificación que permite que el agente de actualización determine si un software de actualización fue exitoso. Por ejemplo, la información de verificación puede incluir, para comparar, valores de referencia para archivos actualizados. El agente de actualización 118 puede verificar también el contenido del paquete.

En el bloque de decisión 1206, se conduce una prueba para determinar si el agente de actualización 118 necesita una versión actualizada del componente de instalación para implementar la actualización. Aquel, diestro en el arte relevante, puede apreciar que en la transmisión de una copia completa de un componente de instalación en un archivo auto extraíble se puede incrementar la cantidad de datos transmitidos por el servicio de actualización 120 para cada actualización de software. De acuerdo, en una implementación ilustrativa del presente invento, una versión básica del componente de instalación se puede almacenar en el dispositivo de cómputo del cliente y actualizar en forma específica para los requerimientos de la actualización de software actual a través de un componente de instalación de un parche Delta. De acuerdo, la información de instalación en el archivo auto extraíble le da instrucciones al agente de actualización 118 acerca de si un componente de instalación de actualización necesita unirse con una versión básica del componente

de instalación en el dispositivo de cómputo del cliente 110. Si una actualización es requerida, en el bloque 1208, el agente de actualización 118 actualiza el componente de instalación básico, como se explica en más detalle mas adelante en referencia a la figura 13.

Una vez que el agente de actualización actualiza el componente de instalación o si el componente de instalación no requiere una actualización, en el bloque 1210, el agente de actualización 118 hace un inventario de los archivos instalados en el dispositivo de cómputo del cliente 110 junto con la versión específica del archivo. En una implementación ilustrativa del presente invento, el agente de actualización 118 puede consultarle al sistema de archivos del dispositivo de cómputo del cliente 110 por todos los archivos identificados en el paquete como correspondientes a la actualización seleccionada. En forma alternativa, si el agente de actualización 118 ha hecho recientemente un inventario, una versión almacenada en la memoria instantánea del inventario puede ser utilizada. En el bloque 1212, el agente de actualización 118 identifica que información de actualización de software se requiere para completar la actualización solicitada. En una implementación ilustrativa del presente invento, el parche manifiesto de almacenaje incluye un mapeo de las versiones de archivos instaladas con el parche Delta requerido. De acuerdo, si la utilización del parche Delta está disponible, el agente de actualización 118 puede utilizar el mapeo para identificar parches Delta particulares y su localidad dentro del archivo de almacenaje del parche. En forma alternativa, si el parche Delta no está disponible o no se puede incrementar, el agente de actualización 118 puede identificar un archivo completo para descarga. Un

En referencia ahora a la figura 12B, en el bloque 1214, el agente de actualización transmite una petición para la información de actualización del software identificado. En una implementación ilustrativa del presente invento, el agente de actualización 118 puede transmitir una petición para parches Delta específicos indicando un rango específico de parches requeridos del archivo de

almacenamiento de parches para ser descargados en el servidor 124 del servicio de actualización 120. Como se describió anteriormente, el archivo de parche incluye un gran número de parches Delta aplicables, en donde cada parche Delta se identifica por su localidad en el archivo de almacenaje de parches. Porque el archivo de almacenaje de parches puede ser bastante grande en algunas implementaciones, el agente y actualización 118 puede utilizar una petición que solicita datos únicamente de una localidad específica en el archivo de almacenamiento de parches como se indica en el almacenaje de parches manifestado. En una implementación alterna del presente invento, el agente de actualización 118 puede solicitar una copia entera del archivo de actualización y/o una copia completa del archivo de almacenamiento de parches.

En una implementación alterna del presente invento, otro servidor de descargas que puede no estar asociado en forma exclusiva con el servicio de actualización 120 puede procesar la petición de actualización del agente 118. En esta implementación, la solicitud del archivo de almacenamiento de parches puede ser transmitida, en su totalidad o en buena parte, a cualquier número adicional de servidores de descargas en red. Los servidores de descarga adicionales pueden ser parte de una red privada que se utiliza para actualizar a los clientes en una red privada. Aún más, el servidor de descarga adicional puede ser parte de una red pública. En el ambiente de una red privada, el servidor de descargas puede obtener una copia completa del archivo de almacenamiento de parches para procesar una solicitud de un cliente. En forma alterna, el servidor de descarga puede almacenar en la memoria inmediata porciones del archivo de almacenamiento de parches cuando procesa solicitudes anteriores de otros clientes y utiliza los datos almacenados para satisfacer la descarga. De acuerdo con el servidor de descarga adicional se puede reducir la carga en las comunicaciones entre el servidor de descargas 124 y el servicio de actualizaciones 120.

En el bloque 1216, el agente de actualización 118 recibe la información de actualización solicitada. En una implementación ilustrativa del presente invento, la

información de actualización solicitada se puede transmitir de dos maneras. De una manera, referida como actualización manual, la solicitud de actualización se envía al servicio de actualización 120 con una solicitud de transferencia de datos directa HTTP. En esta forma, el servicio de actualización 120 puede utilizar todo el ancho de banda disponible para transmitir los datos disponibles al agente de actualización 118. En una segunda manera, referida como actualización automática, la solicitud de actualización es enviada del servicio de actualización 120 con una solicitud para una respuesta indirecta de entrega de datos HTTP. En esta respuesta, el servicio de actualización 120 transmite los datos solicitados en un proceso en segundo plano. Este proceso se puede implementar de tal forma que utilice la cantidad mínima de ancho de banda disponible. Aún más, este proceso se puede interrumpir durante la descarga e iniciar de nuevo cuando haya tiempo disponible. En la comúnmente asignada y co-pendiente aplicación con patente de EU. número 09/505,735 y llamada Sistema y Método para Transferir Datos a través de una Red, presentada el 16 de febrero de 2000 y que se incorpora acá por referencia se describe un sistema y método para transmitir los datos solicitados en un proceso que opera en segundo plano. Una persona diestra en el arte apreciará que una entrega de datos en primer plano o en segundo plano no necesariamente refleja la prioridad del software de actualización seleccionado, pero más bien el ancho de banda asignado para obtener la información de actualización.

Una vez que se recibe la información solicitada del servicio de actualización, en el bloque 1218, el agente de actualización 118 une el parche delta con los archivos correspondientes instalados. En una implementación ilustrativa del presente invento, el agente de actualización 118 puede llevar a la memoria instantánea la versión original del archivo instalado para asegurarse que el archivo seleccionado no cambia durante el proceso de descarga y unión. Aún más, la versión original almacenada en la memoria instantánea se puede utilizar para desinstalar la actualización seleccionada.

35

En el bloque de decisión 1220, se hace una prueba para determinar si el archivo de actualización es válido. En una implementación ilustrativa del presente invento, el agente de actualización 118 puede utilizar un algoritmo marcado para comparar un valor obtenido del paquete de información de actualización y el correspondiente a un archivo de actualización válido con una marca del archivo actual modificado. Si las marcas no concuerdan, el archivo actual modificado no es válido. Uno diestro en el arte podrá apreciar que uno de cualquier número de algoritmos de validación alternos también pueden ser utilizados. Si el archivo actualizado no es válido, la subrutina 1200 regresa al bloque 1214, donde el agente de actualización puede solicitar de nuevo información de actualización. De otra manera, si el agente de actualización 118 no ha podido generar en varios intentos el archivo de actualización, el agente de actualización puede implementar uno de varios procedimientos de recaída. En una implementación del presente invento, el agente de actualización 118 puede solicitar una copia completa del archivo de actualización almacenado en el archivo de almacenamiento de parches e identificado en el parche de almacenaje manifiesto del servicio de actualización 120, en otra implementación del presente invento, el agente de actualización 118 puede solicitar una copia del archivo de actualización en un archivo auto contenido de un servicio de actualización 120. Aún, en otra implementación del presente invento, la subrutina 1200 puede fracasar.

Una vez que se ha seleccionado el archivo válido, en el bloque de decisión 1222, se hace una prueba para determinar si se requieren descargas adicionales. En una implementación ilustrativa del presente invento, la subrutina 1200 entra en un círculo repetitivo que chequea continuamente para descargas adicionales después de haber terminado con la descarga seleccionada. Si el estado de un archivo cambia durante lo descarga, el agente de actualización 118 continuaría solicitando descargas adicionales para el nuevo estado. Si descargas adicionales se necesitan, en el bloque 1224, el agente de actualización 118 hace otro inventario e identifica todos los parches del aplicables la subrutina 1200 luego regresa el bloque 1214.

35

Tan pronto la descarga de actualización se completa, en el bloque de decisión 1226, se hace una prueba para determinar si el estado de la máquina del escribiente ha cambiado. En una implementación ilustrativa del presente invento, puede haber un intervalo de tiempo entre la descarga y la fusión de la información de actualización y la instalación propiamente dicha del archivo de actualización. De acuerdo, antes de instalar el archivo actualizado, el agente de actualización determinar si el dispositivo de cómputo del cliente ha cambiado de estado. Si el estado ha cambiado, el archivo actualizado puede no ser válido y la actualización fracasa en el bloque 1228. De otra manera, si el estado no ha cambiado, el agente de actualización 118 instalar el archivo actualizado en el bloque 1230 y la subrutina 1200 regresa al bloque 1232.

En referencia ahora a la figura 13, una subrutina 1300 implementada por el dispositivo de cómputo del cliente 110 para actualizar a un componente básico de instalación correspondiente al bloque 1208 (figura 12 A) será descrita en el bloque de decisión 1302, se hace una prueba para determinar si un componente de instalación básico se incluye en el archivo auto extraíble que se le transmite al agente de actualización 118 del servicio de actualización 120. En una implementación ilustrativa del presente invento, si los parches delta requeridos para actualizar el instalador básico son comparables en tamaño a la transmisión de un componente de instalación actualizado, un componente nuevo básico de instalación será transmitido. Si un componente de instalación actualizado se incluye, en el bloque 1304, el agente de actualización instala el componente actualizado como el nuevo componente de instalación. Adicionalmente, el componente de instalación actualizado nuevo se puede guardar en la memoria del dispositivo de cómputo del cliente 110 para que sirva de instalador básico en actualización es adicionales. En el bloque 1306, la subrutina regresa.

Si un componente básico de instalación no se incluye en el archivo auto extraíble, en el bloque 1308, el agente de actualización 118 obtiene un componente de instalación básico parche delta del archivo auto extraíble. En una

57

implementación ilustrativa del presente invento, el componente de instalación parche delta corresponde a un código de software que se puede fusionar con el componente básico de instalación para generar un componente básico de instalación actualizado. De acuerdo, en el bloque 1310, el agente de actualización fusiona el componente básico de instalación parche delta con el componente básico de instalación. En el bloque 1312, el agente de actualización 118 designa luego el componente básico de instalación como el componente de instalación actual. En una implementación ilustrativa del presente invento, el componente de instalación actualizado no se guardara luego de que la instalación termine. De acuerdo con esta implementación, el agente de actualización 118 únicamente mantiene un número limitado de componentes básicos de instalación en la memoria del dispositivo de cómputo del cliente 110. De acuerdo, el agente de actualización genera un componente de instalación actualizado temporal en cada instalación. Porque el dispositivo de cómputo de cada cliente 110 sólo puede corresponder a un número limitado de componentes básicos de instalación, el servicio de actualización 120 sólo debe transmitir un componente básico de instalación parche delta para cada dispositivo de cómputo de un cliente. En el bloque 1314, la subrutina 1300 regresa.

Mientras que la implementación preferida del invento se ha ilustrado y descrito, se apreciará que varios cambios se pueden hacer sin apartarse del espíritu y el alcance del invento. Por ejemplo aunque los ejemplos ilustrativos descritos acá aplican a actualización de software, el alcance del presente invento incluye otros usos más allá de la distribución y comunicación de información relacionada con actualizaciones de software. De esta manera, al menos de que tópicos específicos se hayan excluido en esta divulgación, se debe apreciar que el alcance del presente invento aplica a la distribución y comunicación de cualquier tipo de datos, fuera de, o adicional a actualizaciones de software.

57

La implementación del invento en las cuales se reivindica propiedad exclusiva o privilegios se definen de la siguiente forma:

1. En un sistema de computación que incluye el dispositivo de cómputo de un cliente que está en comunicación con un servicio de actualización de software, un método para comunicar actualizaciones de software de un servicio de actualización de software al dispositivo de cómputo del cliente, método que consta de:

(a) obtener una solicitud de sincronización del dispositivo de cómputo del cliente al servicio de actualización de software, donde la solicitud de sincronización incluye un identificador de actualizaciones de software instaladas si el dispositivo de cómputo del cliente almacena el software de actualización instalado;

(b) determinar si la solicitud de sincronización tiene un identificador de por lo menos una actualización de software instalada;

(c) si determina que la solicitud de sincronización tiene un identificador de por lo menos una actualización de software instalada, seleccionar actualizaciones adicionales de software para comunicar al dispositivo de cómputo del cliente, donde la selección de actualizaciones de software adicionales depende del cumplimiento de prerequisites definidos en la actualización de software adicional, donde los prerequisites requieren que la solicitud de sincronización incluya un identificador de por lo menos una actualización de software instalada;

(d) si determina que la petición de sincronización no tiene un identificador para por lo menos una actualización de software instalada, seleccionar el primer nivel de actualizaciones de software para comunicar al dispositivo de cómputo del cliente, donde el primer nivel de actualizaciones de software no incluye un prerequisite;

(e) comunicar un componente de instrucciones de actualización del software seleccionado al dispositivo de cómputo del cliente de un servicio de actualización de software; y

(f) guardar el componente de instrucciones de la actualización de software seleccionada en el dispositivo de cómputo del cliente como una actualización de software instalada si el dispositivo de cómputo del cliente contiene por lo menos un componente que cumple una de las condiciones por de una regla de aplicabilidad guardada en la actualización de software seleccionada.

2. El método como se describió en la reivindicación 1, donde el método incluye además:

Determinar si por lo menos un componente de instrucción de la actualización de software seleccionada provee una indicación de que el servicio de actualizaciones de software tiene actualizaciones de software adicionales disponibles para el dispositivo de cómputo del cliente; y

Repitiendo los elementos entre (a)-(f) si se determina que por lo menos un componente de instrucciones del software de actualización seleccionado provee una indicación de que el servicio de actualización de software tiene actualizaciones de software adicionales disponibles para el dispositivo de cómputo del cliente.

3. El método como se describe en la reivindicación 1, donde el método consta además:

si el componente de instrucciones no da una indicación de que el servicio de actualización de software tiene actualizaciones de software adicionales disponibles para el dispositivo de cómputo del cliente:

comunicar un componente de datos localizado en la actualización de software seleccionada del servicio de actualización de software al dispositivo de cómputo del cliente; y;

generar una vista que contiene texto derivado del componente de datos localizado y comunicado, donde el texto da una descripción de la actualización de software seleccionada; y

en respuesta a haber recibido la orden, comunicar el archivo de datos asociado con la actualización de software seleccionada del servicio de actualización de software al dispositivo de cómputo del cliente.

5. El método como se describe en la reivindicación 1, donde el método consta además de: determinar si el dispositivo de cómputo del cliente almacena actualizaciones de software instaladas; proveer un identificador de actualizaciones de software instaladas en la solicitud de sincronización si se determina que el dispositivo de cómputo del cliente almacena las actualizaciones de software instaladas.

6. El método como se describe en la reivindicación 1, donde el prerequisite define una condición que requiere una combinación de actualizaciones de software referenciadas en la solicitud de sincronización, donde la condición se define por lo menos con un operador lógico que vincula la combinación de actualizaciones de software.

7. El método como se describe en la reivindicación 1, donde el método consta además de:

guardar el componente de instrucciones de la actualización de software seleccionada en el dispositivo de cómputo del cliente como una actualización fracasada si el dispositivo de cómputo del cliente no contiene por lo menos uno de los componentes que cumple por lo menos una condición de aplicabilidad almacenada en la actualización de software seleccionada.

8. El método como se describe en la reivindicación 1, donde el método consta además de:

obtener un identificador de la actualización de software fracasada en la petición de sincronización si el dispositivo de cómputo del cliente almacena el software de actualización fracasado;

si se determina que la solicitud de sincronización consta de un identificador de la actualización de software fracasada, restringe la comunicación de las actualizaciones de software seleccionada y relacionadas con actualización de software fracasada.

9. El método como se describe en la reivindicación 1 donde la solicitud de sincronización generada de la ejecución del módulo de actualización automático en el dispositivo de cómputo del cliente.

10. El método como se describe en la reivindicación 1, donde la solicitud de sincronización la invoca un usuario solicitando una actualización manual

11. El método como se describe en la reivindicación 1, donde la actualización de software seleccionada se configura para actualizar uno o más archivos instalados en el dispositivo de cómputo del cliente, donde comunicar la actualización de software seleccionada del servicio de actualización de software al dispositivo de cómputo del cliente consta de:

obtener un archivo auto extraíble que identifica una pluralidad de parches del todo disponibles para actualizar por lo menos una versión del archivo instalado;

obtener un inventario de uno o más archivos instalados en el dispositivo de cómputo del cliente;

seleccionar uno o más parches Delta apropiados para implementar la actualización de software seleccionada, donde seleccionar uno o más parches apropiados corresponde a un mapeo de los archivos auto extraíbles identificando una pluralidad de parches Delta disponibles al inventario de uno o más archivos instalados en el dispositivo de cómputo del cliente; y

transmitir una solicitud para uno o más parches Delta seleccionados

12. Un medio legible por computador que tiene instrucciones ejecutables por computador para realizar el método descrito en la reivindicación 1.

13. Un sistema de computación con un procesador, una memoria y un ambiente operativo, el sistema de computación operable para ejecutar el método descrito en la reivindicación 1.

14. En un sistema de computación que incluye el dispositivo de cómputo del cliente en comunicación con un servicio de actualización de software, un método para comunicar actualizaciones de software de un servicio de actualización de software al dispositivo de cómputo de un cliente, método que consta de:

obtener una solicitud de autorización del dispositivo de cómputo de un cliente, donde la solicitud de autorización contiene un identificador del módulo de autorización del cliente guardado en el dispositivo de cómputo del cliente;

determinar si el dispositivo de cómputo del cliente está asociado con un grupo objeto, donde se determina si el dispositivo de cómputo del cliente está asociado con un grupo objeto si el módulo de autorización del cliente indica que el dispositivo de cómputo del cliente está asociado con el grupo objeto;

comunicar una Cookie de servidor del servicio de actualización de software al dispositivo de cómputo del cliente si se determina que el dispositivo de cómputo del cliente está asociado con el grupo objeto, donde la Cookie del servidor identifica al grupo objeto;

obtener una solicitud para por lo menos una actualización de software guardada en el servicio de actualización de software, donde la solicitud contiene la Cookie del servidor;

en respuesta a la solicitud que se obtiene, determinar si la actualización de software está asociada con el grupo objeto identificado en la Cookie del servidor; y

si se determina que la actualización de software está asociada con el grupo objeto identificado en la Cookie del servidor, comunicar el software de actualización de servicios de actualización de software al dispositivo de cómputo del cliente.

15.El método como se describe en la reivindicación 14. Donde la solicitud de autorización incluye datos de una Cookie del servidor guardada previamente, y donde la Cookie del servidor comunicada del servicio de actualización de software al dispositivo de cómputo del cliente incluye datos de la Cookie del servidor previamente guardada.

16. El método como se describe en la reivindicación 14, donde se determina que el dispositivo de cómputo del cliente está asociado con un grupo objeto si el módulo de autorización del servidor identifica el mismo grupo objeto señalado en el módulo de autorización del cliente.

17. El método como se describe en la reivindicación 14, donde la Cookie del servidor tiene una fecha de vencimiento, donde el método consta además:

determinar si la fecha de vencimiento de la Cookie del servidor ha pasado;

si se determina que la fecha de vencimiento en la Cookie del servidor ha pasado, restringir la comunicación del actualización de software del servicio de actualización de software al dispositivo de cómputo del cliente.

18. El método como se describe en la reivindicación 14, donde la Cookie del servidor almacena la fecha de vencimiento en un formato de texto claro y en un formato codificado, y en donde la Cookie del servidor contiene datos codificados que identifican por lo menos un grupo objeto, donde el método además consta:

obtener la fecha de vencimiento de la Cookie del servidor, donde el dispositivo de cómputo deficiente obtiene la fecha de vencimiento de la fecha de vencimiento almacenada en formato claro de texto;

determinar si la fecha de vencimiento ha pasado; y

si se determina que la fecha de vencimiento no ha pasado, comunicar la Cookie del servidor del dispositivo de cómputo del cliente al servicio de actualización de software en la petición de por lo menos una actualización de software.

19. El método como se describe en la reivindicación 14, donde el módulo de autorización se guarda en el dispositivo de cómputo del cliente durante una instalación de una aplicación de software.

20. El método como se describe en la reivindicación 14, donde el módulo de autorización se guarda en el dispositivo de cómputo del cliente durante un proceso de actualización de aplicaciones de software.

21. El método como se describe en la reivindicación 14, donde comunicar la actualización de software del servicio de actualización de software al dispositivo de cómputo del cliente incluye la comunicación de un componente de instrucciones de la actualización del software, en donde el método incluye además:

procesar el componente de instrucciones del software de actualización para determinar si el componente de instrucciones da una indicación de que el servicio de actualización de software tiene actualizaciones de software adicionales disponibles para el dispositivo de cómputo del cliente;

comunicar un componente de datos localizado de la actualización de software de un servicio de actualización de software al dispositivo de cómputo del cliente si el componente de instrucciones no da una indicación de que el servicio de actualización de software contiene actualizaciones de software adicionales y disponibles para el dispositivo de cómputo del cliente; y

procesar el componente de datos localizado para seleccionar por lo menos un archivo de actualización para actualizar los archivos instalados en el dispositivo de cómputo del cliente.

22. El método como se describe en la reivindicación 21, donde el procesamiento del componente de datos localizado incluye:

Mostrar en pantalla datos del componente de datos localizado en una pantalla del dispositivo de cómputo del cliente, donde la visualización incluye un componente para generar una solicitud para una actualización seleccionada; y

obtener por lo menos una solicitud indicando la selección de por lo menos un archivo de actualización para actualizar archivos instalados en el dispositivo de cómputo del cliente.

23. El método como se describe en la reivindicación 14, donde el módulo de autorización del servidor es un plug-in de software configurado para añadirse en forma dinámica o removerse del servicio de actualización de software.

24. Es método como se describe en la reivindicación 14, donde la solicitud para por lo menos una actualización de software se genera de un módulo de programa automático de actualización que se ejecuta en el dispositivo de cómputo del cliente.

25. El método como se describe en la reivindicación 14, donde la solicitud para por lo menos una actualización de software la invoca un usuario solicitando una actualización manual.

26. El método como se describe en la reivindicación 14, donde la actualización de software está configurada para actualizar uno o más archivos instalados en el dispositivo de cómputo del cliente, donde la comunicación de actualizaciones de software del servicio de actualización de software al dispositivo de cómputo del cliente consta de:

obtener una selección de uno o más actualizaciones de software disponibles para actualizar uno o más archivos instalados en el dispositivo de cómputo del cliente;

obtener un archivo auto extraíble que identifica una pluralidad parches delta para actualizar por lo menos una versión de un archivo instalado;

obtener un inventario de uno o más archivos instalados en el dispositivo de cómputo del cliente;

seleccionar uno o más parches delta apropiados para implementar la actualización de software seleccionada, donde seleccionar uno o más parches apropiados corresponde a un mapeo del archivo auto extraíble identificando una pluralidad de parches delta disponibles al inventario de uno o más archivos instalados en el dispositivo de cómputo del cliente; y

transmitir una solicitud para uno o más parches delta seleccionados.

27. Un medio legible por computadora que tiene instrucciones ejecutables para realizar el método descrito en la reivindicación 14.

28. Un sistema de computación con un procesador, una memoria y un ambiente operativo, el sistema de computación operable para llevar a cabo el método descrito en la reivindicación 14.

29. Un sistema de computación que incluye el dispositivo de cómputo y un cliente en comunicación con un servidor, un método para comunicar estructuras de datos de un servidor al dispositivo de cómputo y un cliente, método que consta de:

determinar si el dispositivo de cómputo del cliente está asociado con un grupo objeto, donde se determina si el dispositivo de cómputo del cliente está asociado con el grupo objeto si el dispositivo de cómputo del cliente contiene un módulo de autorización indicando que el dispositivo de cómputo del cliente está asociado con un grupo objeto;

comunicar una Cookie de servidor del servidor al dispositivo de cómputo del cliente si se determina que el dispositivo de cómputo del cliente está asociado con un grupo objeto, donde la Cookie del servidor identifica al grupo objeto;

obtener una solicitud para por lo menos una estructura de datos almacenada en el servidor, donde la solicitud contiene la Cookie del servidor;

en respuesta a la solicitud, determinar si la estructura de datos está asociada con el grupo objeto identificado en la Cookie del servidor;

si se determina que la estructura de datos está asociada con el grupo objeto identificado en la Cookie del servidor, comunicar la estructura de datos del servidor al dispositivo de cómputo del cliente.

30. El método como se describe en la reivindicación 29, donde el método consta además de:

Obtener una solicitud de autorización del dispositivo de cómputo del cliente, donde la solicitud de autorización contiene un identificador del módulo de autorización guardado en el dispositivo de cómputo del cliente;

en respuesta a la solicitud de autorización, determinar si el dispositivo de cómputo del cliente está asociado con un grupo objeto, donde se determina que el dispositivo de cómputo del cliente está asociado con el grupo objeto si la solicitud de autorización identifica por lo menos un módulo de autorización almacenado en el dispositivo de cómputo del cliente que está asociado con el grupo objeto.

31. El método como se describe en la reivindicación 29, donde la Cookie del servidor tiene una fecha de vencimiento, donde el método comprende además: determinar si la fecha de vencimiento de la Cookie del servidor ha pasado;

si se determina que la fecha de vencimiento en la Cookie del servidor ha pasado, restringir la comunicación de la estructura de datos del servidor al dispositivo de cómputo del cliente.

32. El método como se describe en la reivindicación 29, donde la Cookie del servidor almacena la fecha de vencimiento en un formato de texto claro y en un formato codificado, y donde la Cookie del servidor contiene datos codificados que identifica por lo menos un grupo objeto, donde el método incluye además:

Obtener la fecha de vencimiento de la Cookie del servidor, donde el dispositivo de cómputo del cliente obtiene la fecha de vencimiento de la fecha de vencimiento almacenada en formato claro de texto; determinar si la fecha de vencimiento no ha pasado;

si se determina que la fecha de vencimiento no ha pasado, comunicar la Cookie del servidor del dispositivo de cómputo del cliente al servicio de actualización de software en una solicitud de por lo menos una estructura de datos.

33. Un medio legible por computadora que tiene instrucciones ejecutables para realizar el método descrito en la reivindicación 29.

34. Un sistema de computación que tiene un procesador, una memoria y un ambiente operativo, el sistema de computación operable para realizar el método descrito en la reivindicación 29.

COMPENDIO DE LA DIVULGACIÓN

Un sistema y método para facilitar la selección e implementación de actualizaciones de software mientras se minimiza el ancho de banda y los recursos de procesamiento requeridos para seleccionada implementar una actualización de software. En una implementación, un servicio de actualización controla el acceso a las actualizaciones de software, u otros tipos de software almacenados en el servidor.

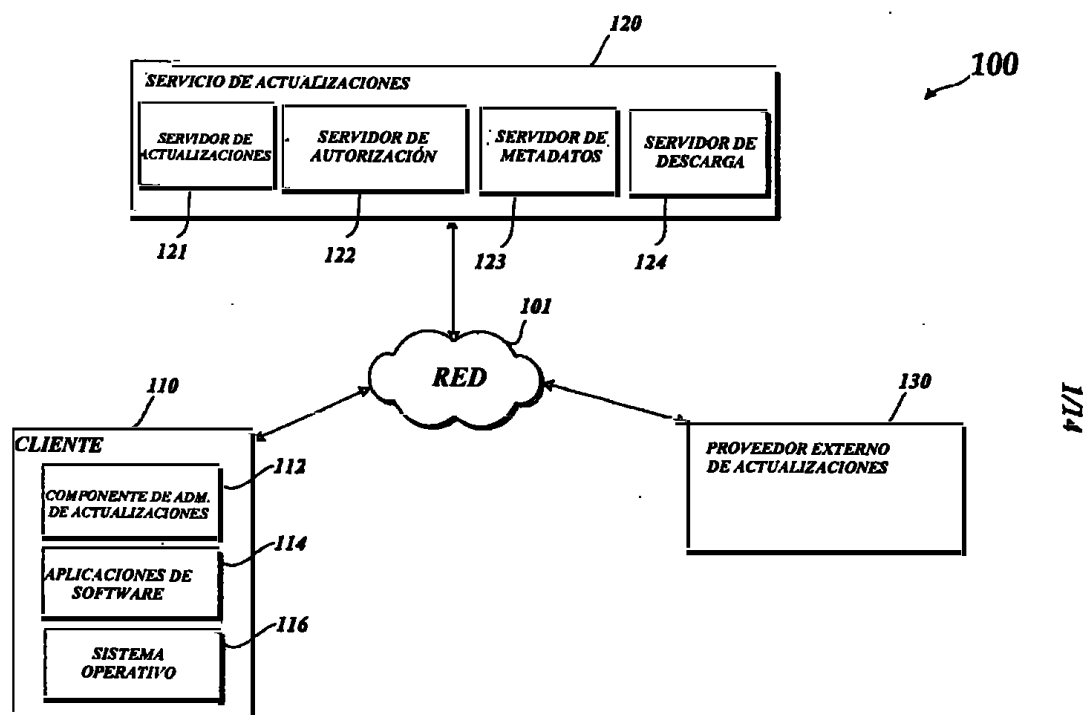


Fig. 1

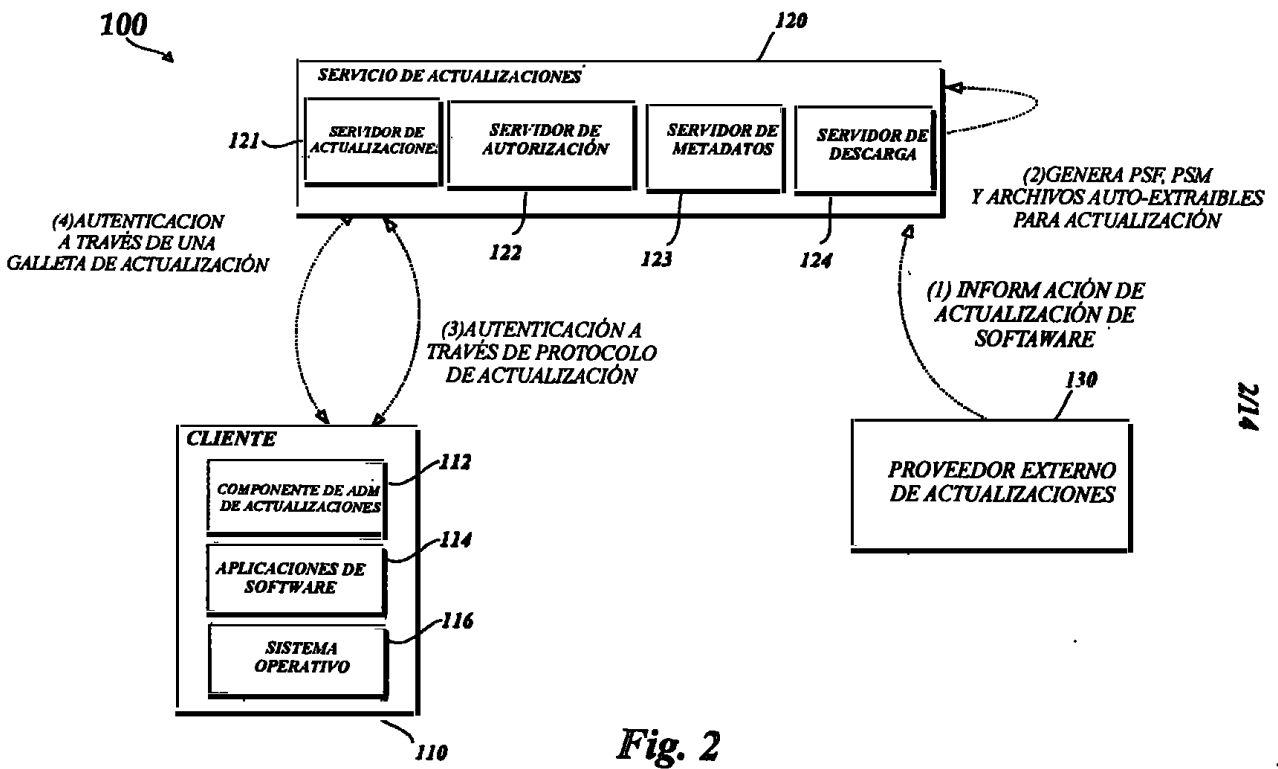


Fig. 2

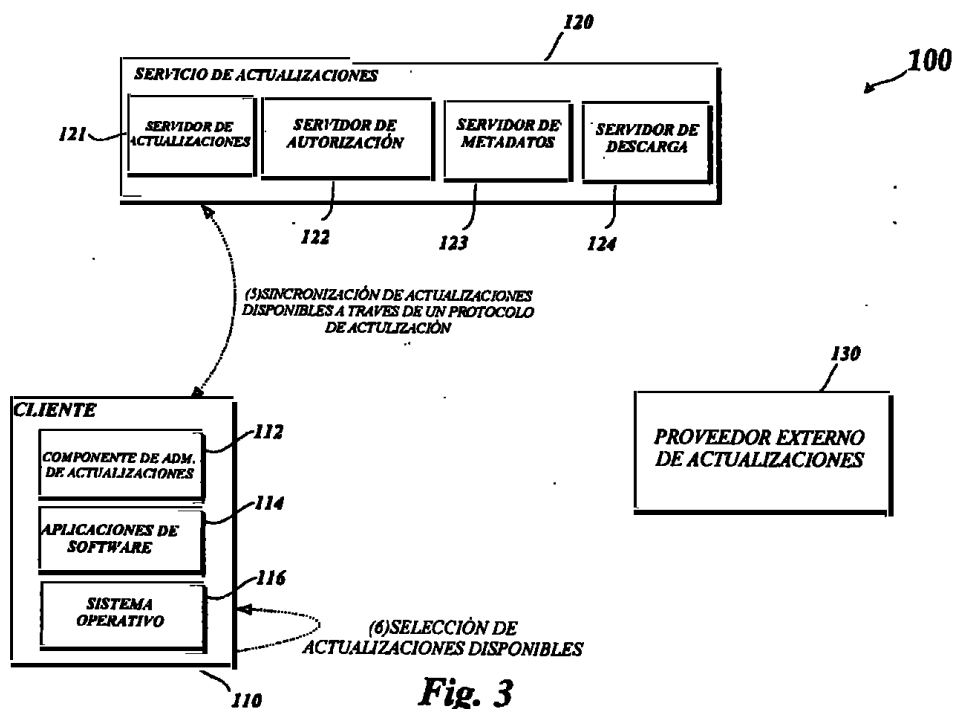
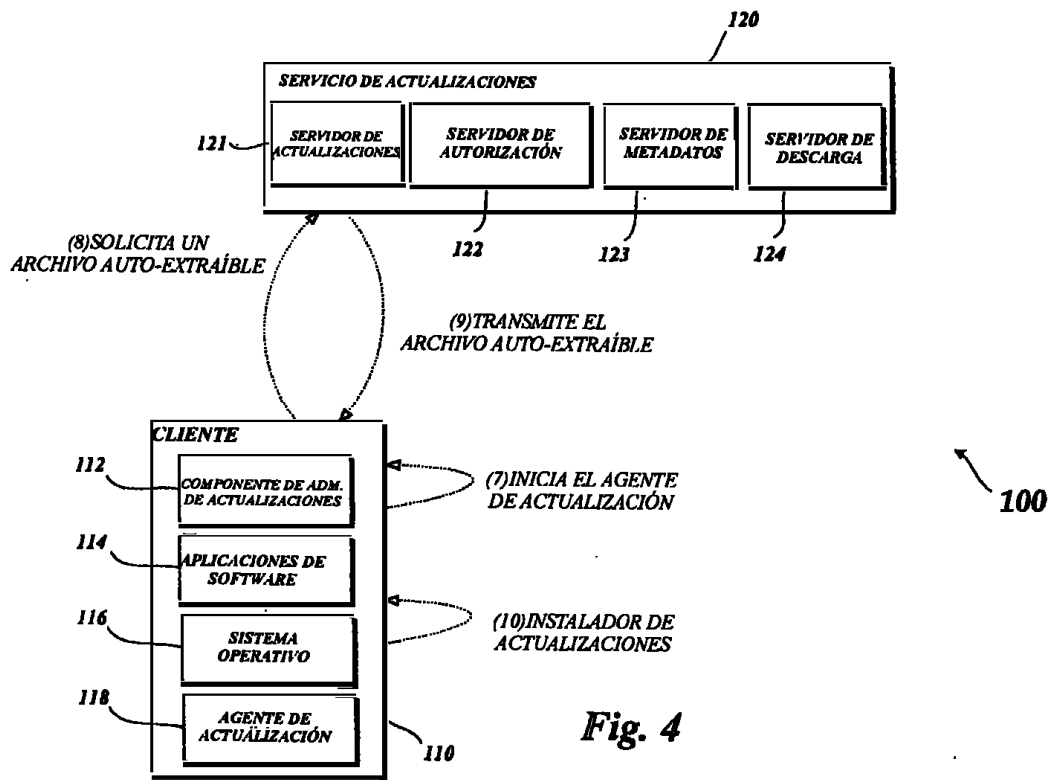


Fig. 3



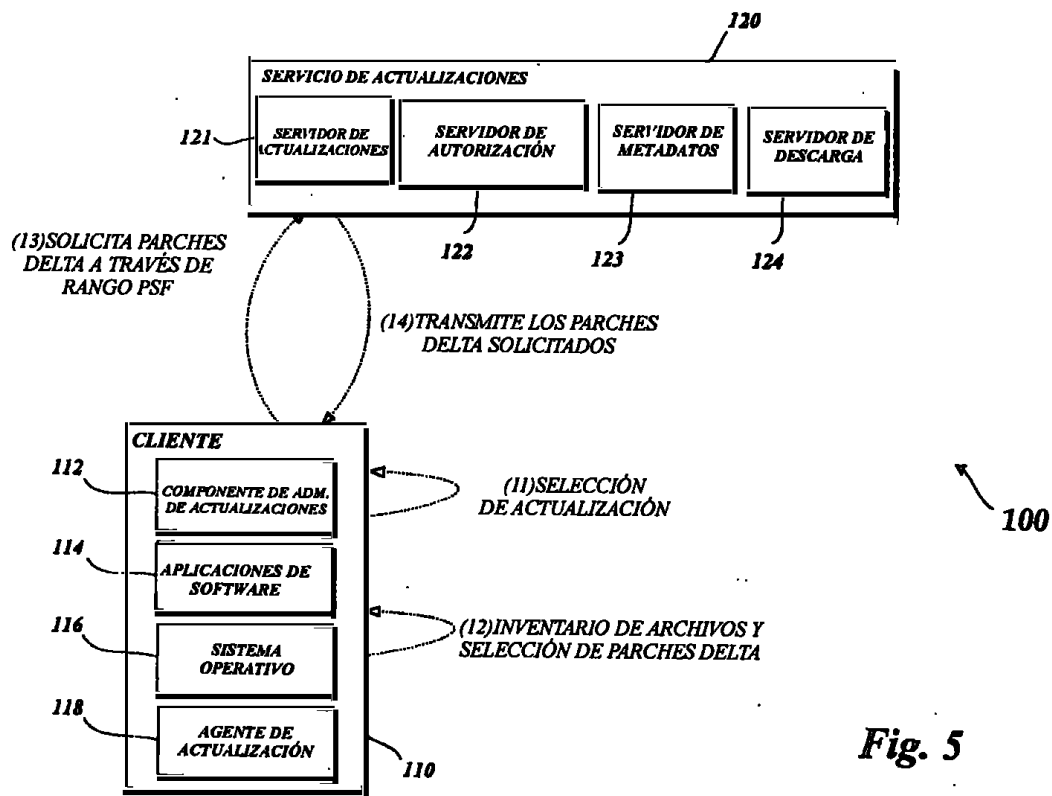
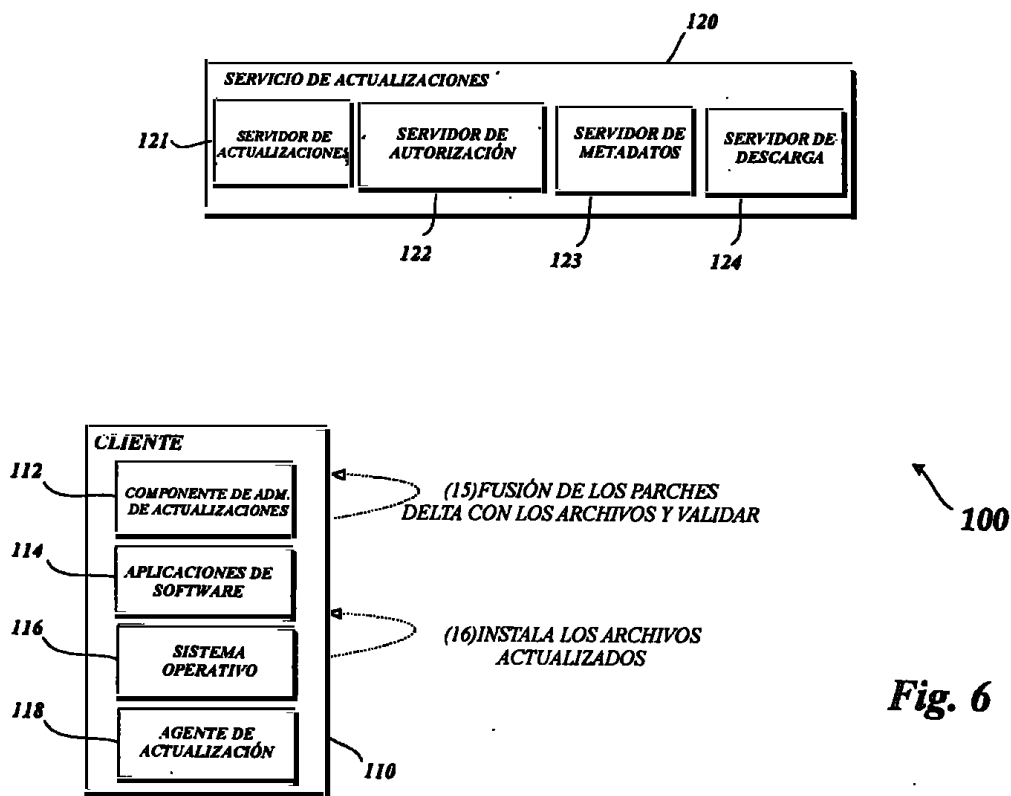


Fig. 5

5/14

74

74



674

0

0

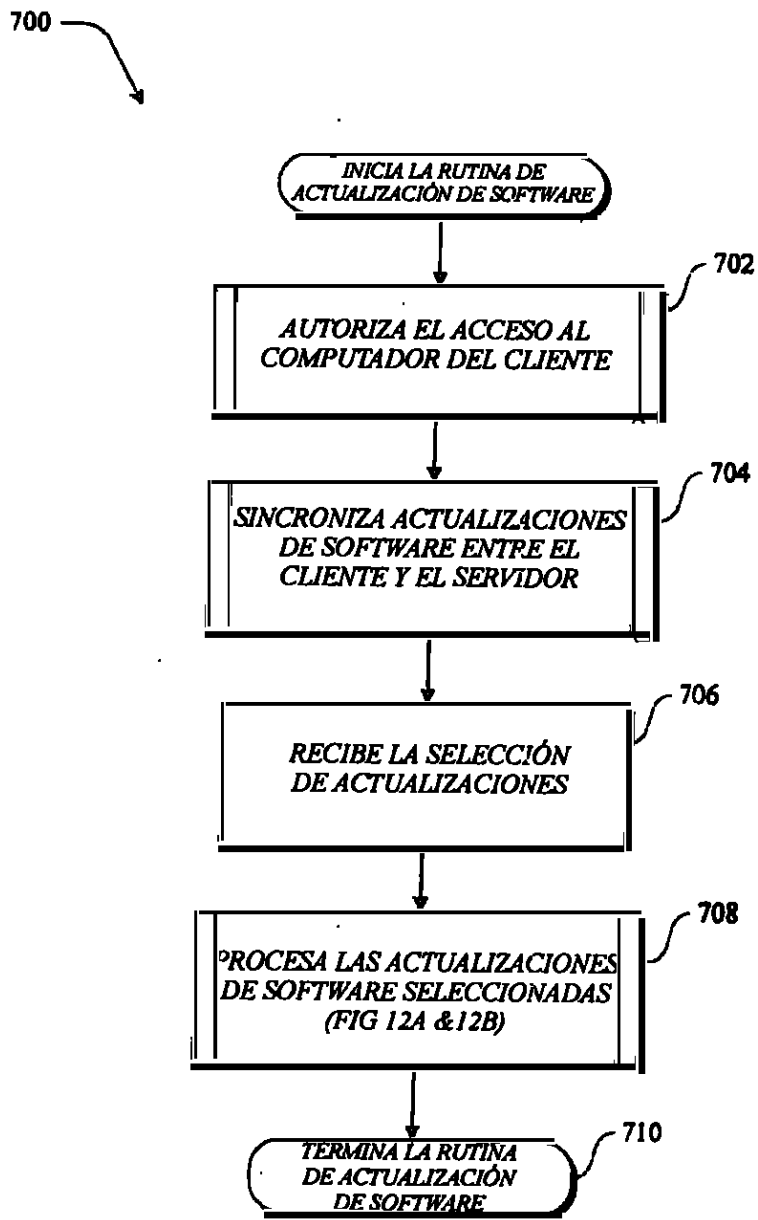


Fig. 7

96

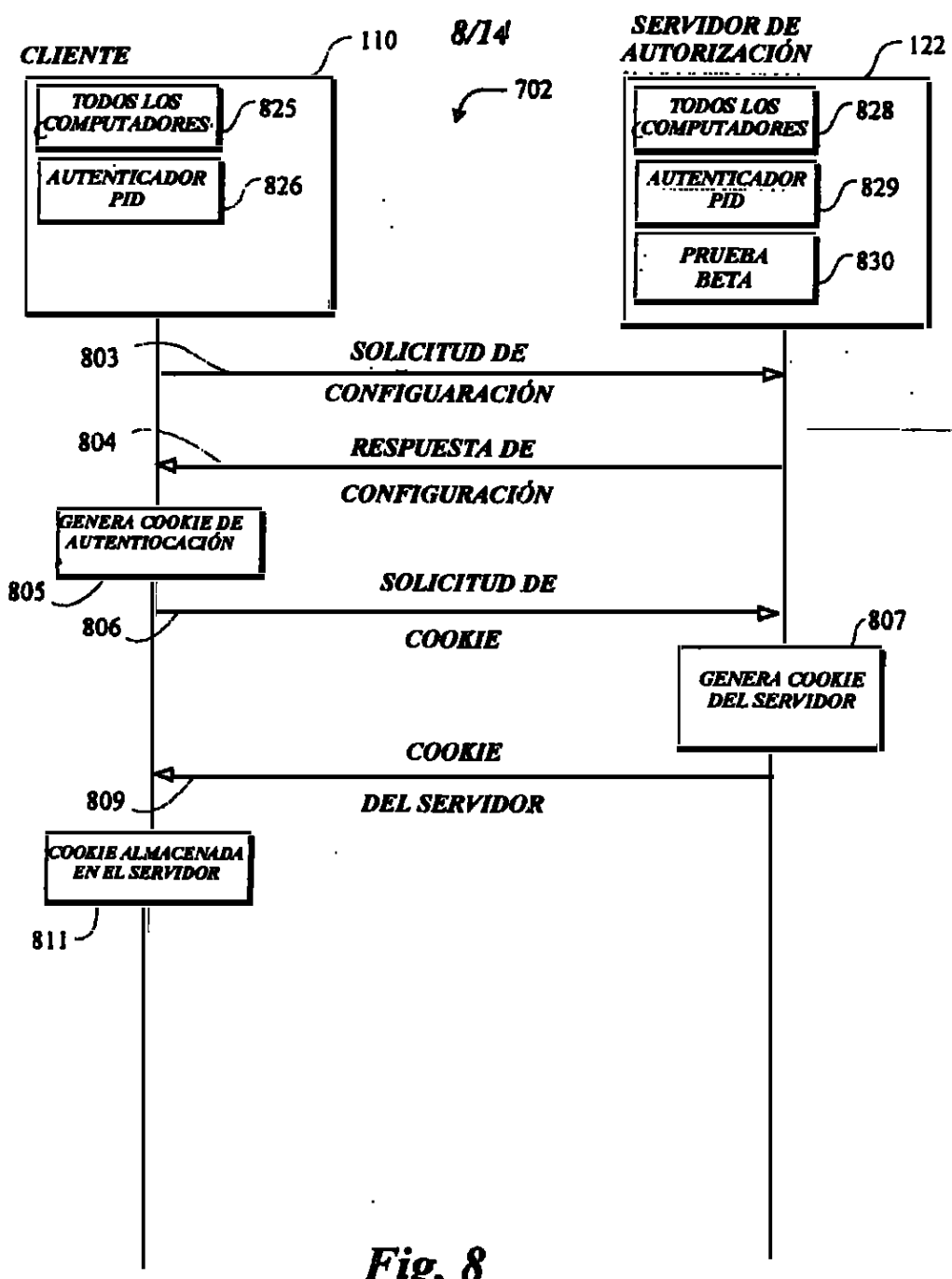
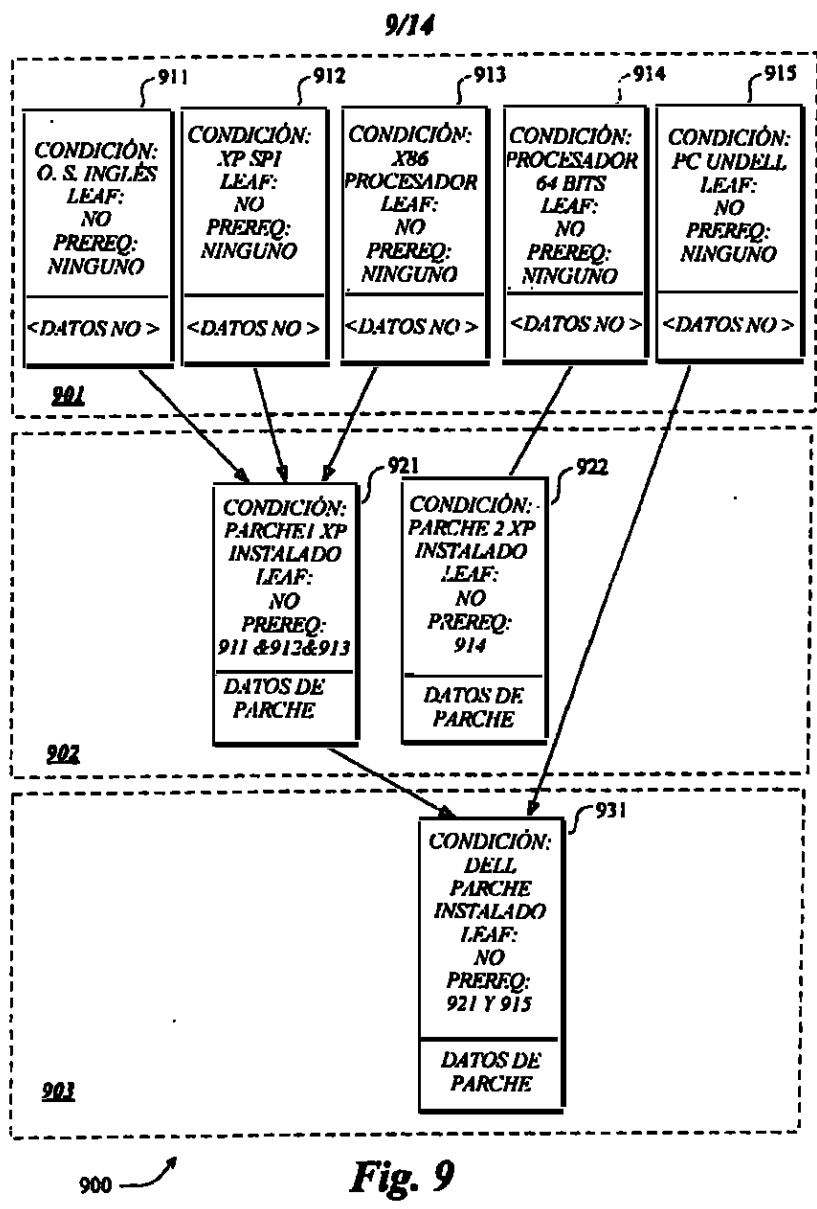


Fig. 8



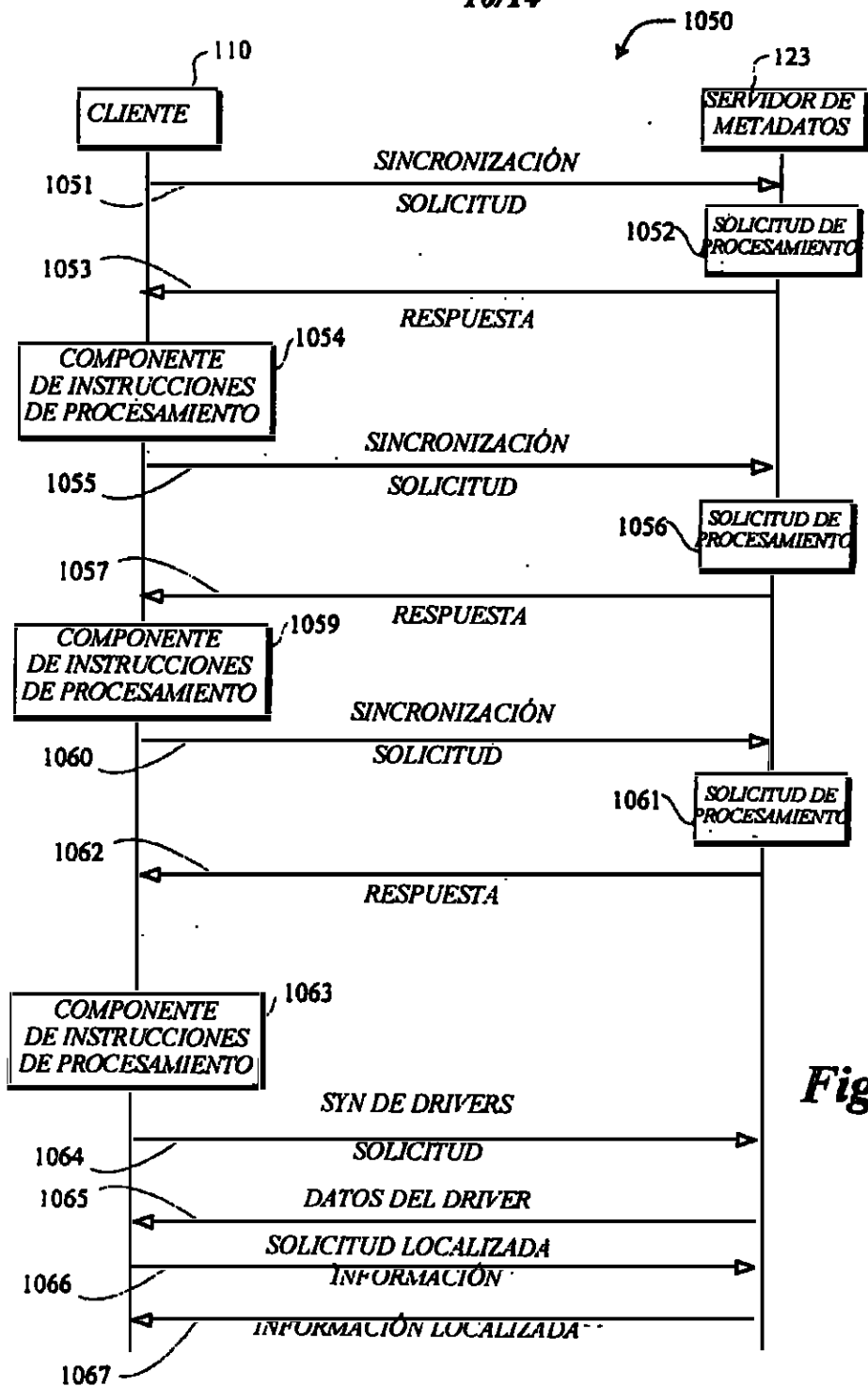


Fig.10

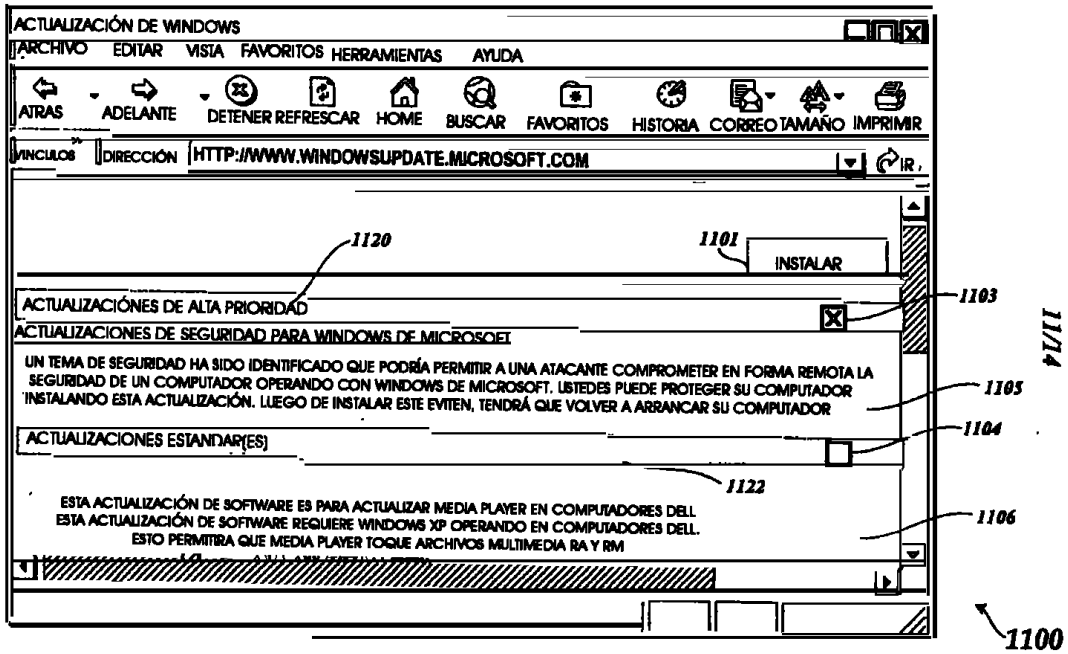


Fig. 11

pg

80

12/14

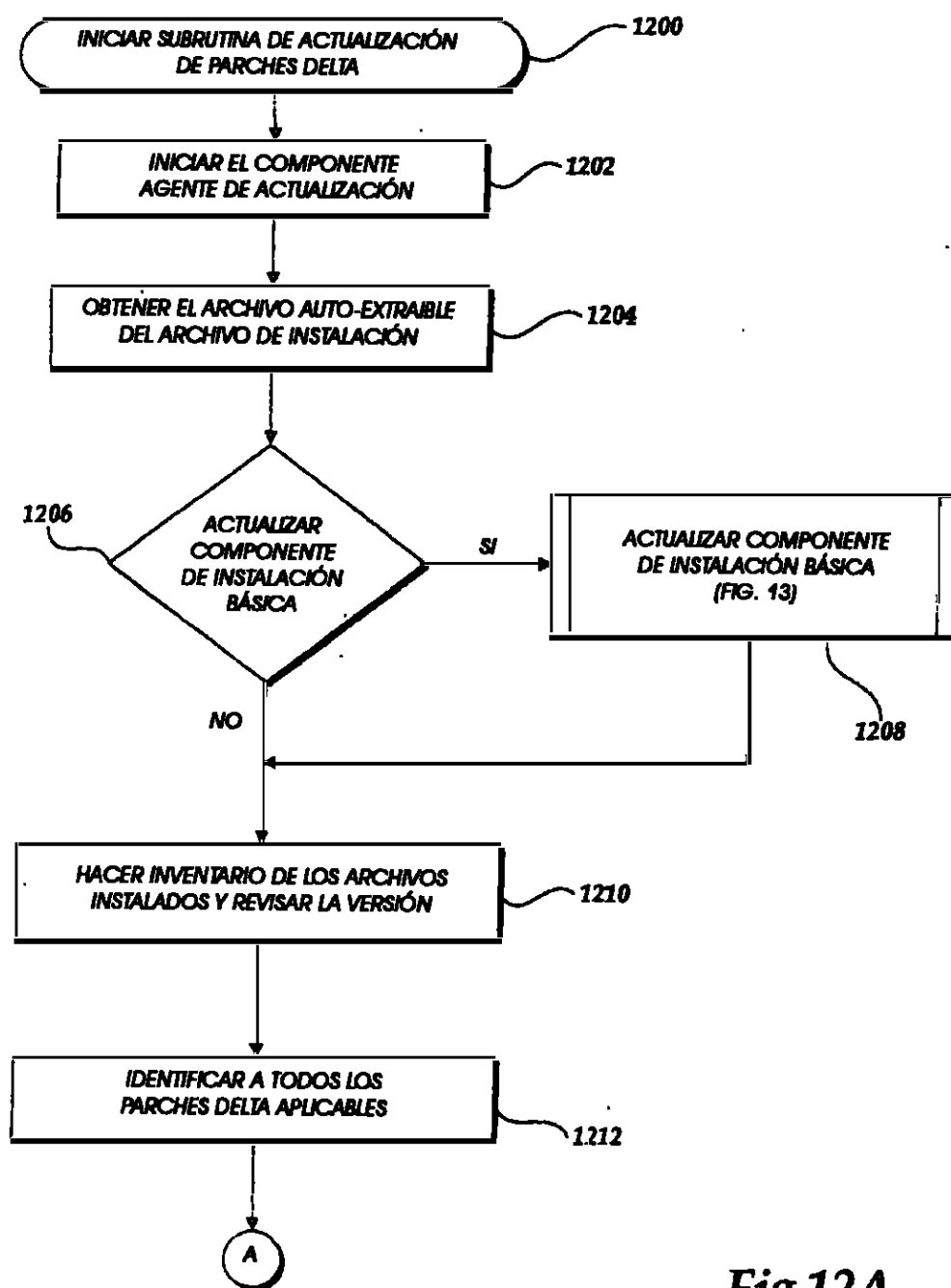
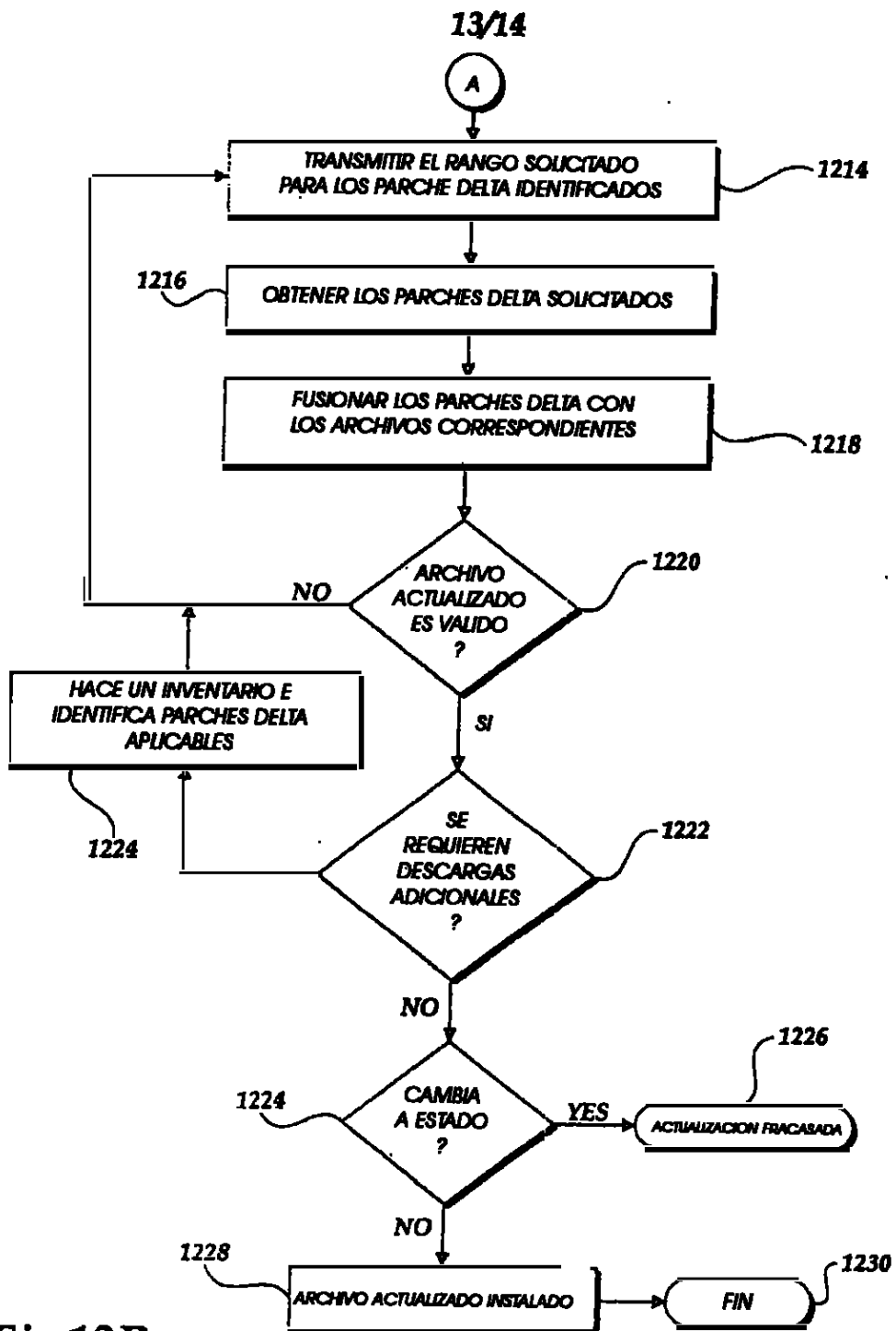


Fig.12A



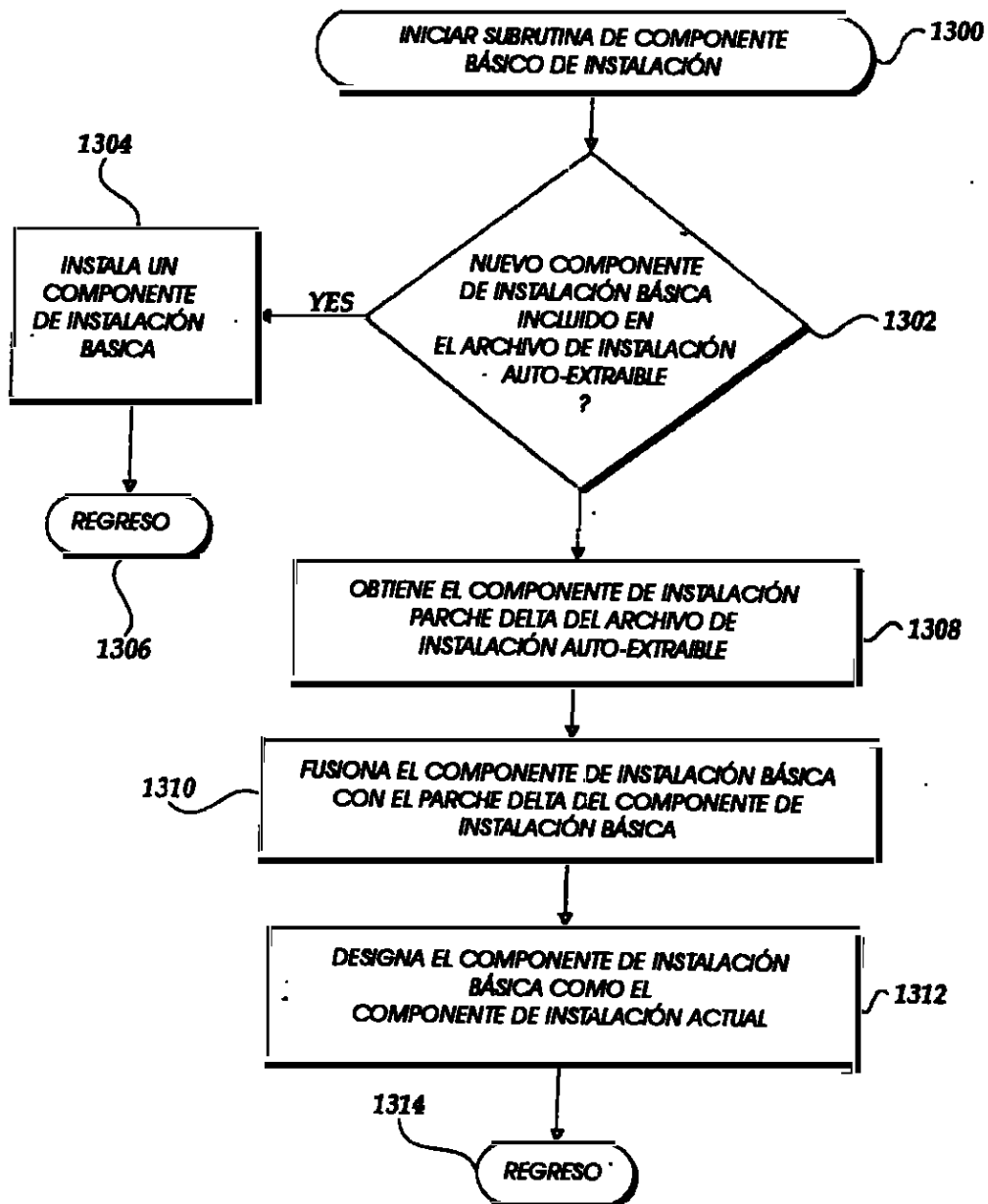


Fig.13

PATENT COOPERATION TREATY

From the RECEIVING OFFICE

25-APP
ADCT# 306474.26
PROSECUTION

PCT

NOTIFICATION OF THE INTERNATIONAL
APPLICATION NUMBER AND OF THE
INTERNATIONAL FILING DATE

(PCT Rule 20.5(c))

To: MAURICIO A. URIBE CHRISTENSEN O'CONNOR JOHNSON KINDNESS PLLC 1420 FIFTH AVENUE SUITE 2800 SEATTLE, WASHINGTON 98101-2347		Date of mailing (day/month/year) 18 Aug 2004
Applicant's or agent's file reference MSFTF123281		IMPORTANT NOTIFICATION
International application No. PCT/US2004/023975	International filing date (day/month/year) 23 Jul 2004	Priority date (day/month/year) 15 Dec 2003
Applicant MICROSOFT CORPORATION		
Title of the invention SYSTEM AND METHOD FOR MANAGING AND COMMUNICATING SOFTWARE UPDATES		

1. The applicant is hereby notified that the international application has been accorded the international application number and the international filing date indicated above.	
2. The applicant is further notified that the record copy of the international application: ☒ was transmitted to the International Bureau on 18 Aug 2004 ☐ has not yet been transmitted to the International Bureau for the reason indicated below and a copy of this notification has been sent to the International Bureau*: ☐ because the necessary national security clearance has not yet been obtained. ☐ because (reason to be specified):	
* The International Bureau monitors the transmittal of the record copy by the receiving Office and will notify the applicant (with Form PCT/IB/301) of its receipt. Should the record copy not have been received by the expiration of 14 months from the priority date, the International Bureau will notify the applicant (Rule 22.1(c)).	
3. FOREIGN TRANSMITTAL LICENSE INFORMATION Completed by: HS ☐ Additional license for foreign transmittal not required. This subject matter is covered by a license already granted or the equivalent U.S. national application. Refer to that license for information concerning its scope. ☐ License for foreign transmittal not required. 37 CFR 5.11(e)(1) or 37 CFR 5.11(e)(2). However, a license may be required for additional subject matter. See 37 CFR 5.15(b). ☒ Foreign transmittal license granted. 35 U.S.C. 184; 37 CFR 5.11 on 13 Aug 2004 (date) ☐ 37 CFR 5.15(a) ☒ 37 CFR 5.15(b)	
Name and mailing address of the receiving Office Mail Stop PCT, Commissioner for Patents P.O. Box 1450, Alexandria, VA 22313-1450 Facsimile No. 703-305-3230	Authorized officer Hallet Saunders Telephone No. 703-305-3653

PCT

REQUEST

The undersigned requests that the present international application be processed according to the Patent Cooperation Treaty.

For receiving Office use only

International Application No.

International Filing Date

Name of receiving Office and "PCT International Application"

Applicant's or agent's file reference (if desired) (12 characters maximum) MSFTF123281

Box No. I TITLE OF INVENTION
SYSTEM AND METHOD FOR MANAGING AND COMMUNICATING SOFTWARE UPDATES

Box No. II APPLICANT ☐ This person is also inventor

Name and address: (Family name followed by given name; for a legal entity, full official designation. The address must include postal code and name of country. The country of the address indicated in this box is the applicant's State (that is, country) of residence (if no State of residence is indicated below.)

MICROSOFT CORPORATION Microsoft Patent Dept.
One Microsoft Way
Redmond, WA 98052
US
DEC 13 2004

D PD ND By:

Telephone No.

Facsimile No.

Teleprinter No.

Applicant's registration No. with the Office

State (that is, country) of nationality:
US

State (that is, country) of residence:
US

This person is applicant for the purposes of: ☒ all designated States ☐ all designated States except the United States of ☐ the United States of America only ☐ the States indicated in the Supplemental Box

Box No. III FURTHER APPLICANT(S) AND/OR (FURTHER) INVENTOR(S)

Name and address: (Family name followed by given name; for a legal entity, full official designation. The address must include postal code and name of country. The country of the address indicated in this box is the applicant's State (that is, country) of residence (if no State of residence is indicated below.)

MEULEMANS, Michael Edward
17168 SE 100th Street
Renton, WA 98059
US

This person is:

☐ applicant only

☐ applicant and inventor

☒ inventor only (If this check-box is marked, do not fill in below.)

Applicant's registration No. with the Office

State (that is, country) of nationality:
US

State (that is, country) of residence:
US

This person is applicant for the purposes of: ☐ all designated States ☐ all designated States except the United States of ☐ the United States of America only ☐ the States indicated in the Supplemental Box

☒ Further applicants and/or (further) inventors are indicated on a continuation sheet.

Box No. IV AGENT OR COMMON REPRESENTATIVE; OR ADDRESS FOR CORRESPONDENCE

The person identified below is hereby/has been appointed to act on behalf of the applicant(s) before the competent International Authorities as: ☒ agent ☐ common representative

Name and address: (Family name followed by given name, for a legal entity, full official designation. The address must include postal code and name of country)

URIBE, Mauricio A.
Christensen O'Connor Johnson Kindness PLLC
1420 Fifth Avenue
Suite 2800
Seattle, WA 98101-2347
US

Telephone No.

206.682.8100

Facsimile No.

206.224.0779

Teleprinter No.

Agent's registration No. with the Office

46,208

☐ Address for correspondence: Mark this check-box where no agent or common representative is/has been appointed and the space above is used instead to indicate a special address to which correspondence should be sent.

Continuation of Box No. III FURTHER APPLICANT(S) AND/OR (FURTHER) INVENTOR(S)	
<i>If none of the following sub-boxes is used, this sheet should not be included in the request.</i>	
Name and address: (Family name followed by given name; for a legal entity, full official designation. The address must include postal code and name of country. The country of the address indicated in this Box is the applicant's State (that is, country) of residence if no State of residence is indicated below.) AVERBUCH, Aaron 2230 Yale Ave. E., #D Seattle, WA 98102 US	This person is: ☐ applicant only ☐ applicant and inventor ☒ inventor only (If this check-box is marked, do not fill in below.) Applicant's registration No. with the Office
State (that is, country) of nationality: US	State (that is, country) of residence: US
This person is applicant ☐ all designated States ☐ all designated States except the United States of ☐ the United States of America only ☐ the States indicated in the Supplemental	
Name and address: (Family name followed by given name; for a legal entity, full official designation. The address must include postal code and name of country. The country of the address indicated in this Box is the applicant's State (that is, country) of residence if no State of residence is indicated below.) ROBERTS, Jason 103 College Ave. #2 Somerville, MA 02144 US	This person is: ☐ applicant only ☐ applicant and inventor ☒ inventor only (If this check-box is marked, do not fill in below.) Applicant's registration No. with the Office
State (that is, country) of nationality: US	State (that is, country) of residence: US
This person is applicant for the ☐ all designated States ☐ all designated States except the United States of ☐ the United States of America only ☐ the States indicated in the Supplemental	
Name and address: (Family name followed by given name; for a legal entity, full official designation. The address must include postal code and name of country. The country of the address indicated in this Box is the applicant's State (that is, country) of residence if no State of residence is indicated below.) SHOWMAN, Ken 15724 NE 117th Street Redmond, WA 98052 US	This person is: ☐ applicant only ☐ applicant and inventor ☒ inventor only (If this check-box is marked, do not fill in below.) Applicant's registration No. with the Office
State (that is, country) of nationality: US	State (that is, country) of residence: US
This person is applicant for the ☐ all designated States ☐ all designated States except the United States of ☐ the United States of America only ☐ the States indicated in the Supplemental	
Name and address: (Family name followed by given name; for a legal entity, full official designation. The address must include postal code and name of country. The country of the address indicated in this Box is the applicant's State (that is, country) of residence if no State of residence is indicated below.) MOHAMMED, Mazhar 127 - 242nd Court SE Sammamish, WA 98074 US	This person is: ☐ applicant only ☐ applicant and inventor ☒ inventor only (If this check-box is marked, do not fill in below.) Applicant's registration No. with the Office
State (that is, country) of nationality: US	State (that is, country) of residence: US
This person is applicant for the ☐ all designated States ☐ all designated States except the United States of ☐ the United States of America only ☐ the States indicated in the Supplemental	
☒ Further applicants and/or (further) inventors are indicated on another continuation sheet.	

85

85

Continuation of Box No. III FURTHER APPLICANT(S) AND/OR (FURTHER) INVENTOR(S)	
<i>If none of the following sub-boxes is used, this sheet should not be included in the request.</i>	
Name and address: <i>(Family name followed by given name; for a legal entity, full official designation. The address must include postal code and name of country. The country of the address indicated in this box is the applicant's State (that is, country) of residence if no State of residence is indicated below.)</i> DADZIE, Joseph G. 9631 - 173rd Place NE Redmond, WA 98052 US	This person is: ☐ applicant only ☐ applicant and inventor ☒ inventor only <i>(If this check-box is marked, do not fill in below.)</i> Applicant's registration No. with the Office
State (that is, country) of nationality: US	State (that is, country) of residence: US
This person is applicant ☐ all designated States ☐ all designated States except the United States of ☐ the United States of America only ☐ the States indicated in the Supplemental	
Name and address: <i>(Family name followed by given name; for a legal entity, full official designation. The address must include postal code and name of country. The country of the address indicated in this box is the applicant's State (that is, country) of residence if no State of residence is indicated below.)</i>	This person is: ☐ applicant only ☐ applicant and inventor ☐ inventor only <i>(If this check-box is marked, do not fill in below.)</i> Applicant's registration No. with the Office
State (that is, country) of nationality:	State (that is, country) of residence:
This person is applicant for the ☐ all designated States ☐ all designated States except the United States of ☐ the United States of America only ☐ the States indicated in the Supplemental	
Name and address: <i>(Family name followed by given name; for a legal entity, full official designation. The address must include postal code and name of country. The country of the address indicated in this box is the applicant's State (that is, country) of residence if no State of residence is indicated below.)</i>	This person is: ☐ applicant only ☐ applicant and inventor ☐ inventor only <i>(If this check-box is marked, do not fill in below.)</i> Applicant's registration No. with the Office
State (that is, country) of nationality:	State (that is, country) of residence:
This person is applicant for the ☐ all designated States ☐ all designated States except the United States of ☐ the United States of America only ☐ the States indicated in the Supplemental	
Name and address: <i>(Family name followed by given name; for a legal entity, full official designation. The address must include postal code and name of country. The country of the address indicated in this box is the applicant's State (that is, country) of residence if no State of residence is indicated below.)</i>	This person is: ☐ applicant only ☐ applicant and inventor ☐ inventor only <i>(If this check-box is marked, do not fill in below.)</i> Applicant's registration No. with the Office
State (that is, country) of nationality:	State (that is, country) of residence:
This person is applicant for the ☐ all designated States ☐ all designated States except the United States of ☐ the United States of America only ☐ the States indicated in the Supplemental	
☐ Further applicants and/or (further) inventors are indicated on another continuation sheet.	

86

Box No.V DESIGNATIONS

The filing of this request constitutes under Rule 4.9(a), the designation of all Contracting States bound by the PCT on the international filing date, for the grant of every kind of protection available and, where applicable, for the grant of both regional and national patents.

However,

- ☐ DE Germany is not designated for any kind of national protection
- ☐ KR Republic of Korea is not designated for any kind of national protection
- ☐ RU Russian Federation is not designated for any kind of national protection

(The check-boxes above may be used to exclude (irrevocably) the designations concerned in order to avoid the ceasing of the effect, under the national law, of an earlier national application from which priority is claimed. See the Notes to Box No. V as to the consequences of such national law provisions in these and certain other States.)

Box No. VI PRIORITY CLAIM

The priority of the following earlier application(s) is hereby claimed:

Filing date of earlier application (day/month/year)	Number of earlier application	Where earlier application is:		
		national application: country or Member	regional application:* regional Office	international application receiving Office
item (1) 15 December 2003 (15.12.03)	10/737,708	US		
item (2)				
item (3)				

☐ Further priority claims are indicated in the Supplemental Box.

The receiving Office is requested to prepare and transmit to the International Bureau a certified copy of the earlier application(s) (only if the earlier application was filed with the Office which for the purposes of this international application is the receiving Office) identified above as:

☒ all items ☐ item (1) ☐ item (2) ☐ item (3) ☐ other, see Supplemental Box

* Where the earlier application is an ARIPO application, indicate at least one country party to the Paris Convention for the Protection of Industrial Property or one Member of the World Trade Organization for which that earlier application was filed (Rule 4.10(b)(ii)).

Box No. VII INTERNATIONAL SEARCHING AUTHORITY

Choice of International Searching Authority (ISA) (if two or more International Searching Authorities are competent to carry out the international search, indicate the Authority chosen; the two-letter code may be used):

ISA/US

Request to use results of earlier search; reference to that search (if an earlier search has been carried out by or requested from the International Searching Authority):

Date (day/month/year) Number Country (or regional Office)

Box No. VIII DECLARATIONS

The following declarations are contained in Boxes Nos. VIII (i) to (v) (mark the applicable check-boxes below and indicate in the right column the number of each type of declaration):

- | | | |
|---|--|---|
| ☐ Box No. VIII (i) | Declaration as to the identity of the inventor | : |
| ☐ Box No. VIII (ii) | Declaration as to the applicant's entitlement, as at the international filing date, to apply for and be granted a patent | : |
| ☐ Box No. VIII (iii) | Declaration as to the applicant's entitlement, as at the international filing date, to claim the priority of the earlier application | : |
| ☐ Box No. VIII (iv) | Declaration of inventorship (only for the purposes of the designation of the United States of America) | : |
| ☐ Box No. VIII (v) | Declaration as to non-prejudicial disclosures or exceptions to lack of novelty: | : |

Box No. IX CHECK LIST: LANGUAGE OF FILING	
This international application contains: (a) in paper form, the following number of sheets: request (including declaration sheets) : 5 description (excluding sequence listing and/or tables related thereto) : 35 claims : 8 abstract : 1 drawings : 14 Sub-total number of sheets : 63 sequence listing : : tables related thereto : : (for both, actual number of sheets if filed in paper form, whether or not also filed in computer readable form; see (c)) Total number of sheets : 63 (b) ☐ only in computer readable form (Section 801(a)(i)) (i) ☐ sequence listing (ii) ☐ tables related thereto (c) ☐ also in computer readable form (Section 801(a)(ii)) (i) ☐ sequence listing (ii) ☐ tables related thereto Type and number of carriers (diskette, CD-ROM, CD-R or other) on which are contained the ☐ sequence listing: ☐ tables related thereto: (additional copies to be indicated under items 9(ii) and/or 10(ii), in right column)	This international application is accompanied by the following item(s) (mark the applicable check-boxes below and indicate in right column the number of each item): 1. ☒ fee calculation sheet : 1 2. ☐ original separate power of attorney : : 3. ☐ original general power of attorney : : 4. ☐ copy of general power of attorney; reference number, if any: : : 5. ☐ statement explaining lack of signature : : 6. ☐ priority document(s) identified in Box No. VI as item(s): : : 7. ☐ translation of international application into (language): : : 8. ☐ separate indications concerning deposited microorganism or other biological material : : 9. ☐ sequence listing in computer readable form (indicate type and number of carriers) (i) ☐ copy submitted for the purposes of international search under Rule 13ter only (and not as part of the international application) : : (ii) ☐ (only where check-box (b)(i) or (c)(i) is marked in left column) additional copies including, where applicable, the copy for the purposes of international search under Rule 13ter : : (iii) ☐ together with relevant statement as to the identity of the copy or copies with the sequence listing mentioned in left column : : 10. ☐ tables in computer readable form related to sequence listing (indicate type and number of carriers) (i) ☐ copy submitted for the purposes of international search under Section 802(b-quater) only (and not as part of the international application) : : (ii) ☐ (only where check-box (b)(ii) or (c)(ii) is marked in left column) additional copies including, where applicable, the copy for the purposes of international search under Section 802(b-quater) : : (iii) ☐ together with relevant statement as to the identity of the copy or copies with the tables mentioned in left column : : 1. ☒ other (specify): Transmittal Letter, Check and Postcard : :
Figure of the drawings which should accompany the abstract: 12a	Language of filing of the international application: English
Box No. X SIGNATURE OF APPLICANT, AGENT OR COMMON REPRESENTATIVE Next to each signature, indicate the name of the person signing and the capacity in which the person signs (if such capacity is not obvious from reading the request).  Mauricio A. Uribe Agent for Applicant Reg. No. 46,206	

For receiving Office use only	
1. Date of actual receipt of the purported international application:	2. Drawings: ☐ received: ☐ not received
3. Corrected date of actual receipt due to later but timely received papers or drawings completing the purported international application:	
4. Date of timely receipt of the required corrections under PCT Article 11(2):	
5. International Searching Authority (if two or more are competent): ISA/	6. ☐ Transmittal of search copy delayed until search fee is paid
For International Bureau use only	
Date of receipt of the record copy by the International Bureau:	

SYSTEM AND METHOD FOR MANAGING
AND COMMUNICATING SOFTWARE UPDATES

FIELD OF THE INVENTION

5 The present invention relates to software and computer networks and, in particular, the present invention relates to a system and method for managing and communicating software updates.

BACKGROUND OF THE INVENTION

10 Most commercially available software products undergo a continual revision process to repair or upgrade features and/or functions. Each revision of a software product or component may require the addition of new files and/or the replacement of existing files with files of newer versions. Once a vendor has isolated a software product problem and created a solution for the problem, it would want to put that fix into an update and make the update widely available to the customers. Software vendors have a
15 business incentive to distribute software updates to customers as quickly and trouble-free as possible.

20 The Internet provides an important channel for customers to obtain the latest updates for software products. The explosive growth of Internet usage has created a common expectation by customers that software products and updates be provided online for downloading. It is also in the interest of software vendors to promote the use of the Internet to distribute updates, because it reduces their costs and allows customers to obtain the fix for an identified problem as soon as the fix is made available for downloading. The vendor sites on the Internet can be designed to make it very simple to discover and locate update files for an application. The technical aspects of file
25 download have mostly disappeared from the user's view, and are now typically handled by the operating system.

30 In a conventional approach, a software vendor constructs a software update as a "package" for download. This package is typically a self-extracting executable file with the setup program and each of the product's updated files embedded and compressed to make the package smaller. The size of the package is generally the sum of the compressed sizes of each changed file, plus the size of the extraction code itself. Upon execution, the package extracts each of the contained files to a temporary location, then starts the setup program to install each file to a proper location in the system's directory.

Files that are shipped in a compressed form are decompressed as they are installed. Any existing file of the same name in the same location would simply be overwritten by the replacement file.

Even though the Internet makes wide and quick distribution of software updates possible, the limited bandwidth of network transmission has caused problems. The sheer sizes of common software applications have caused the download sizes of updates to become unreasonably large. Usually a multitude of fixes for a variety of problems of a product will be grouped into an update. If a vendor updates a software product on a regular basis, the download size of the update package will continue to grow, because the vendor cannot omit files under the assumption that the user already has those files from earlier updates. Because the update package combines a number of whole files, it may be quite large, even when the files are compressed. Sometimes, even on the fastest modem connections, the bandwidth efficiency of the download is decreased.

The time-consuming aspect of the conventional downloading process is, of course, undesirable. In some cases, customers pay long-distance or connection time charges during these file downloads. Any reductions in connection time will reduce the direct monetary cost for these customers. The vendors typically also have some distinguishable costs relating to the sizes of downloads they provide, so reducing the sizes may give them direct monetary benefits as well. Reducing the sizes of downloads will increase their available network bandwidth, allowing them to serve more customers with existing network server equipment.

The long time it takes to download a large update also makes the downloading process more vulnerable to various network connection problems. There are a number of reasons why an Internet session might be disconnected prematurely, including telephone line noise, call-waiting signals, and unintentional commands. Some Internet service providers enforce a connection time limit, limiting the amount of time the user can be on-line in a single session. If the user is downloading a large file when the network connection is cut off, he or she may have to start over. Most common operating systems and file transfer protocols do not allow the file transfer to be resumed, so any interim progress would be lost, and the transfer would have to be restarted. The opportunities for failure are so numerous that many users find it nearly impossible to obtain the update online. If the size of an update package is too large, users may never be able to completely download it.

One attempt to reduce the size of software updates and increase bandwidth efficiency relates to the use of delta patches, or binary patches. One skilled in the relevant art will appreciate that a delta patch corresponds to specialized software code that modifies an existing file when executed by a computing device. Because the delta patch includes specialized software code, a unique delta patch is required for each unique version of a file. As applied to software updates, a software update service can transmit a smaller sized update delta patch instead of transmitting a complete, updated file. The updated delta patch is then utilized to modify the existing file into the updated file.

Although the update delta patches can potentially reduce the amount of data required to update files, current approaches to delta patching are deficient in managing the selection of applicable delta files in situations where a large number of versions of a file exist. Because a unique delta patch is required for each version of a file, typical software update systems can often require hundreds, if not thousands, of unique delta patches to correspond to each unique version of a file. In one approach, some update services supporting delta patching transmit all possible delta patches to a client computing device. However, this approach typically increases the amount of data required to implement the software update as the number of possible update delta patches increase. Accordingly, the number of potentially applicable delta patches can quickly grow to the same size as the complete updated file. In another approach, a networked update software service scans a client machine to select which delta patch may be applicable for each client machine. Although this reduces the amount of delta patch information that is transmitted, it requires additional logic on the software update service to scan the client machines and select an applicable delta patch. The use of the additional logic increases the system resources that must be provided by the service. Further, this approach typically prevents the utilization of network caching, such as is typically achieved by traditional Web servers.

In addition to the above described shortcomings, existing systems are unable to deliver certain types of software updates, such as hardware drivers. As known in the art, specialized software updates, such as those that apply to hardware drivers, are difficult to provide to users on a large-scale distribution because most specialized software updates will only work on client computers with specific hardware. In most cases, for example, if a client computer obtains an incompatible hardware driver upgrade, the installation of the driver upgrade may cause a fatal error, or even prevent the computer from operating.

92

As will be readily understood from the foregoing, there is a need for a system and method having improved communication of software updates between a server and a number of clients. In addition, there exists a need for a software update system and method having improved mechanisms for allowing update services to target specific types of clients when delivering specialized updates.

SUMMARY OF THE INVENTION

The present invention is directed to a system and method for managing software updates. More specifically, the present invention is directed to a system and method for facilitating the selection and implementation of software updates while minimizing the bandwidth and processing resources required to select and implement the software updates. In accordance with an aspect of the present invention, a system and component architecture for processing the software updates is provided.

In accordance with an aspect of the present invention, a method for communicating software updates from the software update service to the client computing device is provided. The method may be implemented in a computer system including a client computing device in communication with a software update service. In accordance with the method, a software update service obtains a synchronization request from a client computing device. The synchronization request includes an identifier of an installed software update if the client computing device stores the installed software update. The software update service determines if the synchronization request comprises an identifier of at least one installed software update. If the software update service determines that the synchronization request comprises an identifier of at least one installed software update, the software update service selects an additional software update for communication to the client computing device. The selection of the additional software update is dependent on the fulfillment of a prerequisite defined in the additional software update, wherein the prerequisite requires the synchronization request to include an identifier for at least one installed software update. Alternatively, if the software update service determines that the synchronization request does not comprise an identifier of at least one installed software update, the software update service selects a first level software update for communication to the client computing device, wherein the first level

92

software update does not include a prerequisite. The software update service then communicates an instruction component of the selected software update to the client computing device from the software update service. The software update service then stores the instruction components of the selected software update in the client computing device as an installed software update if the client computing device contains at least one component that fulfills a condition of an applicability rule stored in the selected software update.

In accordance with another aspect of the present invention, a method for communicating software updates from the software update service to the client computing device is provided. The method may be implemented in a computer system including a client computing device in communication with a software update service. In accordance with the method, a software update service obtains an authorization request from the client computing device. The authorization request contains an identifier of a client authorization module stored on the client computing device. The software update service determines if the client computing device is associated with a target group. The software update service determines that the client computing device is associated with the target group if the client authorization module indicates that the client computing device is associated with the target group. The software update service communicates a server cookie from the software update service to the client computing device if it is determined that the client computing device is associated with the target group, wherein the server cookie identifies the target group. The software update service obtains a request for at least one software update stored on the software update service, wherein the request contains the server cookie. In response to obtaining the request, the software update service determines if a software update is associated with the target group identified in the server cookie. Additionally, if the software update service determines that the software update is associated with the target group identified in the server cookie, the software update service communicates the software update from the software update service to the client computing device.

94

In accordance with a further aspect of the present invention, a method for communicating a data structure from the server to the client computing device is provided. The method may be implemented in a computer system including a client computing device in communication with a server. In accordance with the method, a software update service determines whether the client computing device is associated with a target group if the client computing device contains an authorization module indicating that the client computing device is associated with the target group. The software update service communicates a server cookie from the server to the client computing device if it is determined that the client computing device is associated with the target group. The server cookie identifies the target group. The software update service obtains a request for at least one data structure stored on the server. The request contains the server cookie. In response to obtaining the request, the software update service determines if a data structure is associated with the target group identified in the server cookie. If the software update service determines that the data structure is associated with the target group identified in the server cookie, the software update service communicates the data structure from the server to the client computing device.

BRIEF DESCRIPTION OF THE DRAWINGS

The foregoing aspects and many of the attendant advantages of this invention will become more readily appreciated as the same become better understood by reference to the following detailed description, when taken in conjunction with the accompanying drawings, wherein:

FIGURE 1 is a block diagram of a software update system, including a client computer and an update service providing update software in accordance with the present invention;

FIGURE 2 is a block diagram of the software update system of FIGURE 1 illustrating the authentication of a client computing device with the update service in accordance with the present invention;

FIGURE 3 is a block diagram of the software update system of FIGURE 1 illustrating the synchronization of available updates between a client computing device and the update service in accordance with the present invention;

94

FIGURE 4 is a block diagram of the software update system of FIGURE 1 illustrating the transmission of software update information to a client computing device from the update service in accordance with the present invention;

FIGURE 5 is a block diagram of the software update system of FIGURE 1 illustrating the processing and selection of update information by a client computing device in accordance with the present invention;

FIGURE 6 is a block diagram of the software update system of FIGURE 1 illustrating the merging of delta patches and installation of updated files by a client computing device in accordance with the present invention;

FIGURE 7 is a flow diagram illustrative of a software update routine implemented by a client computing device and an update service for identifying software updates available for installation on the client computing device, in accordance with the present invention;

FIGURE 8 is a protocol diagram of an authorization routine for providing selective access to updates stored on the update service, in accordance with the present invention;

FIGURE 9 is a block diagram of an example set of software updates illustrating an authorization routine, in accordance with the present invention;

FIGURE 10 is a protocol diagram of a synchronization routine for communicating a select group of software updates from a software update service to a client computing device, in accordance with the present invention;

FIGURE 11 is a block diagram illustrating an exemplary section of a graphical user interface for displaying a list of software updates that are available to an individual client computing device, in accordance with the present invention;

FIGURES 12A and 12B are illustrative of a software update processing sub-routine 1200 implemented by the client computing device 110 to retrieve and install requested software in accordance with the present invention; and

FIGURE 13 is a flow diagram illustrative of a sub-routine implemented by a client computing device for updating a baseline installation component in accordance with the present invention.

DETAILED DESCRIPTION OF THE PREFERRED EMBODIMENT

Generally described, the present invention is directed to a system and method for managing software updates. More specifically, the present invention is directed to a

system and method for facilitating the selection and implementation of software updates while minimizing the bandwidth and processing resources required to select and implement the software updates. In accordance with the present invention software updates can correspond to updates for specific software applications or operating systems.

5 Further, software updates can include software drivers or updates to firmware, such as system BIOS. In accordance with an aspect of the present invention, a system and component architecture for processing the software updates is provided. In accordance with another aspect of the present invention, an update protocol and interface to facilitate the authorization and synchronization of client machines with an update service is

10 provided. In accordance with a further aspect of the present invention, a method for updating an installation component and various installed files utilizing delta patches is provided. One skilled in the relevant art will appreciate, however, that additional aspects of the present invention may also be provided in the present application. Further, one skilled in the relevant art will appreciate that each identified aspect may be considered

15 individually or as part of common inventive aspect.

FIGURE 1 software update system 100 is a block diagram illustrative of software update system 100 in accordance with the present invention. Generally described, the software update system 100 may comprise one or more client computing devices 110, an update service 120 and an external update provider 130. Generally described, the update

20 service 120 stores and manages the distribution of software updates that are communicated to and installed on the client computing device 110. The software updates may be provided by the update service 120 or by any number of external update providers 130.

The client computing device 110, the update service 120, and the external update

25 provider 130 electronically communicate via a network 101. The network may be a local area network (LAN) or a larger network, such as a wide area network (WAN) or the Internet. By the use of generally known software, the software update system 100 may be configured to exchange documents, commands, and other known types of information between the client computing device 110 and the servers 121, 122, 123 and 124 of the

30 update service 120. As will be appreciated by those skilled in the art and others, the software update system 100 shown in FIGURE 1 is a simplified example of one suitable system for implementing the present invention and that the present invention is not limited to this example.

As will be described in more detail below, one embodiment the update service 120 comprises a number of servers. As shown in FIGURE 1, the update service 120 includes an update server 121 for managing the overall processes of the update service 120 and coordinating processes of the servers 121, 122, 123 and 124 of the update service 120. The authorization server 122 generates authorization cookies as requested by the client and, in turn, the authorization cookies are used to generate server cookies, that allow client computers to access updates provided by the update service 120. The metadata server 123 provides general information regarding the updates provided by the update service 120. The metadata server 123 allows the system of the present invention to identify specific updates for a specific type of client computer or a specific group of client computers. The download server 124 provides one or more software components for delivering data files associated with software updates provided by the update service 120.

The external update provider 130 may include one or more servers that distribute software updates. The external update provider 130 may be associated with an entity that provides software, software updates, or other data that is to be distributed to groups of client computers. For example, the external update provider 130 may be associated with a third party software developer desiring to use the update service 120 to distribute updates for one or more software applications. In another example, the external update provider 130 may be associated with software update system 120.

The client computing device 110 may be any computing device that stores and executes software applications 114. The client computing device 110 may be formed from any one of a number of different computer products including, but not limited to, personal computers (PCs), personal digital assistants (PDAs), mobile telephones, two-way pagers, etc. As will be appreciated by those of ordinary skill in the art or others, the architecture of the client computing device 110 may take on any suitable form. For example, the client computing device 110 may include a network interface for providing communication with the network 101. The network interface may be configured for use with any wired or wireless network connection, and may be used with any suitable communication protocol, such as the TCP/IP protocol. In addition, the client computing device 110 may include a processing unit, a display, and a memory unit. The memory unit may store the program code necessary for operating the client computing device 110, such as an operating system 116. In addition, the memory unit stores an update

management component 112 for controlling and executing processes of the present invention.

The software update system 100 stores software programs that, when executed, implement the present invention. When executed, the software update system 100 stores, manages, and selectively communicates software updates. As described more fully below, among many other benefits, the present invention provides a mechanism for defining and selecting target groups of client computing devices that are eligible to receive software updates. The present invention also provides an improved mechanism for downloading data files associated with software updates.

For purposes of illustrating the present invention, a detailed description of a working example of the present invention is provided. In describing the working example, reference is made to software updates that may refer to a specific upgrade of a software application, e.g., an upgrade of a media player version 6.0 to media player version 7.0. As will be appreciated by those of ordinary skill in the art, such a software update may include the communication and installation of a number of data files associated with the software update. Thus, for purposes of illustrating the present invention, a distinction is made between a software update and an individual data file containing a software update.

With reference now to FIGURES 2-6, an illustrative interaction between the components of the software update system 100 to update one or more files on the client computing device 110 will be described. With reference to FIGURE 2, the software update service is initiated by the transmission of software update information by one or more external update providers 130. As described above, the external update providers 130 may be associated with software update system 100. Alternatively, the software update information may be transmitted by third-party external update providers 130. In an illustrative embodiment of the present invention, the software update information can include software code utilized to update a file, software code utilized to replace a file, various rules for determining the applicability of the software updates, and/or display information describing the software update. The transmission of the software update information may be completed at any time and does not have to be contemporaneous with the initiation of the other illustrated software update component interactions.

Upon receipt of the software update information from the external update provider 130, the update service 120 generates one or more pieces of data to facilitate the transmission of update information.⁷ The data can include a patch storage file that corresponds to set of software delta patches for updating different versions of a file. The data can also include a patch storage manifest that corresponds to an index mapping particular file versions to a corresponding delta found in the patch storage file.⁸ The data can further include a self-extracting file that corresponds to information the update agent will utilize to request and install specific software update data, as will be described in greater detail below. One skilled in the relevant art will appreciate that the generation of the patch storage file, patch storage manifest, and the self-extracting files may be completed at any time and does not have to be contemporaneously with the other illustrated component interactions.

To initiate the transmission of software update information to clients, a client computing device 110 initiates an authentication request to the update service 120. In an illustrative embodiment of the present invention, the authentication request corresponds to an update protocol interaction between the client computing device 110 and the update service 120, which will be described in greater detail below. Upon completion of the authentication, the update service 120 transmits an authentication cookie to the client computing device 120. With reference now to FIGURE 3, the authenticated client computing device 120 then initiates the synchronization of the available updates with the update server 120. In an illustrative embodiment of the present invention, the synchronization request also corresponds to the update protocol interaction between the client computing device 110 and the update service 120, which will be described in greater detail below. Upon the completion of the synchronization, the client computing device 110 receives the information of all applicable software updates and information describing the updates. However, in an illustrative embodiment of the present invention, no software code to instantiate the update has been downloaded.

With continued reference to FIGURE 3, at sometime during the update process, a selection of the updates to be installed is received. In an illustrative embodiment of the present invention, a user may be presented with software update information received during synchronization and asked to select an appropriate update. Alternatively, the client computing device 110 may be configured in a manner to automatically select all applicable software updates. Further, the client computing device 110 may also have

some rules that allow it to automatically select a subset of the available software updates. Still further, a user may initiate a selection of an update by communicating with the update service 120, such as via an internet Web page.

With reference now to FIGURE 4, the update management component 112
5 instantiates an update agent 118 on the client computing device 110, if an update agent is not already present. The update agent 118 then requests the transmission of a software update information package, such as self-extracting file. The update agent 118 receives the self-extracting file and performs any updates to the installer, as will be described below. Further, the update agent 118 can request any missing or corrupted information
10 from the update service 120.

With reference now to FIGURE 5, once the update agent 118 receives the software update information package, the update agent 118 performs an inventory of the files that are installed on the client computing device 110. Based on a comparison of the inventory and the software update information package, the update agent 118 determines
15 which delta patch, or other update information, will be required to complete the selected updates. The update agent 118 then transmits a request for specific delta updates. In one embodiment of the present invention, the request for software updates may correspond to a direct request transmitted via a direct network connection, which will be referred to as a manual update. In another embodiment of the present invention, the request for software
20 updates may be a background request that is transmitted without requiring overt user action. This embodiment will be referred to as an automatic update.

In an illustrative embodiment of the present invention, if the software update corresponds to a delta patch, the update agent 118 transmits a request to the update service 120 that identifies the particular delta patch identified by the patch storage
25 manifest. Alternatively, in the event that a delta patch is unavailable or if several delta patches have failed, the update agent 118 can initiate a fallback procedure. The fallback procedure can include a request for the transmission of a complete copy of the entire updated file from the patch storage file. The fallback procedure can also include a request for the transmission of a complete copy of the entire updated file from in
30 self-contained package.

In an illustrative embodiment of the present invention, the download server 124 of the update service 120 can directly process the software update request from the update agent 118. Alternatively, the request can also be processed by any number of additional

external download servers such as traditional Web servers that have either received the requested update delta patches from the update service 120. For example, a corporation may utilize an internal server to update client machines. Additionally, the request can be processed by external download servers in which some, or all, of the update delta patches are cached in processing previous requests. Accordingly, in this embodiment, the download can be distributed to a number of additional download servers capable of servicing hyper text transfer protocol ("HTTP") data requests.

With reference to FIGURE 6, once the software update information is received, the update agent 118 merges the delta patch with the installed file to generate an updated file. Additionally, the update agent 118 can validate whether the merger successfully updated the appropriate file. As described above, if a delta patch cannot be validated, the update agent 118 may request the delta patch again or request an entire updated file after a number of failures. Once the update agent 118 obtains the validated and update file, the file is installed on the client computing device 110.

FIGURE 7 is a flow diagram of a software update processing routine 700 illustrating the interaction between a client computing device 110 and the software update service 120 in accordance with the present invention. At block 702, the software update service 120 authorizes access to the client computer 110. In an illustrative embodiment of the present invention, the authorization of access to the client computer can include the generation of a server-issued cookie for allowing access to software updates that are associated with a particular group of computers. A more detailed explanation of the authorization process will be described with regard to FIGURE 8.

At block 704, the client computer 110 and the software update service 120 synchronize update information. In an illustrative embodiment of the present invention, the software update service 120 transmits metadata describing specific software updates to the client computing device 110. The metadata contains information describing the available software updates to allow a user to select one or more updates for installation. A more detailed description of the synchronization process will be described below with regard to FIGURES 9 and 10. At block 706, the client computing device 110 obtains a selection of applicable updates to download. In an illustrative embodiment of the present invention, the selection of applicable updates can correspond to the utilization of a number of unique user interfaces to facilitate user selections. The selection of user interfaces will be described in greater detail with regard to FIGURE 11.

At block 708, the client computing device 110 processes the user selection of applicable software updates and interfaces with the software update service 120 to request specific update information. In an illustrative embodiment of the present invention, the client computing device 110 selects and requests one or more applicable update delta patches. The update agent 118 on the client computing device 110 can then process the requested data to implement the selected software update. At block 710, the routine 700 terminates.

With reference to FIGURE 8, a protocol diagram 800 for authorizing access to client computing devices 110 and corresponding to block 702 (FIGURE 7) will now be described. In an illustrative embodiment of the present invention, the software update service 120 utilizes an extensible targeting mechanism to control client computing device 110 access to updates and other software. The software update service 120 incorporates a mechanism that associates specific software updates with one or more target groups of client computing devices 110. For example, the software update service 120 may limit access of a specific hardware driver update to a particular brand of client computing devices 110 having a specific hardware device. In such an example, the software update service 120 may define a target group of client computing devices 110 having a particular brand name and a specific hardware device and limit the transmission of the particular software download to the target group.

In an illustrative embodiment of the present invention, the extensible targeting mechanism is facilitated by the use of software components ("authorization plug-ins") that define a client computing device's membership to one or more target groups. The presence of an authorization plug-in on a client computing device 110 defines whether the client computing device belongs to the specific target group of the authorization plug-in. A target group, for example, may include all computers having a valid product identification ("PID") number for a particular software application. In such an example, as described in more detail below with respect to FIGURE 8, an authorization plug-in 826 may be installed in the client to read a PID from a memory module of the client computing device and pass the obtained PID to a corresponding PID server plug-in 829. The corresponding PID plug-in, also referred to herein as PID validator 829, utilizes one or more methods to determine if the received PID is valid. Once it is determined that the PID stored on the client computing device 110 is valid, the server generates a server cookie which indicates that the client computing device 110 is a member of a target group

having a valid PID. In another example, a target group may include client computing devices that are designated as beta test computers.

In an illustrative embodiment of the present invention, the authorization server 122 of the software update service 120 contains a number of server authorization plug-ins that define a set of target groups of client computing devices that the authorization server will recognize. Each server authorization plug-in contains components for communicating data with a corresponding client authorization plug-in stored on a client computing device 110. In a similar manner, each client computing device 110 includes one or more client authorization plug-ins that identifies the target groups to which the client belongs. In an illustrative embodiment of the present invention, the client authorization plug-ins can be installed in each client computing device during the installation or upgrade of a software application, such as the installation or upgrade of an operating system. Additionally, the server authorization plug-ins may be dynamically installed or removed by an administrator desiring to control the access to software updates. The authorization plug-ins stored on the client computing device 110 and the authorization server 122 can be an actual software plug-in, or the authorization plug-ins can be hard coded into dynamically linked libraries.

As shown in FIGURE 8, the authorization server 122 contains three example server authorization plug-ins: (1) a first server authorization plug-in 828 defining a target group that includes all computers (hereinafter the "All Computers target group"); (2) a second server authorization plug-in 829 defining a target group that includes computers having a valid PID (hereinafter the "PID target group"); and (3) a third server authorization plug-in 830 defining a target group that includes beta test computers (hereinafter the "beta target group"). Also shown in FIGURE 8, the client computing device 110 contains two client authorization plug-ins: (1) a first client authorization plug-in 825 indicating that the client computing device 110 is a member of the All Computers target group; and (2) a second client authorization plug-in 826 indicating that the client computing device 110 is a member of the PID target group. In this example, the client computing device 110 does not contain an authorization plug-in indicating that it is a member of the beta target group. As will be appreciated by those of ordinary skill in the art, each client authorization plug-in 825 and 826 may be configured to execute one or more functions on the client computing device 110 to assist the validation process. For instance, the second client authorization plug-in 826 may be configured to examine the

104

memory of the client computing device 110 to verify or obtain a PID for an installed software application.

As shown in FIGURE 8, the authorization sub-routine 702 begins when the client computing device 110 communicates a configuration request 803 to the authorization server 122. In an illustrative embodiment of the present invention, the configuration request 803 is formed from any suitable software component configured to obtain information describing the authorization plug-ins stored on the authorization server 122. As will be appreciated by those skilled in the art, the configuration request 803 may utilize a known method referred to as "GetConfig." In response to receiving the configuration request 803, the authorization server 122 communicates a configuration response 804, which includes information identifying all of the authorization plug-ins stored on the authorization server 122. In one embodiment, the configuration response 804 includes an array of strings that identifies and describes all authorization plug-ins stored on the authorization server 122. In the present example, the configuration response 804 includes information that identifies the first server authorization plug-in 828, the second server authorization plug-in 829, and the third server authorization plug-in 830.

At block 805, the client computing device 110 generates one or more authorization cookies in response to receiving the configuration response 804. In the process of block 805, the client computing device 110 generates an authorization cookie for each pair of matching client and server authorization plug-ins. Thus, in the present example, the first client authorization plug-in 825 generates a first authorization cookie associated with the All Computers target group because the first client authorization plug-in 825 and the first server authorization plug-in 828 are both associated with the All Computers target group. In addition, the second client authorization plug-in 826 generates a second authorization cookie associated with the PID target group because the second client authorization plug-in 826 and the second server authorization plug-in 829 are both associated with the PID target group. A third authorization cookie is not generated since the client computing device 110 does not have an authorization plug-in indicating that it is a member of the beta target group.

As will also be appreciated by those skilled in the art, one implementation of the process of block 805 may include the use of a generally known software method referred to in the art as "GetAuthCookie." It will also be appreciated that the generation of each

104

105

authorization cookie may involve additional processing. For instance, the second client authorization plug in 826 may be configured to examine information stored in the client's system registry to retrieve a PID and include the PID in the authorization cookie. In other examples, the process of block 805 may include processes for communicating with other computers or devices. For instance, a client authorization plug-in may communicate with a device, such as a sound card, scanner, video card, etc., to obtain the make and model of the device. In other non-limiting examples, a client authorization plug-in may communicate with a security device, such as a fingerprint reader, to obtain information describing a user.

10 In general, a client authorization plug-in may read configuration information from any component of the client computing device 110 or any other computing device communicatively connected to the client computing device 110. In other examples, a client authorization plug-in may be configured to utilize one or more public or private application programming interfaces (APIs) to gather and encrypt information from the client that will be validated by the corresponding server plug-in. In such examples, PID validator plug-in 826 uses a private API to encrypt the client's PID to pass the encrypted PID to the server for decryption and validation. In other embodiments, other client authorization plug-ins may utilize biometric measurements, such as fingerprint readers or voice prints, to construct an authorization cookie to be passed to the server for validation.

20 In yet another example, a client authorization plug-in may call a Web service or any other service to communicate authorization credentials or any other type of data to the authorization server 122.

In an illustrative embodiment of the present invention, each authorization cookie includes a string that identifies an associated target group. For example, a string may indicate that a particular authorization cookie is associated with the PID target group. Each authorization cookie also includes a data section for communicating data between a client and a server. For example, an authorization cookie associated with the PID target group may have a data section that contains an actual PID. As will be appreciated by those of ordinary skill in the art, the data section can contain any type of data that is stored in any format, such as a byte array. For example, if plug-ins on the client and the server require the communication of public and private keys, such data can be encrypted in the data section of one or more authorization cookies.

705

Once the client computing device 110 generates an authorization cookie for each pair of corresponding client and server authorization plug-ins, the client computing device communicates the generated authorization cookies to the authorization server 122. As shown in FIGURE 8, the client computing device 110 communicates the authorization cookies in a cookie request 806. The cookie request 806 includes any suitable format for communicating an array of the authorization cookies generated in the process of block 805. One implementation of this part of the authorization method 702 may include the use of a generally known software method referred to in the art as "GetCookie."

In one embodiment, the cookie request 806 also includes other authorization server cookies stored in the memory of the client computing device 110. As will become more readily understood from the following description, the memory of the client computing device 110 may store old authorization server cookies that were created in previous executions of the authorization routine 700. By providing the stored authorization server cookies in the cookie request 806, the client computing device 110 will be able to maintain its access privileges that were granted in the previous executions of the authorization sub-routine 702. In the present example, since there are no authorization server cookies stored in the client, the cookie request 806 includes the first authorization cookie associated with the All Computers target group and the second authorization cookie associated with the PID target group.

Next, as shown in block 807, in response to receiving the cookie request 806, the authorization server 122 generates a server cookie. In one embodiment, for each of the received authorization cookies, a call is made to an appropriate server authorization plug-in to generate server cookie data. The server cookie data generated by each server authorization plug-in includes an identifier for each target group identified in the received authorization cookies. In the present example, since the cookie request 806 includes the first authorization cookie associated with the All Computers target group and the second authorization cookie associated with the PID target group, the authorization server 122 generates server cookie data containing an identifier for these respective target groups. On the authorization server 122, the server cookie data is then combined with data of old server cookies, if an old server cookie is received in the cookie request 806, to generate a new server cookie. In one embodiment, the new server cookie is encrypted by the use of a publically available encryption method, such as Triple DES.

107

In an illustrative embodiment of the present invention, the server cookie can include encrypted information that identifies one or more associated target groups. In addition, the server cookie can include expiration data, which is stored in both a clear text format and an encrypted format. The expiration data stored in the clear text format is used by the client computing device 110 for monitoring for the expiration of the server cookie. The expiration data stored in the encrypted format is used by software update service 120 to determine if the client computing device 110 is authorized to receive updates associated with a particular target group. In one embodiment, the expiration data of the server cookie applies to all target groups identified in the server cookie. Alternatively, or in addition to the expiration time that applies to the entire server cookie, the server cookie may include a plurality of expiration data, each of which may apply to individual target groups. As will be appreciated by those of ordinary skill in the art, each server cookie can include additional data. For instance, a server cookie may be configured to store client state information, such as a time stamp of the last execution of the authorization sub-routine 702.

Once generated, the authorization server cookie 809 is communicated from the authorization server 122 to the client computing device 110. Next, as shown in block 811, the server cookie is then stored in the memory of the client computing device 110. When the client computing device 110 determines that at least one component of the server cookie has expired, the client computing device can re-execute the authorization method 702 to obtain a new server cookie. As mentioned above, in each subsequent execution of the authorization method 702, the client computing device 110 may communicate its stored server cookies to the authorization server 122 in the cookie request 806. In one embodiment, the client does not have to send the request 803 unless the server informs the client that the server configuration has changed, i.e., a new authorization plug-in has been added.

In accordance with another aspect of the present invention, the software update service 120 may provide a synchronization sub-routine for synchronizing update information between the metadata server 123 and the client computing device 110. By the use of a unique software update hierarchy, the synchronization sub-routine can efficiently identify specific updates that apply to a particular client computing device. In addition, by the use of the server cookie generated in the authorization sub-routine 702,

107

the synchronization sub-routine can selectively grant access to updates associated with specific target groups.

In accordance with an illustrative embodiment of the present invention, each software update includes three components: (1) an instruction component; (2) a localized data component; and (3) a data component. As will be appreciated by one of ordinary skill in the art, each update may have one or more of the above-described components. For example, an update may contain an instruction component, a localized data component, and a data stream component. In another example, an update may only contain an instruction component for testing one or more conditions of a client computing device. The various components of the software updates are described in more detail below.

Generally described, the instruction component contains two sub-components: (1) an applicability rule that defines one or more conditions to be tested by a client computing device 110; and (2) a set of prerequisites that identifies one or more updates that are required for proper installation of an individual update. As described below, the applicability rule can define a number of conditions related a computer, and each of the conditions can be associated with other conditions by the use of any logical operators. For example, the instruction component may include an applicability rule to determine if a computer has a particular version of Windows® installed. As also described below, the set of prerequisites can identify one or more updates that are required to have been previously installed. For example, as described in more detail below with reference to FIGURE 9, an individual update may contain a prerequisite that lists other updates required for the proper installation of the individual update. In other examples, as is also shown in FIGURE 9, the set of prerequisites can include the use of logical operators to define more complex prerequisites rules.

The instruction component also contains code, such as a Boolean flag, that indicates if there are other updates that depend from a particular update. For illustrative purposes, an update is considered to be a LEAF update if there are no other updates that depend from a particular update. The Boolean flag that is used to indicate if an update is a LEAF is dynamically updated by the metadata server 123 as related updates are added or removed.

The localized data component of each update includes general information describing the update. For instance, the localized data component may include

information describing the features and benefits of the update. The localized data component may also include a text description of the installation procedures of the update. In addition, the localized data component may include any other data or information related to the update. For instance, the localized data may indicate that an update is a high priority update. In another example, the localized data may provide special installation messages, such as a message that indicates that an update cannot be installed with other software updates. The localized information may be in a format that allows for the display of its contained information to a user.

The data component of each update includes one or more binary data streams of the update. In one embodiment, the data component of each update may be associated with one or more data files, such as an executable file, a document, a linked library, etc. As described in more detail below, each update may be associated with a combination of data files, each of which facilitates the actual upgrade, installation, or modification of software used by the client. For instance, the installation of an update may be facilitated by the use of a single CAB file including all the information required to complete a selected update. Alternatively, the installation of the update may be facilitated by use of a number of individual updates used to update one or more files stored on a client computing device.

In an illustrative embodiment of the present invention, software updates can be arranged in a hierarchy that allows for controlled distribution of the software updates. Generally described, the hierarchy of updates defines relationships between the updates and specifically indicates which updates are dependent on other updates. For illustrative purposes, an example set of updates is provided and shown in FIGURE 9. As shown, a hierarchy of sample updates 900 includes a base set of updates 901, a second set of updates 902, and a third set of updates 903. In general, each update in the base set of updates 901 does not have a prerequisite that requires the installation of other updates. However, the sixth update 921 contains a prerequisite that requires the installation of the first update 911, the second update 912, and the third update 913. The seventh update 922 contains a prerequisite that requires the installation of the fourth update 914. The eighth update 931 contains a prerequisite that requires the installation of the sixth update 921 and the fifth update 915. Hence, the eighth update 931 also requires the installation of the first update 911, the second update 912, and the third update 913. For purposes of illustrating the present invention, it is given that all of the updates of the

110

sample set of updates 900 are all associated with the All Computers target group and the PID target group.

Also shown in FIGURE 9, and as described in more detail below, each update contains an applicability rule that specifies conditions for the installation of the update.

5 For example, the first update 911 requires the installation of an English version of an operating system. The second update 912 requires the installation of Windows® XP version SP1. In another example, the sixth update 921 requires the installation of a software patch referred to as the XP PATCH1. Accordingly, the client computing device 110 will not install the updates if the applicability rules have not been satisfied.

10 FIGURE 9 also shows the section of the instruction component that indicates if there are other updates that depend from a particular update. This section is referred to as a LEAF, which may be in the form of a Boolean value indicating that a software update is the last update of a series of related updates. For purposes of illustrating the invention, a particular update is a LEAF update if no other updates list that particular update as a prerequisite. As shown in FIGURE 9, the seventh update 922 and the eighth update 931
15 are the only two LEAF updates.

FIGURE 10 illustrates a protocol diagram of a synchronization sub-routine 704 (FIGURE 7) formed in accordance with the present invention. Generally described, the synchronization sub-routine 704 selectively communicates the instruction components of
20 certain updates between the client computing device 110 and a server, such as the metadata server 123, to identify updates that can be applied to the client computing device 110. As shown in FIGURE 10, to initiate an update, the client computing device 110 first processes installed updates and communicates a synchronization request 1051 to the metadata server 123 requesting one or more updates available to the
25 client. In response to receiving the synchronization request 1051, the metadata server 123 returns a number of updates to the client computing device 110. As will be more readily understood from the following description, the client computing device 110 processes locally-stored data prior to the communication of the synchronization request 1051.

30 The client computing device 110 processes the instruction components of each received update to determine if the condition defined in the applicability rules can be met. If the condition defined in an individual update is met, for purposes of illustrating the synchronization sub-routine 1050, the individual update is "installed," and the installed

110

441

update is saved in a first component of the client's update cache. On the other hand, if the condition defined in an individual update is not met, individual update is considered to be "failed," and the failed update is saved in a second component of the client's update cache. In this description of the synchronization sub-routine 1050, if an update is
5 installed it can be assumed that the prerequisites and the conditions of the applicability rules have been met. Installation of an update for purposes of describing this sub-routine does not necessarily mean that the data files associated with the update are actually installed in the client computing device 110.

In one embodiment, the two components of the client's update cache can be used
10 to categorize the received updates. The first component is used for storing installed, non-LEAF updates; and a second component is used for storing all other updates received by the client, i.e., the updates that were not installed. The second component of the update cache also includes the storage of all LEAF updates. As described in more detail below, updates stored in the update cache can be communicated to and processed by the
15 metadata server 123 to identify other related updates that are available for installation on the client computing device 110.

Returning now to FIGURE 10, details of the synchronization request, which are illustrated as items 1051, 1055, and 1060, will now be described. As will be appreciated by those of ordinary skill in the art, the synchronization request may be initiated by one
20 of a number of different devices, processes, applications, user-initiated commands, requesting an update. The synchronization request may be initiated by a user requesting a list of updates, an automatic update initiated by the client agent, or any other software component requesting information from the metadata server 123 or the update service 120. In one embodiment, the synchronization request includes an authorization
25 server cookie, such as the authorization server cookie generated from the authorization routine 702. The use of the server cookie allows the server to determine if the client is a member of one or more target groups.

Each synchronization request may also include identifiers for each update stored in the client's update cache. More specifically, if one or more updates are stored in the
30 update cache, the synchronization request includes a first component having identifiers for installed, non-LEAF updates; and a second component having identifiers for all other updates, such as LEAF updates, failed updates, and other updates that are not installed. The update identifiers may be in any format including, but not limited to, an array of

+++

112

integers. Alternatively, if there are no updates stored in the client's update cache, the synchronization request is not configured with an update identifier. When a synchronization request is not configured with an update identifier, the synchronization request provides an indication that the client computing device 110 does not have any
5 cached updates.

As shown in FIGURE 10, a first synchronization request 1051 is communicated from the client computing device 110 to the metadata server 123. In the present example, the client's update cache will not contain any updates, given that this is the first execution of the method. Thus, the first synchronization request 1051 does not contain an identifier
10 for a cached update. In response to receiving a synchronization request, as shown in block 1052, the metadata server 123 determines if the synchronization request contains at least one update identifier. If it is determined that the synchronization request does not include an update identifier, the metadata server 123 responds by selecting first level updates for communication to the client computing device 110. As described above, first
15 level updates may include any updates that do not have a prerequisite identifying other updates.

Alternatively, if it is determined that a synchronization request contains at least one update identifier, the metadata server 123 examines the prerequisites of the server's stored updates to select additional updates for delivery to the client. In one embodiment,
20 the metadata server 123 selects updates having fulfilled prerequisites. In the examination of the prerequisites, the server uses the updates of the first component of the synchronization request, which includes identifiers of the non-LEAF updates that are installed on the client.

In addition to selecting updates having fulfilled prerequisites, the server also uses
25 the updates identified in the second component of the synchronization request to filter the selected updates. More specifically, uninstalled updates, LEAF updates and failed updates identified in the second component of the synchronization request are used to filter one or more selected updates. This feature of the present invention allows the system and method of the present invention to avoid multiple transmissions of updates
30 stored on the metadata server 123.

Returning to the present example, since the first synchronization request 1051 does not include an update identifier, the metadata server 123 selects the base level of updates 901 for communication to the client computing device 110. With reference to the

H2

113

sample set of updates shown in FIGURE 9, the base level of updates 901 includes the updates referenced as 911, 912, 913, 914, and 915.

5 In the processing of block 1052, the metadata server 123 also examines the authorization server cookie contained in the synchronization request 1051 to identify the target groups that are associated with the client computing device 110. The metadata server 123 also examines the target groups of the updates selected in the process of block 1052. The process of block 1052 then filters out all selected updates that are not associated with a target group identified in the received authorization server cookie. In the present example, since all of the selected updates 911, 912, 913, 914, and 915 are
10 associated with the PID and All Computer target groups, all of the selected updates are sent to the client computing device 110.

The metadata server 123 then communicates the selected updates in a synchronization response 1053 to the client computing device 110. In general, each synchronization response includes the instruction component of each update sent by the
15 server 120. Thus, in the present example, the first synchronization response 1053 includes the instruction components for the updates referenced as 911, 912, 913, 914, and 915. In one embodiment, each synchronization response does not include the localized data component or the data component of each update.

Next, as shown in block 1054, the client computing device 110 processes the
20 instruction components of each received update to determine if the condition defined in the applicability rules can be met. With reference again to FIGURE 9, the client computing device 110 processes the instruction components of the received updates 911-915. For purposes of illustrating the present invention, it is given in this example that the operating system of the client computing device 110 is an English
25 installation of Windows® version XP SP1. It is also given that the client computing device 110 is a Dell PC and is running a 32-bit, X86 processor. Thus, in the processing of the instruction components of the sample set of updates, the client computing device 110 would determine that the condition defined in the first update 911 would be met because the computer contains an English OS. The condition defined in the second
30 update 912 would be met because the operating system is Windows® version XP SP1. The condition defined in the third update 913 would be met because the client computing device 110 is running an X86 processor. The condition defined in the fifth update 915 would be met because the client computing device 110 is a Dell PC. As a result, the first

113

114

update 911, second update 912, third update 913, and fifth update 915 are all saved in the first component of the client's update cache. The condition defined in the fourth update 914 would not be met because the client computing device 110 is not running a 64-bit X86 processor. Thus, the fourth update 914 is considered to be a failed update and saved in the second component of the client's update cache.

Returning to FIGURE 10, in the processing of block 1054, the client computing device 110 also determines if a subsequent synchronization request is required. In one embodiment, it is determined that a subsequent synchronization request is required if at least one of the received updates indicates that it is not a LEAF update. In the present example, it is determined that a subsequent synchronization request is required because all of the received updates are not LEAF updates. Thus, the client computing device 110 communicates a subsequent synchronization request 1055 to the metadata server 123.

As summarized above, a synchronization request includes identifiers for each update stored in the client's update cache. Thus, in the present example, the subsequent synchronization request 855 includes a first data component indicating that the first update 711, second update 712, third update 713, and fifth update 715 are installed on the client. In addition, the subsequent synchronization request 855 includes a second data component indicating that the fourth update 711 is not successfully installed on the client.

In response to receiving the subsequent synchronization request 1055, as summarized above, the metadata server 123 determines if the subsequent synchronization request 1055 contains at least one update identifier. If it is determined that subsequent synchronization request contains at least one update identifier, the metadata server 123 examines the prerequisites of all stored updates to select additional updates for delivery to the client.

With reference again to the present example, in the processing of block 856, the metadata server 123 would select the sixth update 921 because its prerequisites are met. More specifically, as shown in FIGURE 9, the sixth update 921 is selected for communication to the client computing device 110 because its prerequisite, which requires installation of the first update 711, second update 912, and third update 913, is fulfilled. The seventh update 922 and eighth update 931 would not be selected for communication to the client because their prerequisites are not fulfilled. More specifically, the synchronization request 1055 did not contain an identifier for the fourth update 914, which is a prerequisite for the seventh update 922. In addition, the

114

115

synchronization request 855 did not contain an identifier for the sixth update 921, which is a prerequisite for the eighth update 931.

Returning to FIGURE 10, the synchronization sub-routine 1050 continues by communicating the selected updates in a subsequent response 1057 from the metadata server 123 to the client computing device 110. With reference again to the present example, the subsequent response 1057 would include information related to the sixth update 721, is communicated to the client 110 in a subsequent response 1057.

Upon receiving the subsequent response 1057, the client computing device 110 processes the instruction components of the subsequent response 857. Similar to the process of block 854, the client computing device 110 processes the instruction components of each received update to determine if the condition defined in the applicability rules is met. In the present example, if it is given that the XP PATCH1 is installed in the client computing device, the sixth update 921 is considered to be installed and the update is written to the update cache of the client computing device 110. Since the sixth update 921 is not a LEAF update, the client computing device 110 sends another synchronization request 1060 that includes all of the updates stored in the first and second component of the client's update cache. The synchronization request 1060 also includes the authorization server cookie.

In the present example, by use of the above-described processing of the metadata server 123, the synchronization request 1060 is processed at block 1061 where the server selects the eighth 731 update. The eighth 931 update is selected because the synchronization request 1060 indicates that the fifth and the sixth updates 915 and 921 are installed in the client computing device 110. Given that the eighth update 931 is associated with the same target groups identified in the authorization server cookie, the instruction component of the eighth update 931 is communicated to the client computing device 110 in another response 1062. The eighth update 931 is then processed in block 1063 in a manner similar to the process of blocks 1054 and 1059. Since all of the received updates of the response 1062 are LEAF updates, a subsequent synchronization request is not sent back to the metadata server 123.

At the client computing device 110, after it is determined that all of the received updates are LEAF updates, or if no updates are received in the response 1062, the synchronization sub-routine 1050 communicates a driver synchronization request 1064 from the client computing device 110 to the metadata server 123. As will be appreciated

TTS

by one of ordinary skill in the art, the driver synchronization request 1064 may include information describing all of the hardware installed in the client computing device 110 and information describing the installed software. Similar to the prior software synchronization requests (1051, 1055 and 1060), the driver synchronization request 1064
5 may communicate the installed updates to the server. In addition, all of the driver updates currently cached on the client, if any, are communicated to the server.

In response to receiving the driver synchronization request 1064, the metadata server 123 responds by sending all of the driver updates that apply to the client computing device 110 that are not already cached on the client. A driver update is sent to
10 the client computing device 110 in a response 1065 if its prerequisites and conditions are met. The response 1065 communicating the driver updates preferably communicates the instruction component of each update. The driver updates are then written to the update cache of the client computing device.

After the receiving the response 1065 containing the driver updates, the synchronization sub-routine 1050 sends a request 1066 for the localized data of each of
15 the received software and hardware updates. As summarized above, the localized data component of each update includes general information describing the update. For instance, the localized data component may include information describing the features and benefits of the update. The localized data component may also include a text
20 description of the installation procedures of the update. In addition, the localized data component may include any other data or information related to the update.

Thus, upon receiving request 1066 for the localized data of each of the received software updates and hardware updates, the metadata server 123 responds by sending all of the localized data for all of the received software updates and hardware updates saved
25 in the update cache of the client. Once received, the localized data can be processed by a software application to determine which of the updates needs to be installed. Alternatively, the received localized data can be displayed to a user to inform the user of all updates that are available to the client computing device 110. In one embodiment, the received localized data can be displayed on a Web page. In the present example,
30 localized data may be received by the client for the sixth and eighth updates 921 and 931. If localized data is stored in the base updates 911, 912, 913 and 915, the localized data for those update would also be received by the client.

117

FIGURE 11 illustrates one example of a Web page 1100 displaying an example of localized data associated with the updates that are available to the client. For illustrative purposes, the Web page 1100 comprises a first detailed description 1105 of an update and a second detailed description 1106 of another update. Also shown, each update is
5 respectively associated with selection mechanisms 1103 and 1104 for receiving a user selection of the updates. Also shown, the Web page 1100 is configured with a control button 1101 for allowing a user to control the communication of the selection of updates to a server, such as the metadata server 123 or download server 124.

In one aspect of the present invention, the client performs a number of processes
10 to enhance the display of the Web page 1100. For instance, the client computing device 110 examines the localized data of each update to determine if a particular update is of high priority. Such a feature can be facilitated by locating text in the localized data, or in other component of a particular update, that indicates that the particular update is a high priority or emergency update. If the client computing device 110 detects a high
15 priority update or emergency update, the client displays the high priority update in a visible section of the Web page 1100, such as the top section of the page. In addition, the client may generate a visual indicator, such as a specialized text message 1120, indicating that the update is a high priority update.

The client computing device 110 may also examine the localized data of each
20 update to determine if a particular update requires an exclusive installation, i.e., an update having an installation file that cannot be simultaneously installed with an installation file of another update. Such a feature can be facilitated by locating text in the localized data, or in other component of a particular update, that indicates that the particular update requires an exclusive installation. If the client computing device 110 detects such an
25 update, the client displays a visual indicator, such as the text message 1122 shown in FIGURE 11, with the description of the updates that require an exclusive installation.

Returning to FIGURE 7, the software update routine 700 continues at block 708 where the client computing device 110 receives a selection of the updates. As noted above, in response to actuation of the control button 1101, the selection of the one or
30 more updates can be obtained by the metadata server 123 or the download server 124. Once the selection of one or more updates is received the software update routine 700 continues at block 708 where the selected software updates are processed.

117

118

In accordance with still another aspect of the present invention, the software update service 120 may provide a method for selecting and transmitting information between the software update service and the client computing device 110. FIGURES 12A and 12B are illustrative of a software update processing sub-routine 1200 implemented by the client computing device 110 to retrieve and install requested software in accordance with the present invention. As described above, the software update processing sub-routine 1200 may be implemented once a selection of software updates has been generated or received. With reference to FIGURE 12A, at block 1202, the update management component 111 instantiates an update agent 118. In an illustrative embodiment of the present invention, the update agent 118 is a specialized software component for determining what software update information is required to complete a requested software update, to generate a required version of an installation component of the update agent, to generate updated files by merging existing files with delta patches, and/or to initiate the installation of updated files. In the event that an update agent 118 is already instantiated, block 1202 may be omitted.

At block 1204, the update agent 118 obtains software update information from the update service 120. In an illustrative embodiment of the present invention, the software update information transmitted by the update service 120 is in the form of a package, such as a self-extracting file, that includes a variety of data that may be utilized by the update agent. In one aspect, the package can include a list of all the files that correspond to a particular software update. Additionally, the package can include copy of at least a portion of the patch storage manifest that maps specific versions of files to be updated to a corresponding software update delta patch stored in the patch storage file on the update service 120. The package can also include installation information for each file to be updated that can include an identification of a version of an installation component required to complete the installation. Further, the package can also include an installation component for the update agent 118 or a delta patch to update a version of an installation component already stored on the client computing device 110. Still further, the package can include verification information to allow the update agent to determine whether a software update was successful. For example, the verification information can include reference hash values for updated files for comparison. The update agent 118 may also verify the contents of package.

118

At decision block 1206, a test is conducted to determine whether the update agent 118 needs to update a version of the installation component to implement the update. One skilled in the relevant art will appreciate that the transmission of a complete copy of an installation component in the self-extracting file can increase the amount of data transmitted by the update service 120 for each software update. Accordingly, in an illustrative embodiment of the present invention, a baseline version of an installation component may be stored in the client computing device and updated specifically for the requirements of the current software update by way of an installation component delta patch. Accordingly, the installation information in the self-extracting file instructs the update agent 118 whether or not any included installation component updates need to be merged with the baseline version of the installation component on the client computing device 110. If an update is required, at block 1208, the update agent 118 updates the baseline installation component, as will be explained in greater detail below with regard to FIGURE 13.

Once the update agent updates the installation component or if the installation component does not require an update, at block 1210, the update agent 118 performs an inventory of files installed on the client computing device 110 and the specific version of the file. In an illustrative embodiment of the present invention, the update agent 118 may query the client computing device 110 file system for all files identified in the package as corresponding to the selected update. Alternatively, if the update agent 118 has recently conducted an inventory, a cached version of the inventory may be utilized. At block 1212, the update agent 118 identifies what software update information is required to complete the requested update. In an illustrative embodiment of the present invention, the patch storage manifest includes a mapping of versions of an installed file to a required delta patch. Accordingly, if delta patching is available, the update agent 118 will utilize the mapping to identify a particular delta patch and its offset location within the patch storage file. Alternatively, if delta patch is not available or cannot be implemented, the update agent 118 may identify an entire file for download.

With reference now to FIGURE 12B, at block 1214, the update agent transmits a request for the identified software update information. In an illustrative embodiment of the present invention, the update agent 118 can transmit a request for specific delta patches by indicating a specific range of patches required from the patch storage file to the download server 124 of the update service 120. As described above,

the patch storage file includes a large number of applicable delta patches, in which each delta patch is identified by its location with the patch storage file. Because the patch storage file may be rather large in some implementations, the update agent 118 can utilize a request that only requests for data from specific locations in the patch storage file as indicated from the patch storage manifest. In an alternate embodiment of the present invention, the update agent 118 may request an entire copy of an update file and/or a complete copy of the patch storage file.

In an alternate embodiment of the present invention, another download server that may not be exclusively associated with the update service 120 can process the update agent 118 request. In this embodiment, the request patch storage file may be transmitted, in whole or in part, to any number of additional download servers on a network. The additional download servers may be part of a private network utilized to update clients on the private network. Further, the additional download server may be part of public network. In a private network environment, the download servers may obtain a complete copy of the patch storage file for processing client requests. Alternatively, the download servers may also cache portions of the patch storage file in processing previous data requests from other clients and utilize the cache data to satisfy the download. Accordingly, the additional download servers can reduce the communications strain on the download server 124 of the update service 120.

At block 1216, the update agent 118 receives the requested update information. In an illustrative embodiment of the present invention, the requested update information may be transmitted in two approaches. In a first approach, referred to as manual update, the update request is sent to the update service 120 with a request for a direct HTTP data delivery response. In this approach, the update service 120 may utilize all of the entire bandwidth available to transmit the requested data to the update agent 118. In a second approach, referred to as an automatic update, the update request is sent to the update service 120 with a request for an indirect HTTP data delivery response. In this response, the update service 120 transmits the requested data as a background process. The background process may be implemented in a manner to utilize a minimal amount of available bandwidth. Further, the background process may be interrupted during the download process and restarted at the next available time. A description of a system and method for transmitting requested data via a background process is described in commonly assigned and copending U.S. Patent Application No. 09/505,735, entitled

121

System and Method for Transferring Data Over a Network, and filed on February 16, 2000, which is hereby incorporated by reference. One skilled in the relevant art will appreciate that foreground or background data delivery is not necessarily reflective of a priority of the selected software update, but rather how bandwidth is allocated to obtain the update information.

Once the requested information is received from the update service, at block 1218, the update agent 118 merges the delta patch with the corresponding installed files. In an illustrative embodiment of the present invention, the update agent 118 may cache the original version of the installed file to ensure that the selected file does not change during the download and merging process. Further, the cached original version of the installed file may be used to uninstall the selected update.

At decision block 1220, a test is conducted to determine whether the updated file is valid. In an illustrative embodiment of the present invention, the update agent 118 may utilize a hashing algorithm to compare a reference hash value obtained from the update information package and corresponding to a valid file update with a hash from the current modified file. If the hashes do not match, the current modified file is not valid. One skilled in the relevant art will appreciate that any one of a number of alternative validation algorithms may also be utilized. If the updated file is not valid, the sub-routine 1200 returns to block 1214, where the update agent may request the update information again. Alternatively, if the update agent 118 has unsuccessfully attempted to generate the update file several times, the update agent may implement one of several fallback procedures. In one embodiment of the present invention, the update agent 118 may request a completed copy of the updated file stored in the patch storage file and identified from the patch storage manifest from the update service 120. In another embodiment of the present invention, the update agent 118 may request a copy of the updated file in a self-contained file from the update service 120. In still another embodiment of the present invention, the sub-routine 1200 may otherwise fail.

Once the selected file is valid, at decision block 1222, a test is conducted to determine whether any additional downloads are required. In an illustrative embodiment of the present invention, the sub-routine 1200 enters into an iterative loop that continuously checks for additional downloads after the completion of a previously selected download. If the state of a file changes during the download, the update agent 118 would continue to request additional downloads for the new change of state. If

121

additional downloads are required, at block 1224, the update agent 118 performs another inventory and identifies all applicable delta patches. The sub-routine 1200 then returns to block 1214.

Once all the requested update downloads are complete, at decision block 1226, a
5 test is conducted to determine whether the state of the client machine has changed. In an illustrative embodiment of the present invention, time may elapse between the downloading and merging of update information and the actual installation of the updated file. Accordingly, prior to installing the updated file, the update agent determines whether the client computing device state has changed. If the state has changed, the file
10 update may not be valid and the update fails at block 1228. Alternatively, if no state change has occurred, the update agent 118 installs the updated file at block 1230 and the sub-routine 1200 returns at block 1232.

With reference now to FIGURE 13, a sub-routine 1300 implemented by the client computing device 110 for updating a baseline installation component corresponding to
15 block 1208 (FIGURE 12A) will be described. At decision block 1302, a test is conducted to determine whether a new baseline installation component is included in the self-extracting file transmitted to the update agent 118 from the update service 120. In an illustrative embodiment of the present invention, if the delta patches required to update the baseline installer are comparable in size to the transmission of an updated installation
20 component, a new baseline installation component will be transmitted. If an updated installation component is included, at block 1304, the update agent installs the updated baseline installation component as the new installation component. Additionally, the new updated installation component may be saved in the client computing device 110 memory to serve as a baseline installer for additional updates. At block 1306, the sub-routine
25 returns.

If an updated baseline installation component is not included in the self-extracting file, at block 1308, the update agent 118 obtains a baseline installation component delta patch from the self-extracting file. In an illustrative embodiment of the present invention, the baseline installation component delta patch corresponds to software code that can be
30 merged with the baseline installation component to generate an updated baseline installation component. Accordingly, at block 1310, the updated agent merges the baseline installation component delta patch with the baseline installation component. At block 1312, the update agent 118 then designates the updated baseline installation

122

component as the current installation component. In an illustrative embodiment of the present invention, the updated installation component will not be saved after the installation is complete. In accordance with this embodiment, the update agent 118 only maintains a limited number of baseline installation components in the client computing
5 device 110 memory. Accordingly, the update agent generates a temporary, updated installation component at each installation. Because each client computing device 110 can only correspond to a limited number of baseline installation components, the update service 120 is only required to transmit a single baseline installation component delta patch for each client computing device. At block 1314, the sub-routine 1300 returns.

10 While the preferred embodiment of the invention has been illustrated and described, it will be appreciated that various changes can be made therein without departing from the spirit and scope of the invention. For instance, although the illustrative examples described herein apply to software updates, the scope of the present invention includes other uses beyond the distribution and communication of information
15 related to software updates. Accordingly, unless specific subject matter is expressly excluded in this disclosure, it is to be appreciated that the scope of the present invention applies to distribution and communication of any type of data, other than, or in addition to software updates.

The embodiments of the invention in which an exclusive property or privilege is claimed are defined as follows:

1. In a computer system including a client computing device in communication with a software update service, a method for communicating software updates from the software update service to the client computing device, the method comprising:

(a) obtaining a synchronization request from the client computing device to the software update service, wherein the synchronization request includes an identifier of an installed software update if the client computing device stores the installed software update;

(b) determining if the synchronization request comprises an identifier of at least one installed software update;

(c) if it is determined that the synchronization request comprises an identifier of at least one installed software update, selecting an additional software update for communication to the client computing device, wherein the selection of the additional software update is dependent on the fulfillment of a prerequisite defined in the additional software update, wherein the prerequisite requires the synchronization request to include an identifier for at least one installed software update;

(d) if it is determined that the synchronization request does not comprise an identifier of at least one installed software update, selecting a first level software update for communication to the client computing device, wherein the first level software update does not include a prerequisite;

(e) communicating an instruction component of the selected software update to the client computing device from the software update service; and

(f) storing the instruction components of the selected software update in the client computing device as an installed software update if the client computing device contains at least one component that fulfills a condition of an applicability rule stored in the selected software update.

2. The method as recited in Claim 1, wherein the method further comprises:

determining if at least one instruction component of the selected software update provides an indication that the software update service contains additional software updates available to the client computing device; and

repeating elements (a) – (f) if it is determined that at least one instruction component of the selected software update provides an indication that the software update service contains additional software updates available to the client computing device.

3. The method as recited in Claim 1, wherein the method further comprises:
if the instruction component does not provide an indication that the software update service contains additional software updates available to the client computing device:

communicating a localized data component of the selected software update from the software update service to the client computing device; and

generating a display containing text derived from the communicated localized data component, wherein the text provides a description of the selected software update.

4. The method as recited in Claim 3, wherein the method further comprises:
obtaining a command identifying a selected data file associated with the selected software update; and

in response to receiving the command, communicating the data file associated with the selected software update from the software update service to the client computing device.

5. The method as recited in Claim 1, wherein the method further comprises:
determining if the client computing device stores the installed software update;
providing an identifier of the installed software update in the synchronization request if it is determined that the client computing device stores the installed software update.

6. The method as recited in Claim 1, wherein the prerequisite defines a condition which requires a combination of software updates referenced in the synchronization request, wherein the condition is defined by at least one logical operator linking the combination of software updates.

7. The method as recited in Claim 1, wherein the method further comprises:
storing the instruction components of the selected software update in the client computing device as a failed software update if the client computing device does not contain at least one component that fulfills at least one condition of an applicability rule stored in the selected software update.

8. The method as recited in Claim 7, wherein the method further comprises:

T23

126

obtaining an identifier of the failed software update in the synchronization request if the client computing device stores the failed software update;

if it is determined that the synchronization request comprises an identifier of the failed software update, restricting the communication of selected software updates related to the failed software update.

9. The method as recited in Claim 1, wherein the synchronization request generated from an automatic update program module executing on the client computing device.

10. The method as recited in Claim 1, wherein the synchronization request is invoked by a user requesting a manual update.

11. The method as recited in Claim 1, wherein the selected software update is configured to update one or more files installed on the client computing device, wherein communicating the selected software update from the software update service to the client computing device comprises:

obtaining a self-extracting file identifying a plurality of available delta patches for updating at least one version of an installed file;

obtaining an inventory of the one or more files installed on the client computing device;

selecting one or more applicable delta patches to implement the selected software updates, wherein selecting the one or more applicable patches corresponds to a mapping of the self-extracting file identifying the plurality of available delta patches to the inventory of the one or more files installed on the client computing device; and

transmitting a request for one or more selected delta patches.

12. A computer-readable medium having computer executable instructions for performing the method recited in Claim 1.

13. A computer system having a processor, a memory and an operating environment, the computer system operable to perform the method recited in Claim 1.

14. In a computer system including a client computing device in communication with a software update service, a method for communicating software

126

127

updates from the software update service to the client computing device, the method comprising:

obtaining an authorization request from the client computing device, wherein the authorization request contains an identifier of a client authorization module stored on the client computing device;

determining if the client computing device is associated with a target group, wherein it is determined that the client computing device is associated with the target group if the client authorization module indicates that the client computing device is associated with the target group;

communicating a server cookie from the software update service to the client computing device if it is determined that the client computing device is associated with the target group, wherein the server cookie identifies the target group;

obtaining a request for at least one software update stored on the software update service, wherein the request contains the server cookie;

in response to obtaining the request, determining if a software update is associated with the target group identified in the server cookie; and

if it is determined that the software update is associated with the target group identified in the server cookie, communicating the software update from the software update service to the client computing device.

15. The method as recited in Claim 14, wherein the authorization request includes data of a previously stored server cookie, and wherein the server cookie communicated from the software update service to the client computing device includes the data of the previously stored server cookie.

16. The method as recited in Claim 14, wherein it is determined that the client computing device is associated with a target group if a server authorization module identifies the same target group identified in the client authorization module.

17. The method as recited in Claim 14, wherein the server cookie comprises an expiration time, wherein the method further comprises:

determining if the expiration time of the server cookie has passed;

if it is determined that the expiration time of the server cookie has passed, restricting the communication of the software update from the software update service to the client computing device.

127

18. The method as recited in Claim 14, wherein the server cookie stores the expiration time in a clear text format and in an encrypted format, and wherein the server cookie contains encrypted data identifying at least one target group, wherein the method further comprises:

obtaining the expiration time from the server cookie, wherein the client computing device obtains the expiration time from the expiration time stored in the clear text format;

determining if the expiration time has not passed; and

if it is determined that the expiration time has not passed, communicating the server cookie from the client computing device to the software update service in the request for at least one software update.

19. The method as recited in Claim 14, wherein the authorization module is stored on the client computing device during an installation of a software application.

20. The method as recited in Claim 14, wherein the authorization module is stored on the client computing device during an update process of a software application.

21. The method as recited in Claim 14, wherein communicating the software update from the software update service to the client computing device includes communicating an instruction component of the software update, and wherein the method further comprises:

processing the instruction component of the software update to determine if the instruction component provides an indication that the software update service contains additional software updates available to the client computing device;

communicating a localized data component of the software update from the software update service to the client computing device if the instruction component does not provide an indication that the software update service contains additional software updates available to the client computing device; and

processing the localized data component for the selection of at least one update file for updating files installed on the client computing device.

22. The method as recited in Claim 21, wherein processing the localized data component includes:

708

displaying data of the localized data component on a display of the client computing device, wherein the display includes a component for generating a request for a selected update; and

obtaining at least one request indicating the selection of at least one update file for updating files installed on the client computing device.

23. The method as recited in Claim 14, wherein the server authorization module is a software plug-in configured to be dynamically added or removed from the software update service.

24. The method as recited in Claim 14, wherein the request for at least one software update is generated from an automatic update program module executing on the client computing device.

25. The method as recited in Claim 14, wherein the request for at least one software update is invoked by a user requesting a manual update.

26. The method as recited in Claim 14, wherein the software update is configured to update one or more files installed on the client computing device, wherein communicating the software update from the software update service to the client computing device comprises:

obtaining a selection of one or more available software updates for updating one or more files installed on the client computing device;

obtaining a self-extracting file identifying a plurality of available delta patches for updating at least one version of an installed file;

obtaining an inventory of the one or more files installed on the client computing device;

selecting one or more applicable delta patches to implement the selected software updates, wherein selecting the one or more applicable patches corresponds to a mapping of the self-extracting file identifying the plurality of available delta patches to the inventory of the one or more files installed on the client computing device; and

transmitting a request for one or more selected delta patches.

27. A computer-readable medium having computer executable instructions for performing the method recited in Claim 14.

28. A computer system having a processor, a memory and an operating environment, the computer system operable to perform the method recited in Claim 14.

29. In a computer system including a client computing device in communication with a server, a method for communicating a data structure from the server to the client computing device, the method comprising:

determining if the client computing device is associated with a target group, wherein it is determined that the client computing device is associated with the target group if the client computing device contains an authorization module indicating that the client computing device is associated with the target group;

communicating a server cookie from the server to the client computing device if it is determined that the client computing device is associated with the target group, wherein the server cookie identifies the target group;

obtaining a request for at least one data structure stored on the server, wherein the request contains the server cookie;

in response to obtaining the request, determining if a data structure is associated with the target group identified in the server cookie;

if it is determined that the data structure is associated with the target group identified in the server cookie, communicating the data structure from the server to the client computing device.

30. The method as recited in Claim 29, wherein the method further comprises:

obtaining an authorization request from the client computing device, wherein the authorization request contains an identifier of the authorization module stored on the client computing device;

in response to obtaining the authorization request, determining if the client computing device is associated with a target group, wherein it is determined that the client computing device is associated with the target group if the authorization request identifies at least one authorization module stored on the client computing that is associated with the target group.

31. The method as recited in Claim 29, wherein the server cookie comprises an expiration time, wherein the method further comprises:

determining if the expiration time of the server cookie has passed;

if it is determined that the expiration time of the server cookie has passed, restricting the communication of the data structure from the server to the client computing device.

32. The method as recited in Claim 29, wherein the server cookie stores the expiration time in a clear text format and in an encrypted format, and wherein the server cookie contains encrypted data identifying at least one target group, wherein the method further comprises:

obtaining the expiration time from the server cookie, wherein the client computing device obtains the expiration time from the expiration time stored in the clear text format;

determining if the expiration time has not passed; and

if it is determined that the expiration time has not passed, communicating the server cookie from the client computing device to the software update service in the request for at least one data structure.

33. A computer-readable medium having computer executable instructions for performing the method recited in Claim 29.

34. A computer system having a processor, a memory and an operating environment, the computer system operable to perform the method recited in Claim 29.

ABSTRACT OF THE DISCLOSURE

A system and method for facilitating the selection and implementation of software updates while minimizing the bandwidth and processing resources required to select and implement the software updates. In one embodiment, an update service controls access to
5 software updates, or other types of software, stored on a server.

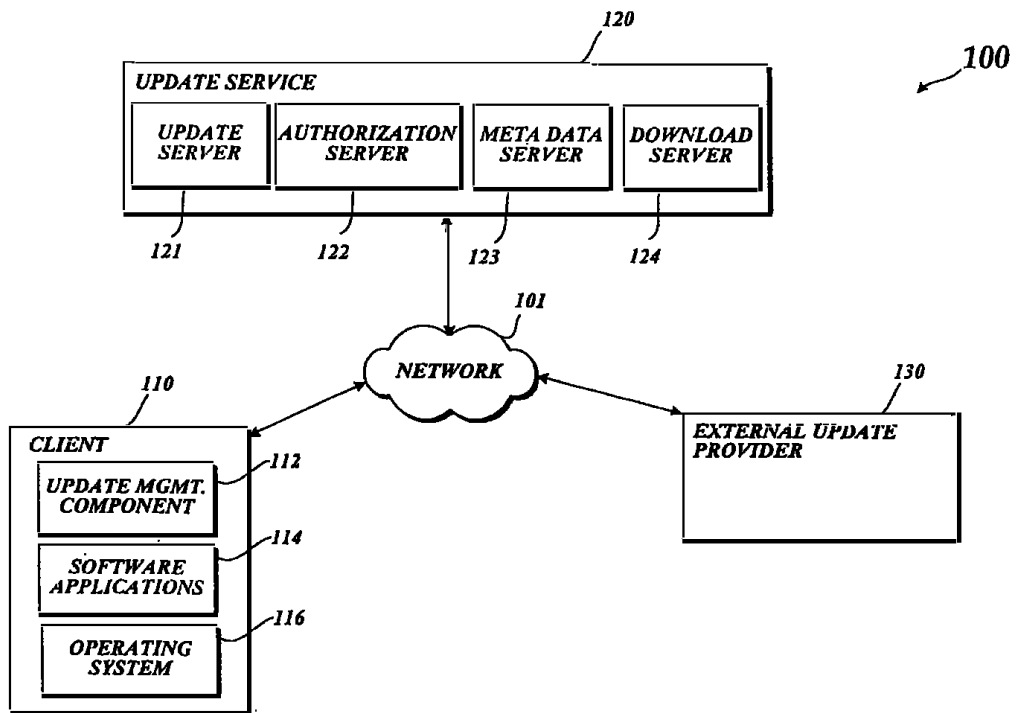


Fig. 1

45

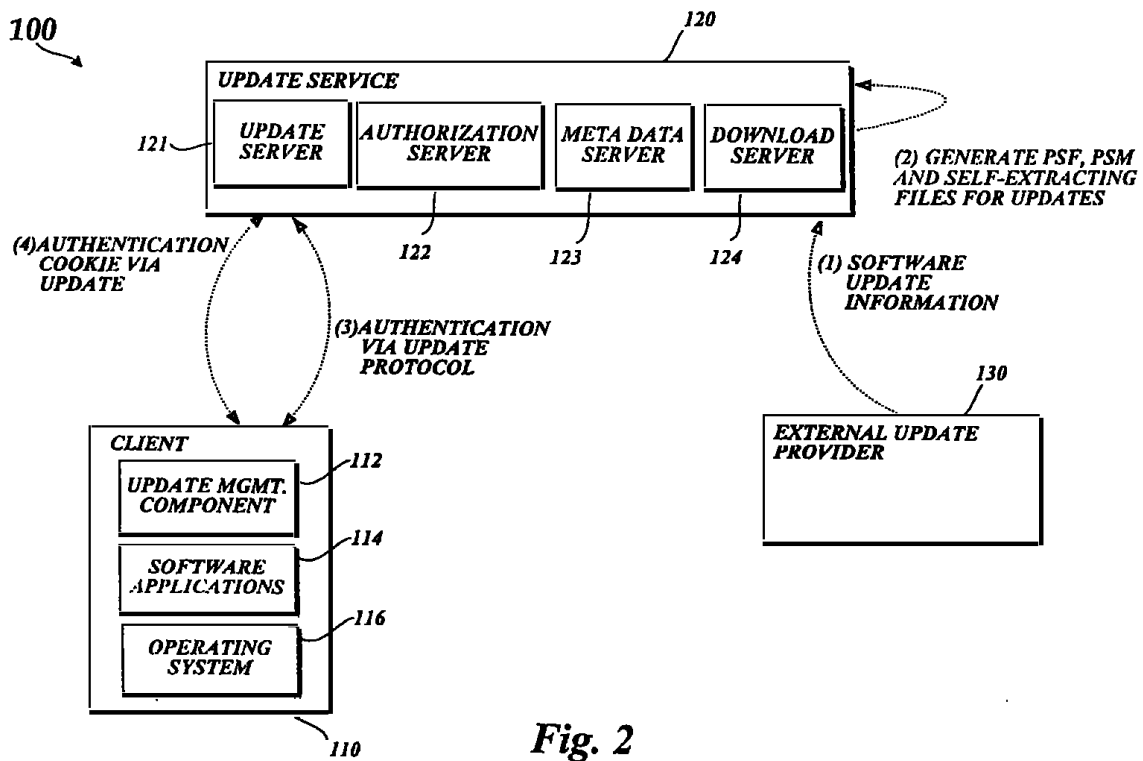
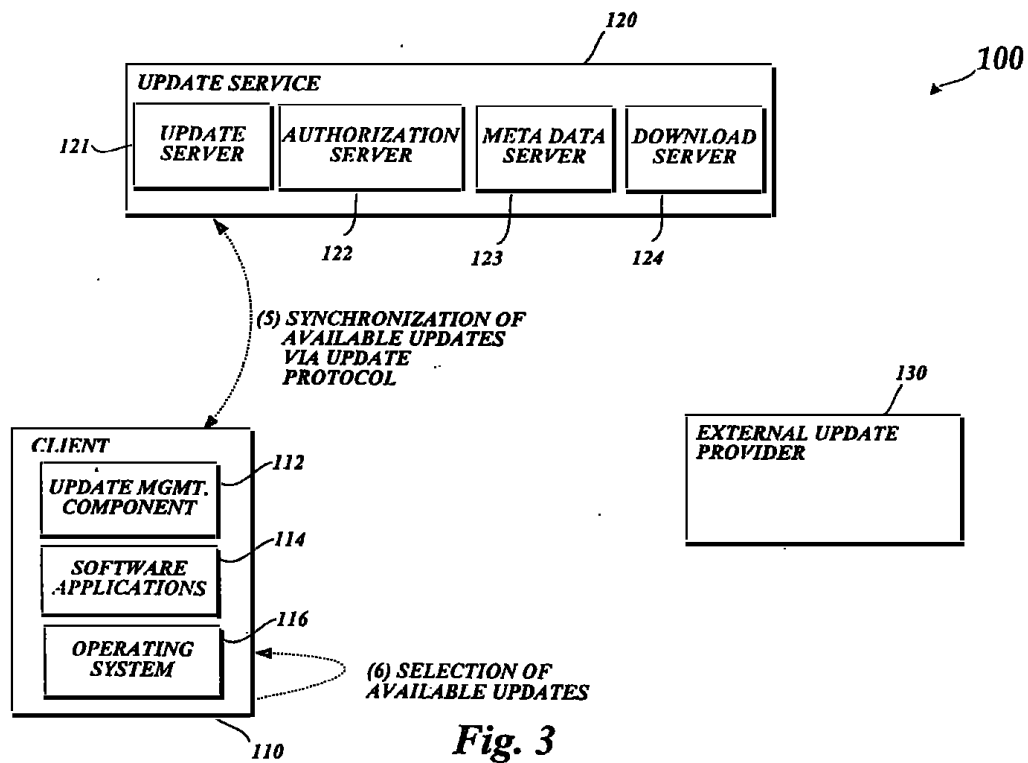
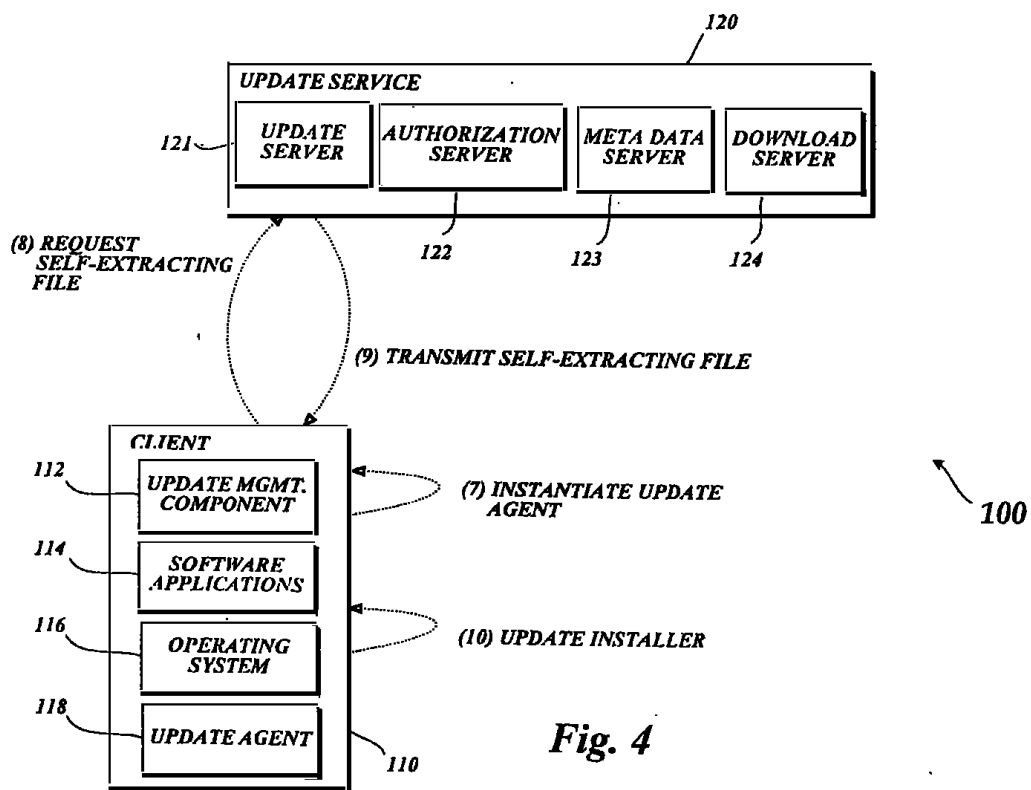


Fig. 2

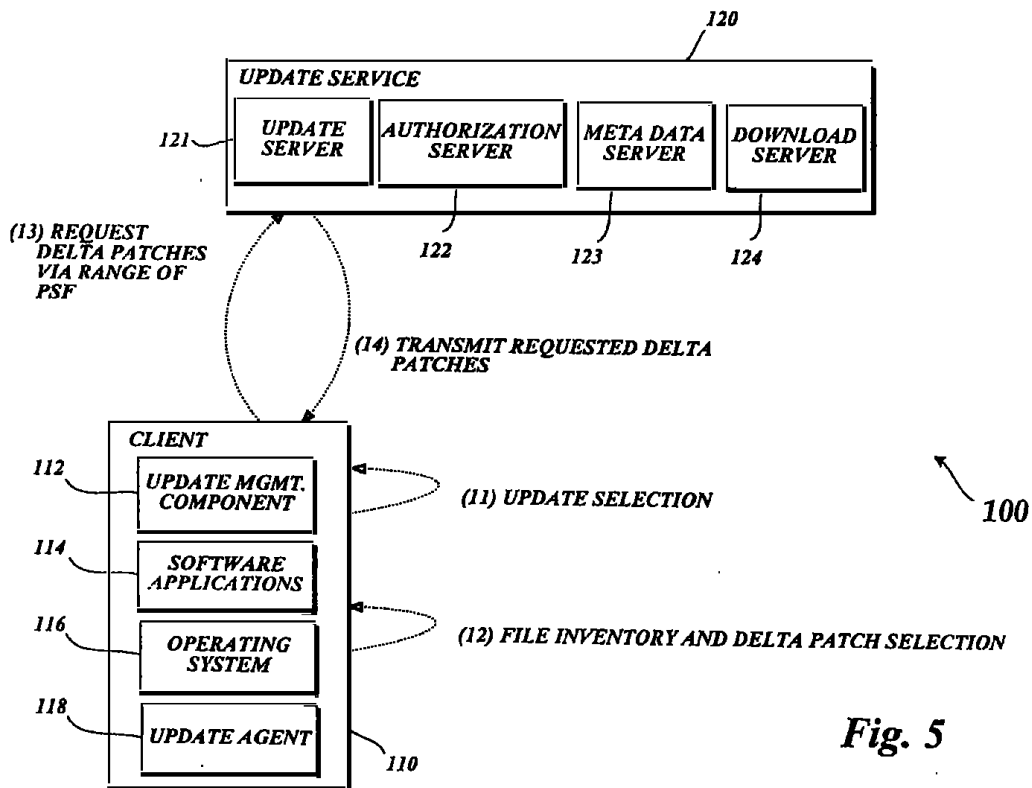
2/14





4/14

38



137

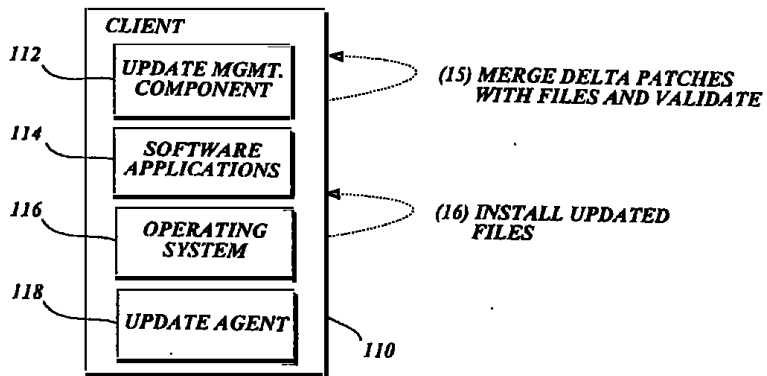
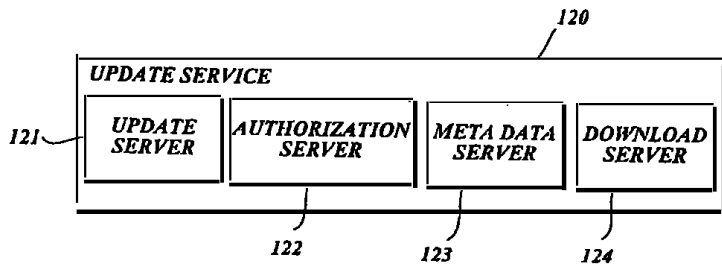


Fig. 6

6/14

46

7/14

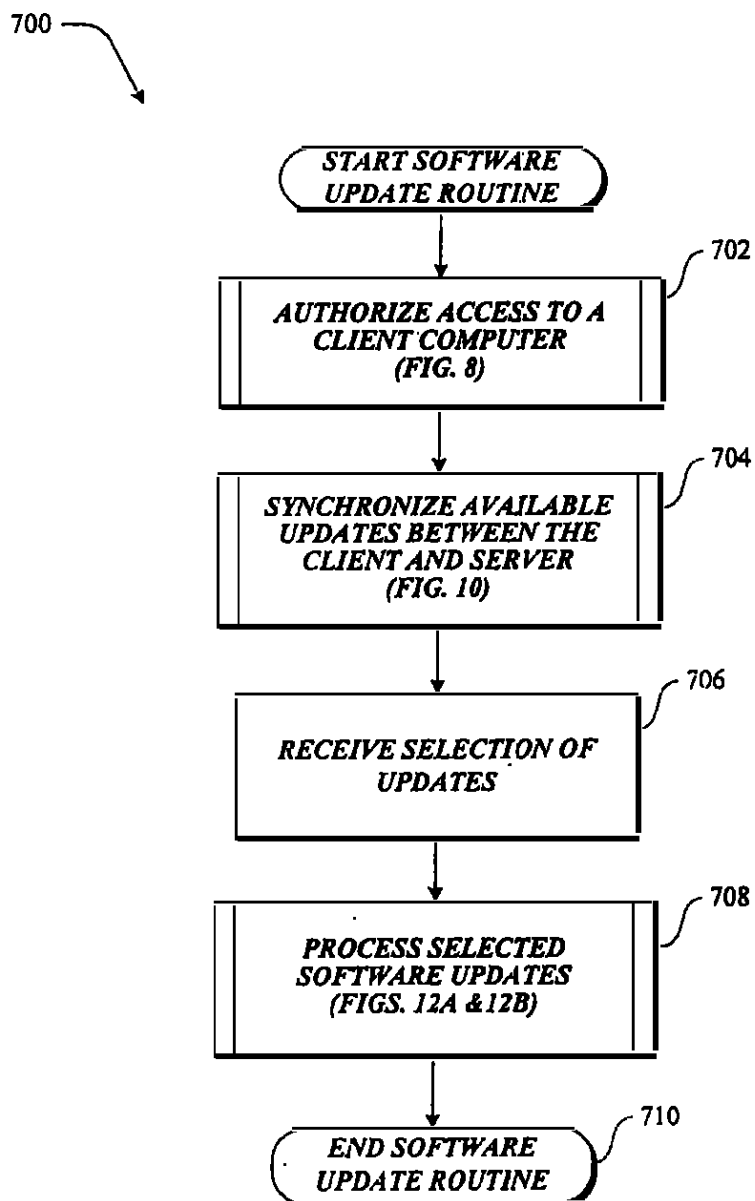


Fig. 7

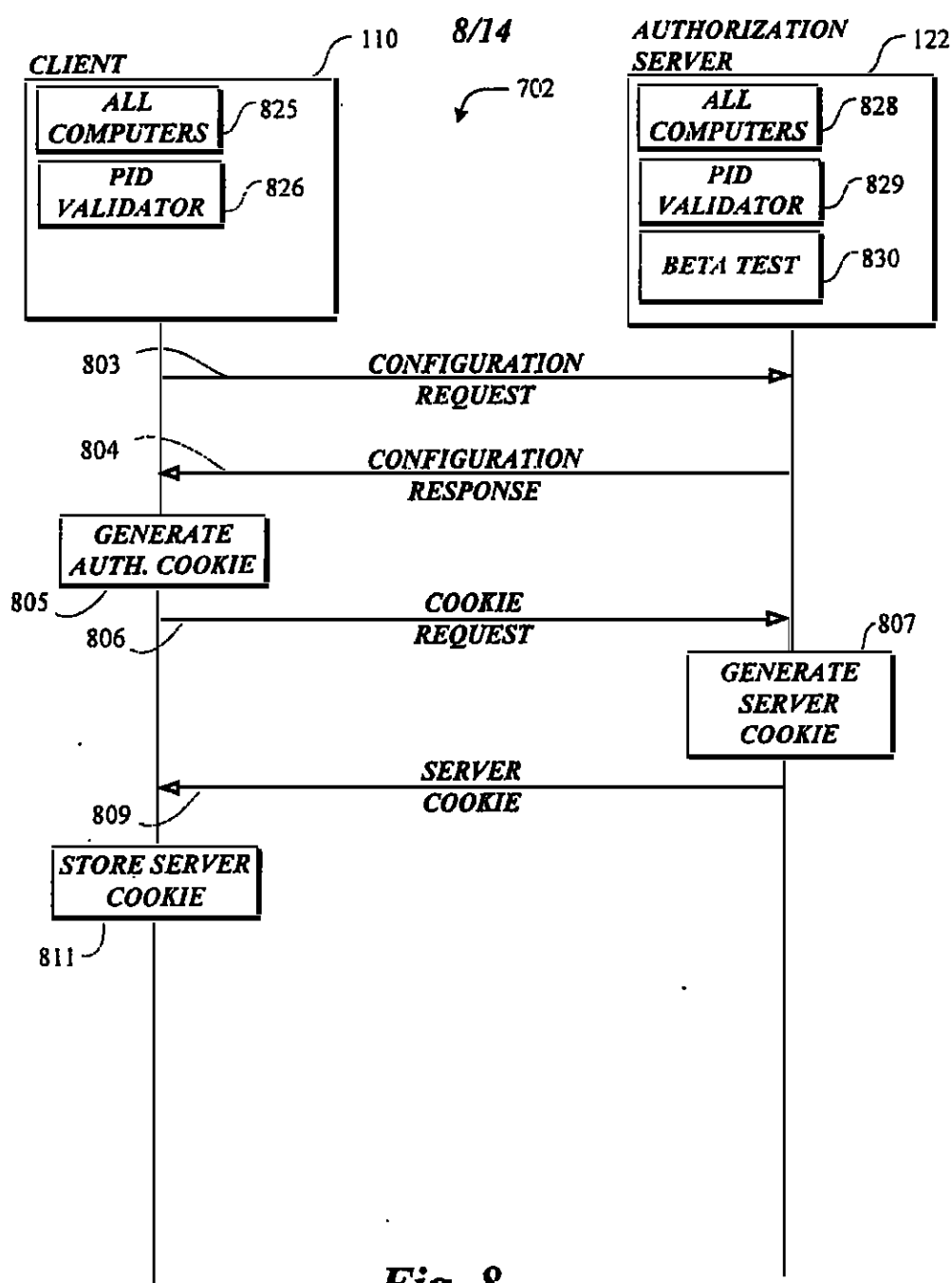


Fig. 8

140

9/14

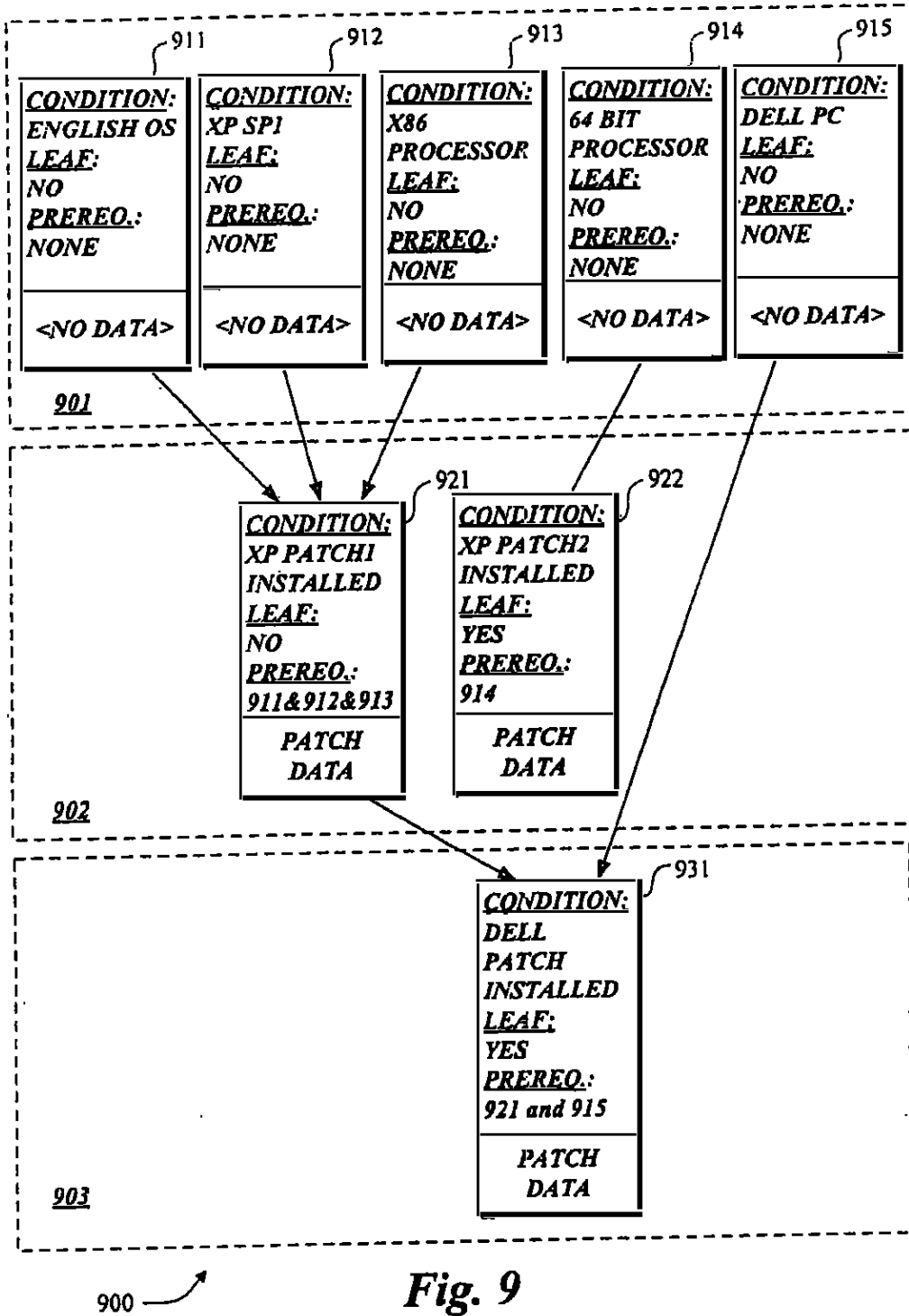


Fig. 9

141

10/14

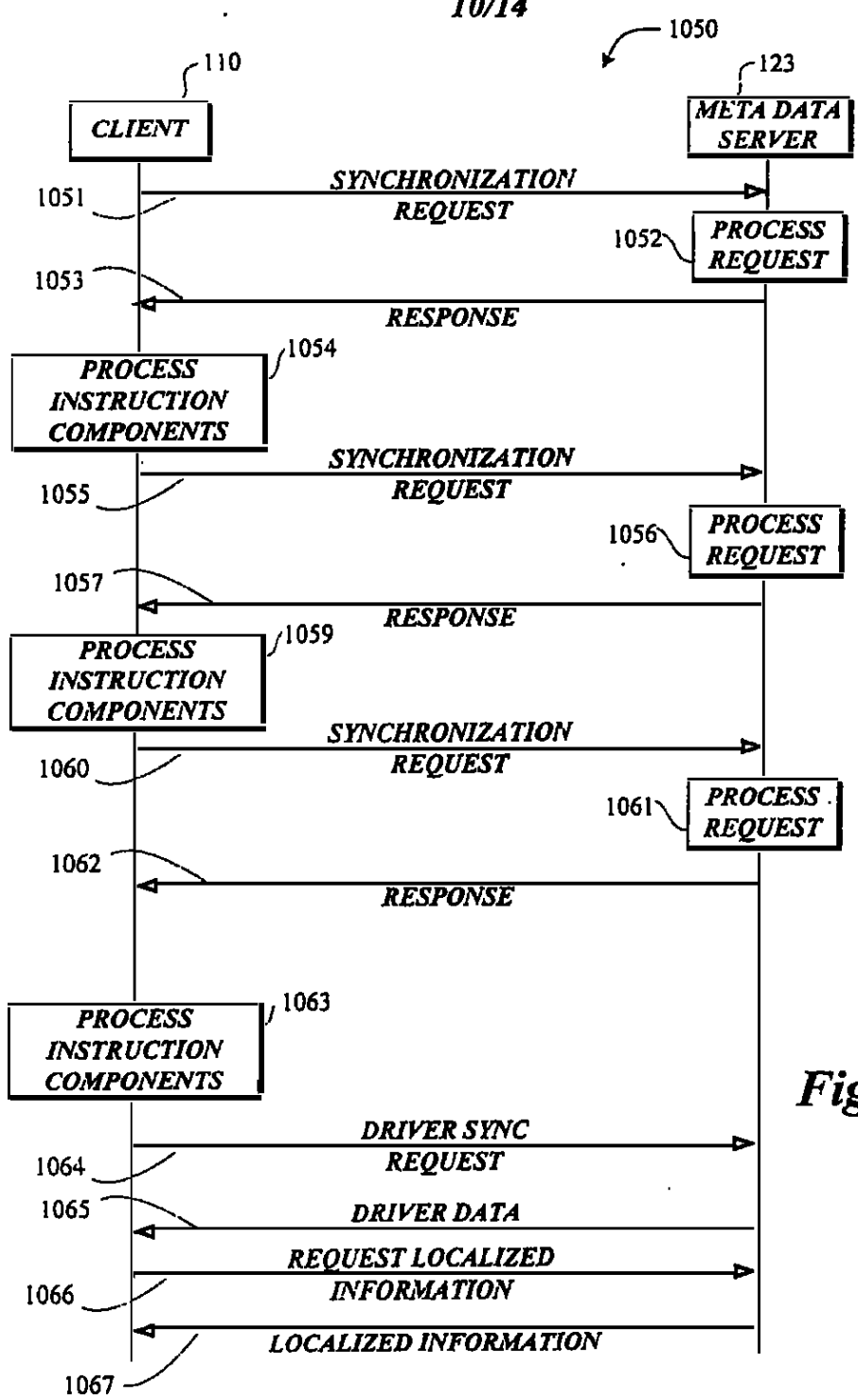


Fig.10

142

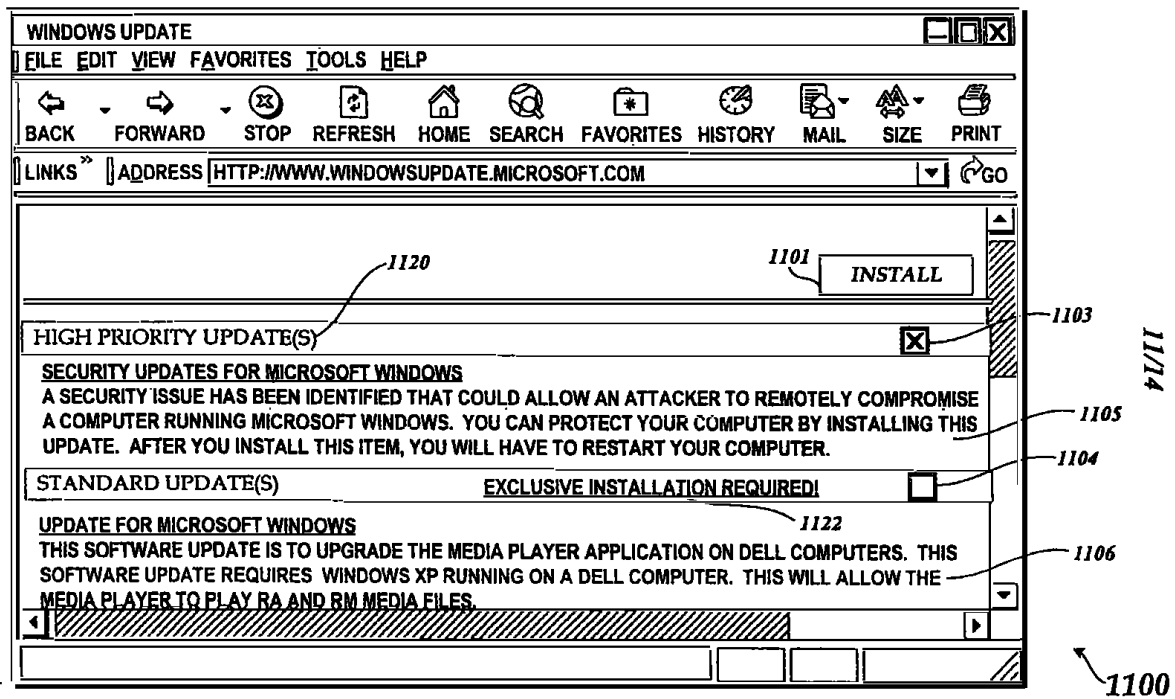


Fig. 11

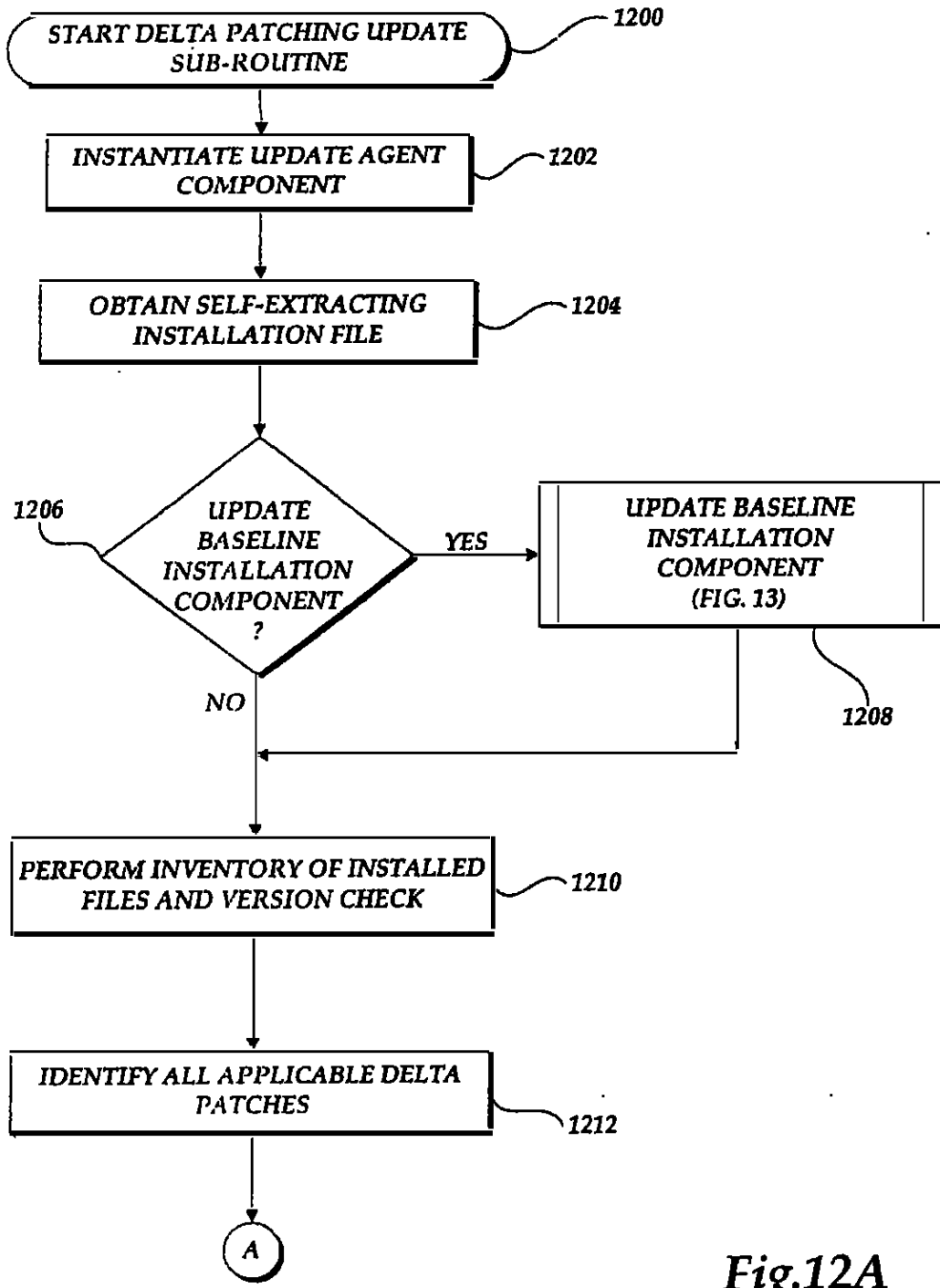


Fig.12A

144

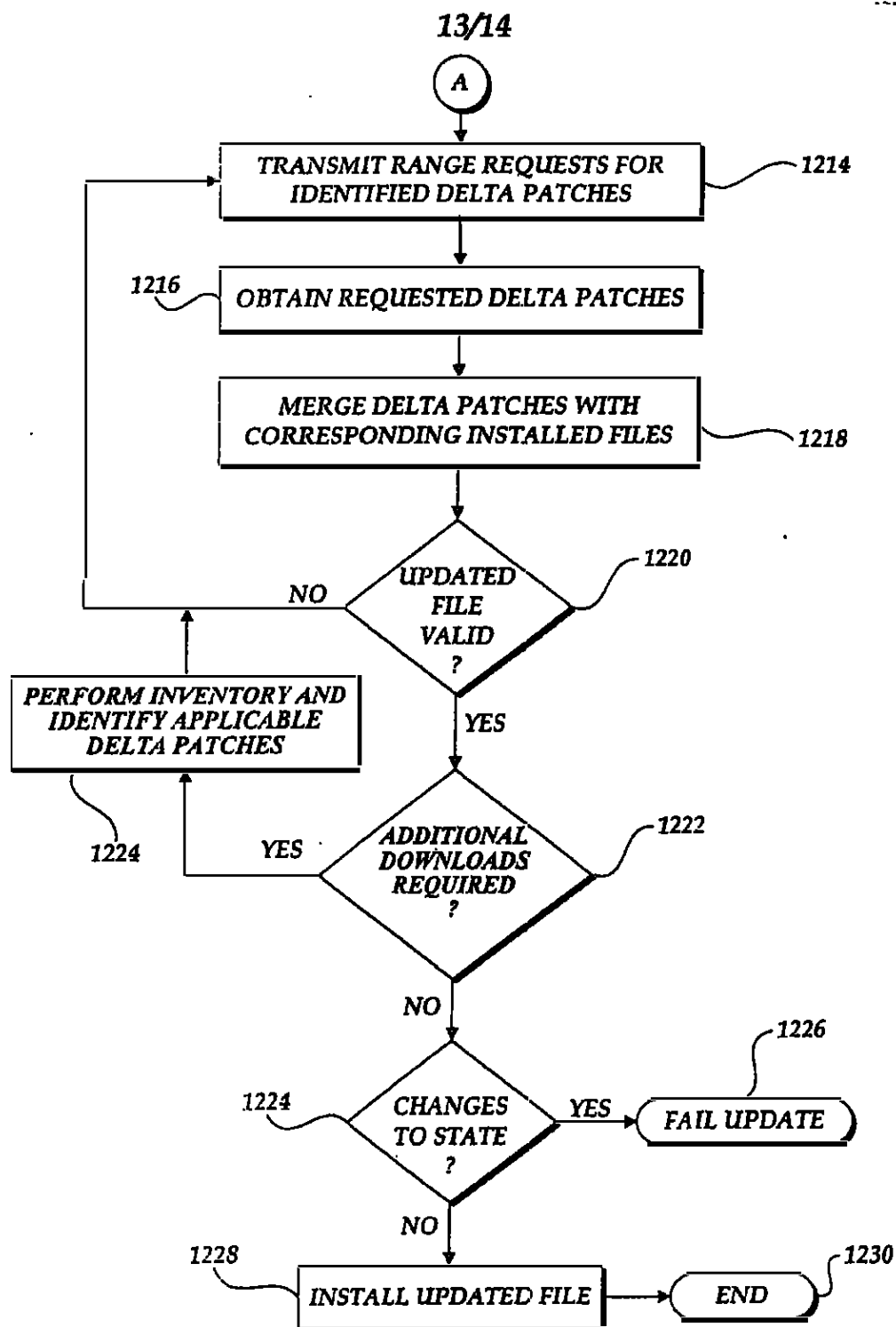


Fig.12B

145

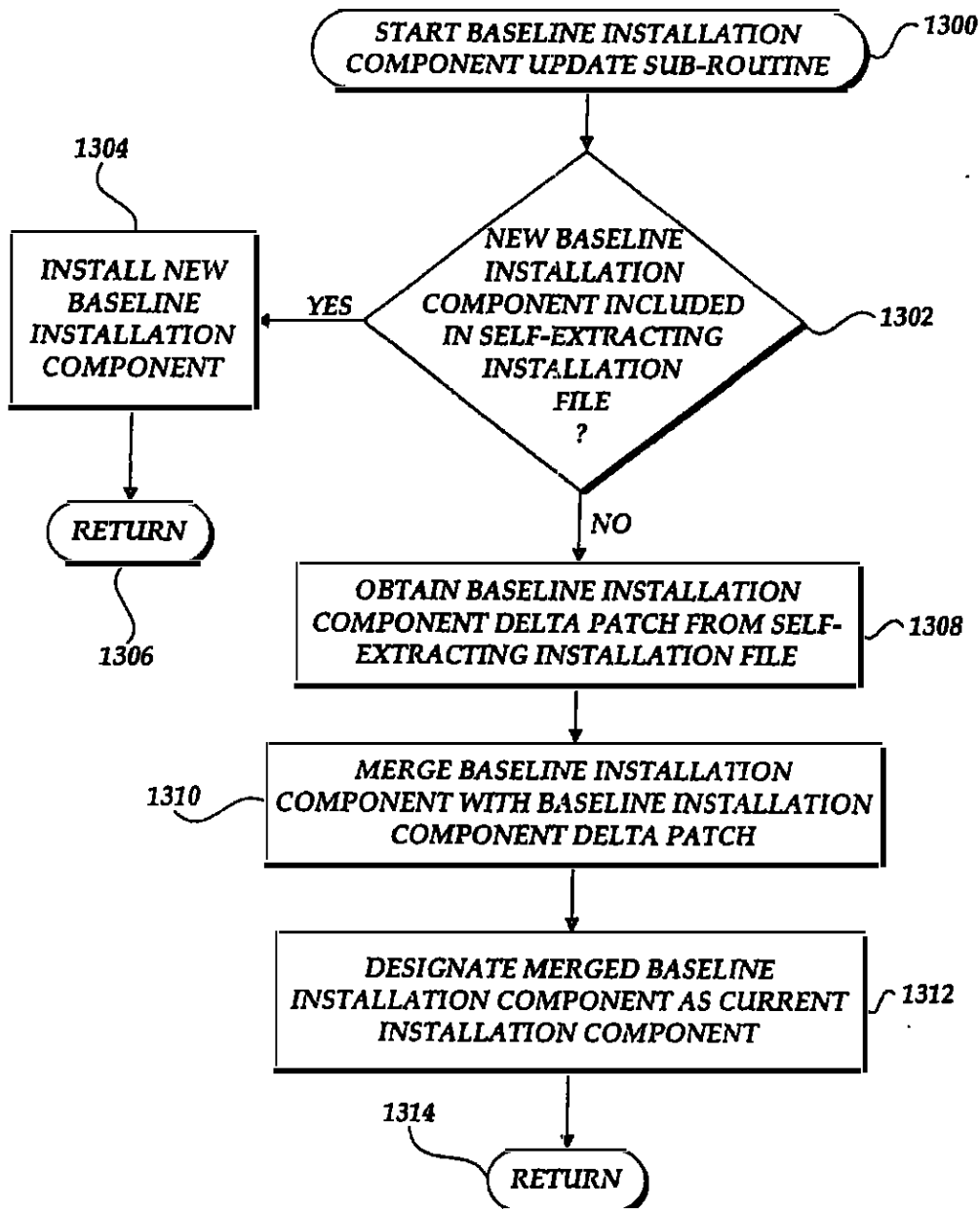


Fig.13



APOSTILLE

(Convention de La Haye du 5 octobre 1961)

1. Country: United States of America

This public document

2. has been signed by W. Montgomery

3. acting in the capacity of Certifying Officer

4. bears the seal/stamp of Patent and Trademark Office

Certified

5. at Washington, D.C.

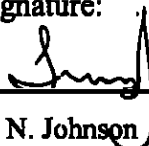
6. the second of December, 2004

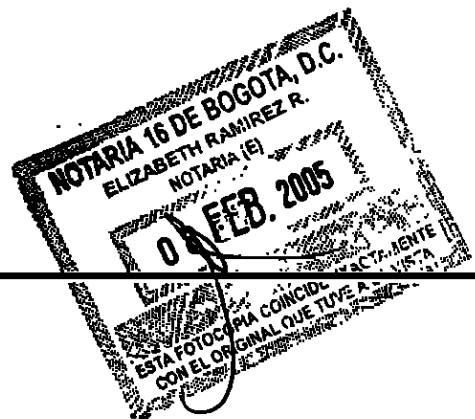
7. by Assistant Authentication Officer, United States Department of State

8. No. 05005024-4

9. Seal/Stamp:

10. Signature:


Sonya N. Johnson



A 1251524



UNITED STATES OF AMERICA

TO ALL TO WHOM THESE PRESENTS SHALL COME:

**UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office**

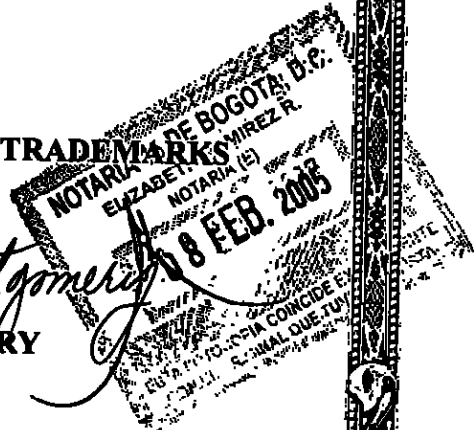
November 24, 2004

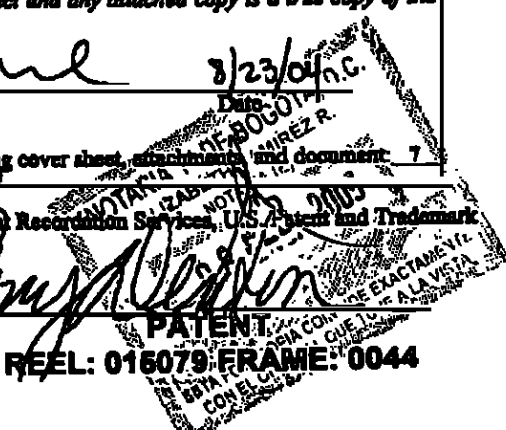
**THIS IS TO CERTIFY THAT ANNEXED IS A TRUE COPY FROM THE
RECORDS OF THIS OFFICE OF A DOCUMENT RECORDED ON
August 23, 2004.**

**By Authority of the
COMMISSIONER OF PATENTS AND TRADEMARKS**



W. Montgomery
W. MONTGOMERY
Certifying Officer



RECORDATION FORM COVER SHEET PATENTS ONLY		Patent and Trademark Office Attorney Docket No.: MSFT121845
Director - U.S. Patent and Trademark Office: Please record the attached original documents or copy thereof.		
1. Conveyance Type: ☒ New ☐ Re submission (Non-Recordation) Document ID# _____ Correction of Error: Real _____, Frame _____	2. Conveyance Type: ☒ Assignment ☐ Security Agreement ☐ License ☐ Change of Name ☐ Merger ☐ Other:	
3. Names of conveying parties: Execution Date: Michael Edward Meulemans June 22, 2004 Aaron Averbuch June 23, 2004 Jason Roberts August 13, 2004 Ken Showman June 29, 2004 Mazhar Mohamed June 22, 2004 Joseph G. Dadaio June 22, 2004 Additional name(s) of conveying party(ies) attached? ☐ Yes ☒ No	4. Name and address of receiving party: Name: Microsoft Corporation Address: One Microsoft Way Redmond, Washington 98052 Additional name(s) & address(es) attached? ☐ Yes ☒ No	
Application number(s) and/or patent number(s): A. Patent Application No(s). 10/737,708 B. Patent No(s). Additional numbers attached? ☐ Yes ☒ No If this document is being filed together with a new patent application, enter the date the patent application was signed by the first named executing inventor: _____		
6. Name and address of party to whom correspondence concerning document should be mailed: Mauricio A. Uribe, Esq. Customer No. 26389 CHRISTENSEN O'CONNOR JOHNSON KINDNESS LLC 1420 Fifth Avenue, Suite 2800 Seattle, WA 98101-2347 206.682.8100	7. Total number of applications and/or patents involved: 1 8. Total fee (37 C.F.R. 3.41): \$40.00 9. ☒ The fee is authorized to be charged to Deposit Account No. 03-1740. 10. The Director is hereby authorized to charge any fees under 37 C.F.R. § 3.41 which may be required by this paper, or credit any overpayment, to Deposit Account No. 03-1740.	
DO NOT USE THIS SPACE		
11. Statement and signature: To the best of my knowledge and belief, the foregoing information is true and correct and any attached copy is a true copy of the original document. Mauricio A. Uribe Name of Attorney or Agent Registration No. 46,206 Direct Dial 206.695.1728 Signature:  Date: 8/23/04 n.c. Total number of pages, including cover sheet, attachments, and document: 7		
I hereby certify that this correspondence is being transmitted via facsimile to Assignment Recordation Services, U.S. Patent and Trademark Office, at facsimile number 703-306-5995, on the below date. Date: August 23 2004 MAU:md MSFT121845 703109100  REEL: 016079 FRAME: 0044		

REEL: 016079 FRAME: 0044

149

150

8PM

CHRISTENSEN OCONNOR

NO. 9834 P. 3

THE UNITED STATES PATENT AND TRADEMARK OFFICE

Patent Number 10/737,708
Date December 15, 2003
Applicant Michael Edward Meulmans et al.
Assignor Microsoft Corporation
Patent Docket No. MSFT121845
**SYSTEM AND METHOD FOR MANAGING AND
COMMUNICATING SOFTWARE UPDATES**

PATENT ASSIGNMENT

CITIES TO THE ASSIGNMENT

ASSIGNEE

Michael Edward Meulmans
168 S.E. 100th Street
Arlington, Washington 98059

Jason Averbuch
230 Yale Ave. E., #D
Seattle, Washington 98102

Jason Roberts
103 College Avenue #2
Somerville, Massachusetts 02146

Ken Showman
15724 NE 117th Street
Redmond, Washington 98052

Mazhar Mohammed
127 - 242nd Court S.E.
Sammamish, Washington 98074

Joseph G. Daddie
9631 - 173rd Place NE
Redmond, Washington 98052

Microsoft Corp. 01/05/04



150

29PM

CHRISTENSEN OCONNOR

NO. 9834 P. 4

151

Microsoft Corporation
Corporation in the State of Washington
Microsoft Way
Redmond, Washington 98052

ASSIGNMENT

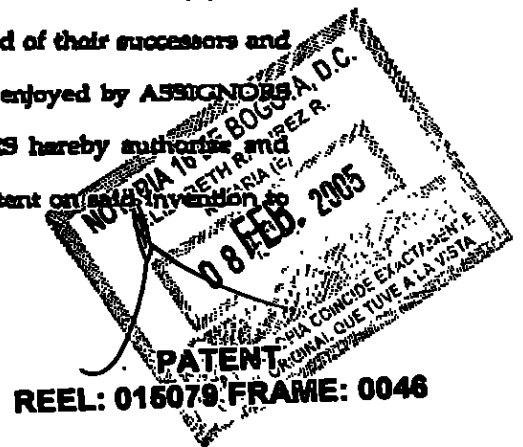
WHEREAS, ASSIGNORS (listed above) are inventors of an invention entitled "SYSTEM AND METHOD FOR MANAGING AND COMMUNICATING SOFTWARE UPDATES," as described and claimed in the specification forming part of an application for United States letters patent referenced above;

WHEREAS, Microsoft Corporation (hereinafter referred to as ASSIGNEE), a corporation of the State of Washington having a place of business at One Microsoft Way, Redmond, Washington 98052, is desirous of acquiring the entire right, title and interest in and to the invention and in and to any letters patent that may be granted therefor in the United States and in any and all foreign countries;

NOW, THEREFORE, in exchange for good and valuable consideration, the receipt of which is hereby acknowledged, ASSIGNORS hereby sell, assign and transfer unto ASSIGNEE, the entire right, title and interest in and to said invention, said application and any and all letters patent which may be granted for said invention in the United States of America and its territorial possessions and in any and all foreign countries, and in any and all divisions, reissues and continuations thereof, including the right to file foreign applications directly in the name of ASSIGNEE and to claim priority rights deriving from said United States application to which said foreign applications are entitled by virtue of international convention, treaty or otherwise, said invention, application and all letters patent on said invention to be held and enjoyed by ASSIGNEE and its successors and assigns for their use and benefit and of their successors and assigns as fully and entirely as the same would have been held and enjoyed by ASSIGNORS had this assignment, transfer and sale not been made. ASSIGNORS hereby authorize and request the Director of Patents and Trademarks to issue all letters patent on said invention to

DEPT-101-10101-101.DOC

Page 2 of 3



151

THE ASSIGNORS agree to execute all instruments and documents required for the
ag and prosecution of applications for United States and foreign letters patent on said
ation, for litigation regarding said letters patent, or for the purpose of protecting title to
invention or letters patent therefor.

6/22/04

Michael Edward Meulenhans
Michael Edward Meulenhans

6/23/04

Amron Averbuch
Amron Averbuch

Date

Jason Roberts
Jason Roberts

6/29/2004

Ken Showman
Ken Showman

Date

6/22/04

Mahzar Mohammed
Mahzar Mohammed

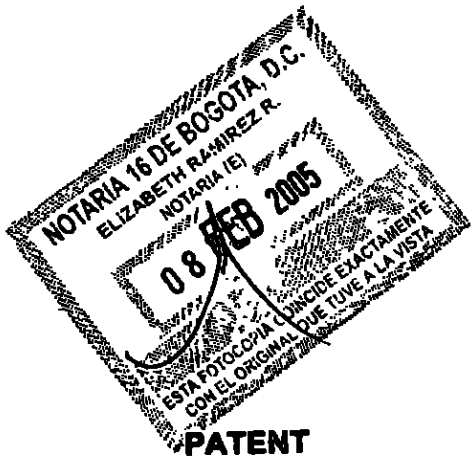
Date

6/22/04

Joseph C. Dadzia
Joseph C. Dadzia

Date

MAU:md



FS9

29PM

CHRISTENSEN OCONNOR

NO. 9834 P. 6

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

Application Number..... 10/737,708
Date..... December 15, 2009
Inventorship..... Michael Edward Meulemans et al
Assignee..... Microsoft Corporation
Attorney's Docket No. MSFT121845

SYSTEM AND METHOD FOR MANAGING AND
COMMUNICATING SOFTWARE UPDATES

PATENT ASSIGNMENT

PARTIES TO THE ASSIGNMENT

Assignors:

Michael Edward Meulemans
17168 S.E. 100th Street
Renton, Washington 98059

Aaron Averbuch
2230 Yale Ave. E., #D
Seattle, Washington 98102

Jason Roberts
168 College Avenue #2
Somerville, Massachusetts 02144

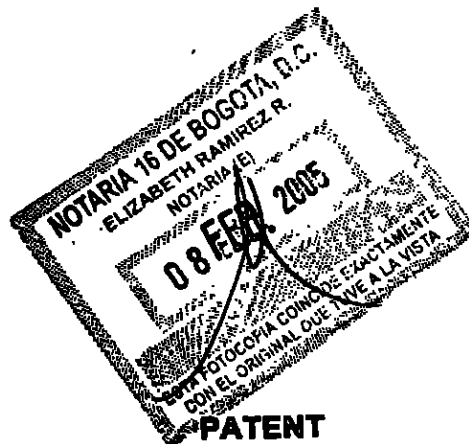
1310 W Chapel Hill St
Durham, NC 27701

JJR

Ken Showman
15724 NE 117th Street
Redmond, Washington 98052

Mazhar Mohammed
127 - 242nd Court S.E.
Sammamish, Washington 98074

Joseph G. Dadzie
9631 - 173rd Place NE
Redmond, Washington 98052



Page 1 of 2

PATENT
REEL: 015079 FRAME: 0048

152

154

~~agent:~~

Microsoft Corporation
Corporation in the State of Washington
One Microsoft Way
Redmond, Washington 98052

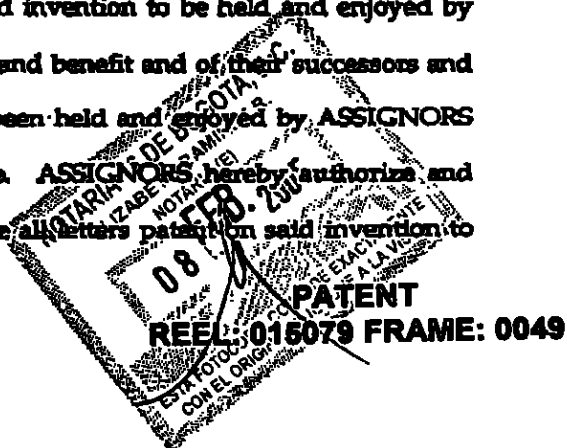
AGREEMENT

WHEREAS, ASSIGNORS (listed above) are inventors of an invention entitled "SYSTEM AND METHOD FOR MANAGING AND COMMUNICATING SOFTWARE UPDATES," as described and claimed in the specification forming part of an application for United States letters patent referenced above;

WHEREAS, Microsoft Corporation (hereinafter referred to as ASSIGNEE), a corporation of the State of Washington having a place of business at One Microsoft Way, Redmond, Washington 98052, is desirous of acquiring the entire right, title and interest in and to the invention and in and to any letters patent that may be granted therefor in the United States and in any and all foreign countries;

NOW, THEREFORE, in exchange for good and valuable consideration, the receipt of which is hereby acknowledged, ASSIGNORS hereby sell, assign and transfer unto ASSIGNEE, the entire right, title and interest in and to said invention, said application and any and all letters patent which may be granted for said invention in the United States of America and its territorial possessions and in any and all foreign countries, and in any and all divisions, reissues and continuations thereof, including the right to file foreign applications directly in the name of ASSIGNEE and to claim priority rights deriving from said United States application to which said foreign applications are entitled by virtue of international convention, treaty or otherwise, said invention, application and all letters patent on said invention to be held and enjoyed by ASSIGNEE and its successors and assigns for their use and benefit and of their successors and assigns as fully and entirely as the same would have been held and enjoyed by ASSIGNORS had this assignment, transfer and sale not been made. ASSIGNORS hereby authorize and request the Director of Patents and Trademarks to issue all letters patent on said invention to

Page 2 of 2



154

Assignee:

Microsoft Corporation
Corporation in the State of Washington
One Microsoft Way
Redmond, Washington 98052

AGREEMENT

WHEREAS, ASSIGNORS (listed above) are inventors of an invention entitled "SYSTEM AND METHOD FOR MANAGING AND COMMUNICATING SOFTWARE UPDATES," as described and claimed in the specification forming part of an application for United States letters patent referenced above;

WHEREAS, Microsoft Corporation (hereinafter referred to as ASSIGNEE), a corporation of the State of Washington having a place of business at One Microsoft Way, Redmond, Washington 98052, is desirous of acquiring the entire right, title and interest in and to the invention and in and to any letters patent that may be granted therefor in the United States and in any and all foreign countries;

NOW, THEREFORE, in exchange for good and valuable consideration, the receipt of which is hereby acknowledged, ASSIGNORS hereby sell, assign and transfer unto ASSIGNEE, the entire right, title and interest in and to said invention, said application and any and all letters patent which may be granted for said invention in the United States of America and its territorial possessions and in any and all foreign countries, and in any and all divisions, reissues and continuations thereof, including the right to file foreign applications directly in the name of ASSIGNEE and to claim priority rights deriving from said United States application to which said foreign applications are entitled by virtue of international convention, treaty or otherwise, said invention, application and all letters patent on said invention to be held and enjoyed by ASSIGNEE and its successors and assigns for their use and benefit and of their successors and assigns as fully and entirely as the same would have been held and enjoyed by ASSIGNORS had this assignment, transfer and sale not been made. ASSIGNORS hereby authorize and request the Director of Patents and Trademarks to issue all letters patent on said invention to

Page 2 of 2

REEL: 016079 FRAME: 0049

156

23. 2004 3:30PM

CHRISTENSEN OCONNOR

NO. 9834 P. 8

ASSIGNEE. ASSIGNORS agree to execute all instruments and documents required for the making and prosecution of applications for United States and foreign letters patent on said invention, for litigation regarding said letters patent, or for the purpose of protecting title to said invention or letters patent therefor.

Date

Michael Edward Meulemans

Date

Aaron Averbuch

8/13/04
Date

Jason Roberts
Jason Roberts

Date

Ken Showman

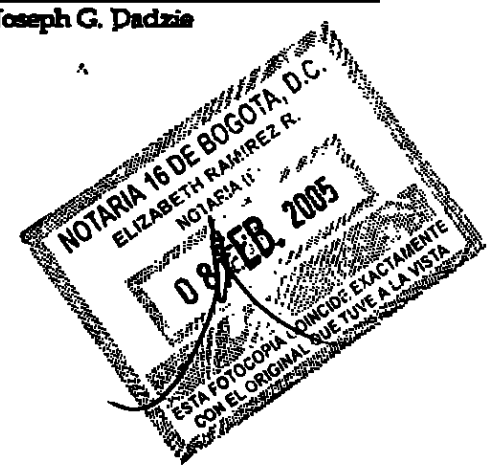
Date

Mazhar Mohammed

Date

Joseph G. Dadzie

MAU:md



REC RDED: 08/23/2004

PATENT
REEL: 016079 FRAME: 0060

156

TRADUCCIÓN DEL INGLÉS AL ESPAÑOL DE UNA APOSTILLA ORIGINAL ANEXA A UN DOCUMENTO ENTREGADO EN UN FOLDER DE LA OFICINA DE PATENTES Y MARCAS REGISTRADAS DEL DEPARTAMENTO DE COMERCIO DE LOS ESTADOS UNIDOS, CON COBERTURA DE PLÁSTICO TRANSPARENTE, REALIZADA POR NORA JIMÉNEZ DE ADAM, TRADUCTORA E INTÉRPRETE OFICIAL. TEL: 571-6917116; FEBRERO 1 DE 2005

"APOSTILLE

(Convención De La Haya de octubre 5 de 1961)

- 1. País: ESTADOS UNIDOS DE AMÉRICA
- 2. Este documento público fue firmado por: W. MONTGOMERY
- 3. En su calidad de: FUNCIONARIO DE CERTIFICACIONES
- 4. Lleva el sello de: LA OFICINA DE PATENTES Y MARCAS REGISTRADAS

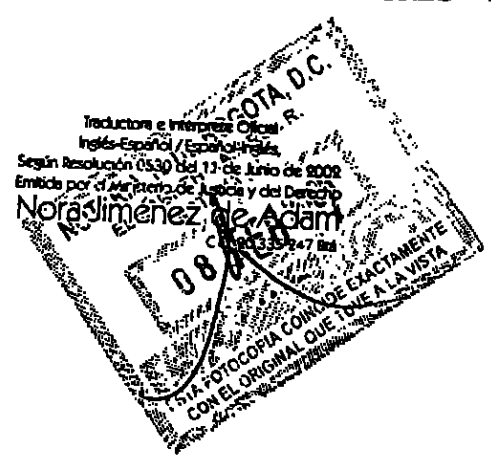
CERTIFICADO

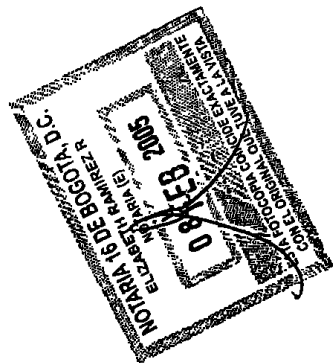
- 5. En: WASHINGTON, D.C.
- 6. El día: dos de diciembre de 2004
- 7. Por: ASISTENTE DEL FUNCIONARIO DE AUTENTICACIONES DEL DEPARTAMENTO DE ESTADO DE ESTADOS UNIDOS
- 8. No. 05005024-4
- 9. Sello: SELLO SECO DEL DEPARTAMENTO DE ESTADO DE ESTADOS UNIDOS
- 10. Firma: SONYA N. JOHNSON

CERTIFICO QUE ESTA ES UNA TRADUCCION FIEL DEL INGLES AL ESPAÑOL DE UN DOCUMENTO QUE HE TENIDO A LA VISTA; QUE SOY TRADUCTORA E INTERPRETE OFICIAL ACREDITADA POR RESOLUCIONES 0035 Y 530 DE LA UNIVERSIDAD NACIONAL DE COLOMBIA Y DEL MINISTERIO DE LA JUSTICIA RESPECTIVAMENTE, Y QUE ESTOY REGISTRADA ANTE EL MINISTERIO DE RELACIONES EXTERIORES DE COLOMBIA

Bogotá, 1° de febrero de 2005

Nora Jiménez de Adam
NORA JIMÉNEZ DE ADAM
C.C. 20.335.247





NO SE ASUMI
RESPONSABILIDAD DEL TEXTO
2005 FEB - 7 A 11: 53
JEFE DE LEGALIZACIONES
SANTAFE DE BOGOTÁ D.C.

MINISTERIO DE RELACIONES
EXTERIORES
086632
Cuya firma aparece en el
DOCUMENTO DESEMPAÑA
LAS FUENTES INDICADAS

TRADUCCIÓN DEL INGLÉS AL ESPAÑOL DE OCHO PÁGINAS DE UN DOCUMENTO ORIGINAL ENTREGADO EN UN FOLDER DE LA OFICINA DE PATENTES Y MARCAS REGISTRADAS DEL DEPARTAMENTO DE COMERCIO DE LOS ESTADOS UNIDOS, CON COBERTURA DE PLÁSTICO TRANSPARENTE, REALIZADA POR NORA JIMÉNE DE ADAM, TRADUCTORA E INTÉRPRETE OFICIAL. TEL: 571-6917116; FEBRERO 1 DE 2005

SEGUNDA PÁGINA: Emitida en papelería especial con reborde negro-gris con el número A 1251524 en la parte superior izquierda del documento: **ESTADOS UNIDOS DE AMÉRICA - A TODOS LOS BENEFICIARIOS DE ESTE DOCUMENTO: EL DEPARTAMENTO DE COMERCIO DE LOS ESTADOS UNIDOS** Oficina de Patentes y Marcas Registradas de los Estados Unidos

Noviembre 24 de 2004

ESTE DOCUMENTO CERTIFICA QUE LOS ANEXOS SON COPIA FIEL DEL DOCUMENTO QUE FUE REGISTRADO EN ESTA OFICINA EL DÍA 23 DE AGOSTO DE 2004.

Estampilla dorada del Departamento de Comercio - Oficina de Patentes y Marcas Registradas de los Estados Unidos

Bajo la potestad del COMISIONADO DE PATENTES Y MARCAS REGISTRADAS

(Firma)

W. MONTGOMERY

Funcionario de Certificaciones

TERCERA PÁGINA: Papel blanco con información del envío de fax: 23-2004, 3:28PM; CHRISTENSEN O'CONNOR No. 9834, P.2. Al margen derecho se encuentra al número CH \$40.00 031740 10737708

FORMATO DE PORTADA DE REGISTRO LEGAL

SÓLO PARA PATENTES



7

Oficina de Patentes y Marcas Registradas

PATENTE NO. DE EXPEDIENTE DEL ABOGADO: MSFT121845

PARA: Director de la Oficina de Patentes y Marcas Registradas: Por favor registre el (los) documento (s) originales anexos o copia (s).

1. Tipo de Solicitud

☒ Nueva

☐ Re-emisión (no de registro legal)

No. de Identificación del documento

☐ Corrección de Error PTO.

No. de Carrete No. de Pantalla

2. Tipo de Transferencia

☒ Cesión

☐ Licencia

☐ Fusión

☐ Acuerdo de Garantía

☐ Cambio de Razón Social

☐ Otros:

3. Parte (s) que Transfieren

Nombre	Fecha (s) de Ejecución
1. Michael Edward Meulemanis	Junio 22, 2004
2. Aaron Averbuch	Junio 23, 2004
3. Jason Roberts	Agosto 13, 2004
4. Ken Showman	Junio 29, 2004
5. Mazhar Mohammed	Junio 22, 2004
6. Joseph G. Dadzie	Junio 22, 2004

¿Se anexa (n) nombre (s) adicional (es)? ☐ Sí ☒ No

4. Nombre y Dirección de la Parte Receptora

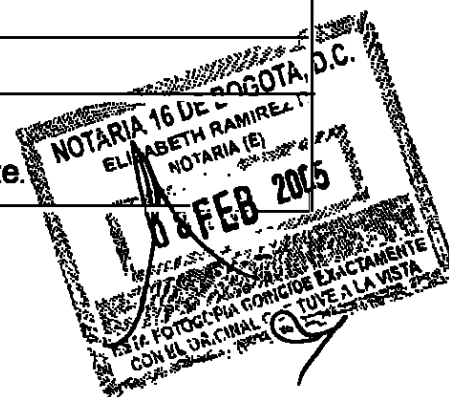
Nombre:	MICROSOFT CORPORATION
Dirección:	One Microsoft Way
Ciudad:	Redmond
Estado/País:	WASHINGTON
Código Postal:	98052

¿Se anexa (n) nombre (s) adicional (es)? ☐ Sí ☒ No

5. Número (s) de Solicitud y/o número (s) de patente:

A. No. (s) de Patente. 10/737,708

B. No. (s) de Patente.



¿Se anexan números adicionales? <u> </u> Sí <u>X</u> No . Si este documento está siendo presentado junto con una nueva solicitud, ingrese la fecha en que la primera solicitud fue firmada.								
6. Nombre y Dirección del Correspondiente Mauricio A. Uribe, Esq. Customer No. 26389 CHRISTENSEN O'CONNOR JOHNSON KINDNESS PLLC 1420 Fifth Avenue, Suite 2800 Seattle, WA 98101-2347 206.682.8100	7. Número total de solicitudes y/o patentes involucradas: 1 8. Costo total: (37 C.F.R. 3.41): \$40.00 9. <u>X</u> se autoriza a debitar la cuenta de Depósito No. 03-1740 para el pago 10. el Director queda autorizado por el presente para debitar las sumas cualesquiera bajo 37 C.F.R. 3.41, que puedan ser requeridas por este documento, o acreditar cualquier sobrepago, a la Cuenta de Depósito No. 03-1740							
NO UTILICE ESTE ESPACIO								
11. Declaración y Firma: <i>A mi saber, la siguiente información es verdadera y correcta y cualquier copia anexa es una copia fiel al documento original.</i> <table><tr><td><u>Mauricio A. Uribe</u></td><td><u>(Firma)</u></td><td><u>8/23/04</u></td></tr><tr><td>Nombre del Abogado o Agente</td><td>Firma</td><td>Fecha</td></tr></table> No. de Registro 46,206 Teléfono Directo 206.695.1728 No. total de páginas, incluyendo cubierta, anexos y documento: 7			<u>Mauricio A. Uribe</u>	<u>(Firma)</u>	<u>8/23/04</u>	Nombre del Abogado o Agente	Firma	Fecha
<u>Mauricio A. Uribe</u>	<u>(Firma)</u>	<u>8/23/04</u>						
Nombre del Abogado o Agente	Firma	Fecha						

Certifico que esta correspondencia está siendo transmitida vía fax al Assignment Recordation Service de la Oficina de Patentes y Marcas Registradas, al número de fax 703-306-5995 en la fecha indicada a continuación

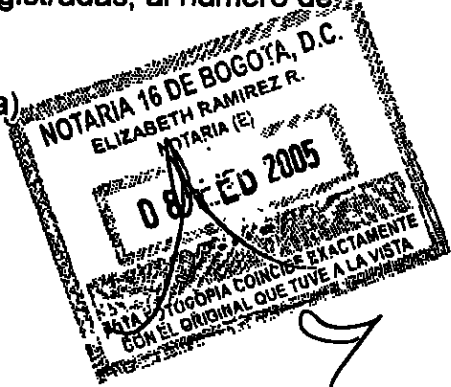
Fecha: agosto 23, 2004

MAU:md

700109100

PATENTE

(Firma)



Carrete: 015079; Pantalla: 0044

CUARTA PÁGINA: Papel blanco con información del envío de fax: 23-2004,
3:28PM; CHRISTENSEN O'CONNOR No. 9834, P.3.

**EN LA OFICINA DE PATENTES Y MARCAS REGISTRADAS DE LOS
ESTADOS UNIDOS**

Número de solicitud...10/737,708

Fecha de Solicitud...diciembre 15 de 2003

Autoría del Invento...Michael Edward Meulemans et al.

Cesionario...Microsoft Corporation

Expediente del Abogado No....MSFT121845

Título...SYSTEM AND METHOD FOR MANAGING AND COMMUNICATING
SOFTWARE UPDATES (Sistema y Método para la Administración y
Comunicación de Actualizaciones de Software)

CESIÓN DE LA PATENTE

PARTES DE LA CESIÓN

Cedente (s):

Michael Edward Meulemans

17168 SE 100th Street

Kenton, Washington 98059

Aaron Averbuch

2230 Yale Ave. E., #D

Seattle, Washington 98102

Jason Roberts

103 College Avenue #2

Somerville, Massachussets 02144



Ken Showman
15724 NE 117th Street
Redmond, Washington 98052

Mazhar Mohammed
127-242nd Court S.E.
Sammamish, Washington 98074

Joseph G. Dadzie
9631-173rd Place NE
Redmond, Washington 98052

Página 1 de 3

PATENTE

Carrete: 015079; Pantalla: 0045

QUINTA PÁGINA: Papel blanco con información del envío de fax: 23-2004,
3:29PM; CHRISTENSEN O'CONNOR No. 9834, P.4.

Cesionario:

Microsoft Corporation
Corporation in the State of Washington
One Microsoft Way
Redmond, WA 98052

CONVENIO

CONSIDERANDO QUE EL (LOS) CEDENTE (S) (mencionados anteriormente)
son el (los) creador (es) de un invento llamado "Sistema y Método para la
Administración y Comunicación de Actualizaciones de Software", de acuerdo con
lo descrito y afirmado en la especificación que hace parte del formulario para
solicitud de las patentes de Estados Unidos arriba indicados.
CONSIDERANDO, que Microsoft Corporation (en adelante, el CESIONARIO) una



corporación del estado de Washington, cuya sede es One Microsoft Way, Redmond, WA 98052, intenta adquirir la totalidad de los derechos, títulos e intereses en el invento y por éste y en o por cualquier patente que pueda serle otorgado en los Estados Unidos y en cualquiera y todos los países extranjeros; AHORA, POR LO TANTO, a cambio de contraprestaciones onerosas, el recibo de las cuales se reconocen en este documento, los CEDENTES por el presente venden, ceden y transfieren al CESIONARIO la totalidad de los derechos, títulos e intereses en y por dicho invento, solicitud y cualesquiera y todas las patentes que puedan ser otorgadas sobre el mismo en los Estados Unidos de América y sus posesiones territoriales, así como en todos y cualesquiera países extranjeros, y en todos y cualesquiera divisiones, re-emisiones y extensiones de la misma, incluyendo el derecho a registrar solicitudes en el extranjero directamente a nombre del CESIONARIO y a reclamar derechos prioritarios derivados de dicha solicitud en los Estados Unidos, a los cuales tienen derecho dichas solicitudes extranjeras en virtud de convenios internacionales, tratados u otras maneras; dicho invento, solicitud, y todas las patentes sobre el mismo serán conservados y en posesión del CESIONARIO y sus sucesores y apoderados, para su uso y beneficio y el de sus sucesores y apoderados, tan completa y totalmente como si fuesen conservados y en posesión de los CEDENTES, si esta cesión, transferencia o venta no se hubiera realizado. Los CEDENTES por el presente autorizan y solicitan al Comisionado de Patentes y Marcas Registradas emitir todas las patentes sobre dicho invento al CESIONARIO. Los CEDENTES acuerdan ejecutar todos los instrumentos y documentos requeridos para llevar a cabo y proseguir la solicitud de las patentes en los Estados Unidos y en el extranjero sobre dicho invento para fines de litigios relacionados con los mismos, o para proteger dicho título sobre el invento o patentes.

Página 2 de 3

PATENTE

Carrete: 015079; Pantalla: 0046



SEXTA PÁGINA: Papel blanco con información del envío de fax: 23-2004,
3:29PM; CHRISTENSEN O'CONNOR No. 9834, P.5.

6/22/ 04
Fecha

(Firma)
Michael Edward Meulemans

6/23/ 04
Fecha

(Firma)
Aaron Averbuch

(en blanco)
Fecha

(en blanco)
Jason Roberts

6/29/ 04
Fecha

(Firma)
Ken Showman

6/22/ 04
Fecha

(Firma)
Mazhar Mohammed

6/22/ 04
Fecha
MAU:md

(Firma)
Joseph G. Dadzie

Página 3 de 3

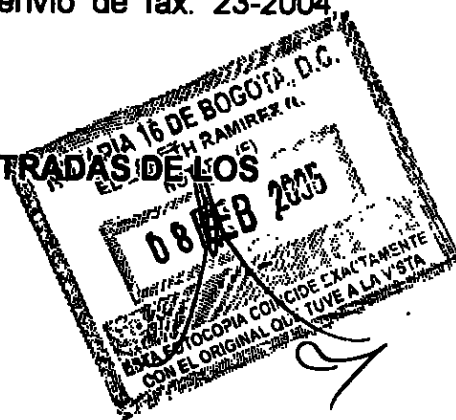
PATENTE

Carrete: 015079; Pantalla: 47

SÉPTIMA PÁGINA: Papel blanco con información del envío de fax: 23-2004,
3:29PM; CHRISTENSEN O'CONNOR No. 9834, P.6.

**EN LA OFICINA DE PATENTES Y MARCAS REGISTRADAS DE LOS
ESTADOS UNIDOS**

Número de solicitud...10/737,708



Fecha de Solicitud...diciembre 15 de 2003

Autoría del Invento...Michael Edward Meulemans et al.

Cesionario...Microsoft Corporation

Expediente del Abogado No.....MSFT121845

Título...SYSTEM AND METHOD FOR MANAGING AND COMMUNICATING
SOFTWARE UPDATES (Sistema y Método para la Administración y
Comunicación de Actualizaciones de Software)

CESIÓN DE LA PATENTE

PARTES DE LA CESIÓN

Cedente (s):

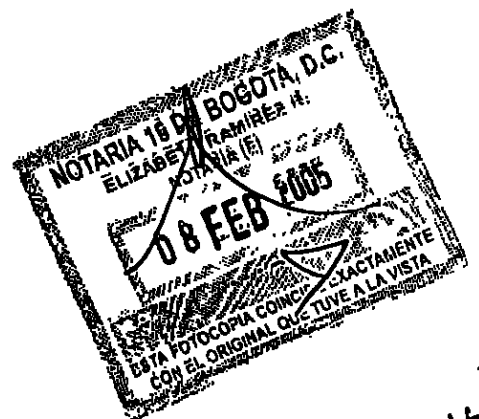
Michael Edward Meulemans
17168 SE 100th Street
Kenton, Washington 98059

Aaron Averbuch
2230 Yale Ave. E., #D
Seattle, Washington 98102

Jason Roberts (la dirección previa está tachada y aparece la siguiente a mano)
1310 W. Chapel Hill St. JJR
Durham, NC 27701

Ken Showman
15724 NE 117th Street
Redmond, Washington 98052

Mazhar Mohammed
127-242nd Court S.E.
Sammamish, Washington 98074



Joseph G. Dadzie
9631-173rd Place NE
Redmond, Washington 98052

Página 1 de 3

PATENTE

Carrete: 015079; Pantalla: 0048

OCTAVA PÁGINA: Papel blanco con información del envío de fax: 23-2004,
3:29PM; CHRISTENSEN O'CONNOR No. 9834, P.7.

Cesionario:

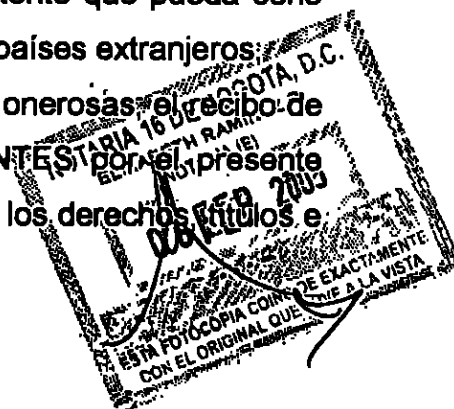
Microsoft Corporation
Corporation in the State of Washington
One Microsoft Way
Redmond, WA 98052

CONVENIO

CONSIDERANDO QUE EL (LOS) CEDENTE (S) (mencionados anteriormente)
son el (los) creador (es) de un invento llamado "Sistema y Método para la
Administración y Comunicación de Actualizaciones de Software", de acuerdo con
lo descrito y afirmado en la especificación que hace parte del formulario para
solicitud de las patentes de Estados Unidos arriba indicados;

CONSIDERANDO, que Microsoft Corporation (en adelante el CESIONARIO), una
corporación del estado de Washington, cuya sede es One Microsoft Way,
Redmond, WA 98052, intenta adquirir la totalidad de los derechos, títulos e
intereses en el invento y por éste y en o por cualquier patente que pueda serle
otorgado en los Estados Unidos y en cualquiera y todos los países extranjeros;

AHORA, POR LO TANTO, a cambio de contraprestaciones onerosas, el cedente
las cuales se reconocen en este documento, los CEDENTES por el presente
venden, ceden y transfieren al CESIONARIO la totalidad de los derechos, títulos e



Notario
17 de Junio de 2004
Justicia y el Derecho
Z de

168

intereses en y por dicho invento, solicitud y cualesquiera y todas las patentes que puedan ser otorgadas sobre el mismo en los Estados Unidos de América y sus posesiones territoriales, así como en todos y cualesquiera países extranjeros, y en todos y cualesquiera divisiones, re-emisiones y extensiones de la misma, incluyendo el derecho a registrar solicitudes en el extranjero directamente a nombre del CESIONARIO y a reclamar derechos prioritarios derivados de dicha solicitud en los Estados Unidos, a los cuales tienen derecho dichas solicitudes extranjeras en virtud de convenios internacionales, tratados u otras maneras; dicho invento, solicitud, y todas las patentes sobre el mismo serán conservados y en posesión del CESIONARIO y sus sucesores y apoderados, para su uso y beneficio y el de sus sucesores y apoderados, tan completa y totalmente como si fuesen conservados y en posesión de los CEDENTES, si esta cesión, transferencia o venta no se hubiera realizado. Los CEDENTES por el presente autorizan y solicitan al Comisionado de Patentes y Marcas Registradas emitir todas las patentes sobre dicho invento al CESIONARIO. Los CEDENTES acuerdan ejecutar todos los instrumentos y documentos requeridos para llevar a cabo y proseguir la solicitud de las patentes en los Estados Unidos y en el extranjero sobre dicho invento para fines de litigios relacionados con los mismos, o para proteger dicho título sobre el invento o patentes.

Página 2 de 3

PATENTE

Carrete: 015079; Pantalla: 0049

NOVENA PÁGINA: Papel blanco con información del envío de fax: 23-2004, 3:29PM; CHRISTENSEN O'CONNOR No. 9834, P8.

(en blanco)
Fecha

(en blanco)
Michael Edward Meuleman

(en blanco)
Fecha

(en blanco)
Aaron Averbuch



168

8/13/04
Fecha

(Firma)
Jason Roberts

(en blanco)
Fecha

(en blanco)
Ken Showman

(en blanco)
Fecha

(en blanco)
Mazhar Mohammed

(en blanco)
Fecha

(en blanco)
Joseph G. Dadzie

MAU:md
Registrado el 08/23/2004

PATENTE
Carrete: 015079; Pantalla: 0050

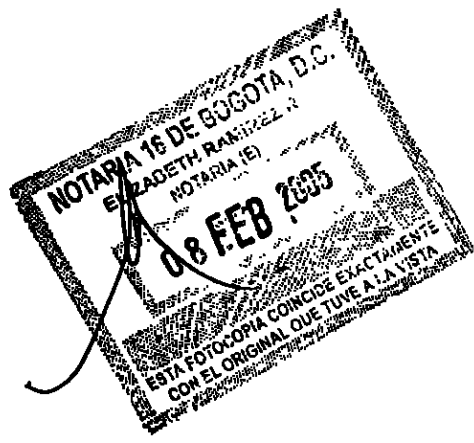
**CONTRAPORTADA: LOGO DEL DEPARTAMENTO DE COMERCIO DE
ESTADOS UNIDOS – OFICINA DE PATENTES Y MARCAS REGISTRADAS**

CERTIFICO QUE ESTA ES UNA TRADUCCION FIEL DEL INGLES AL
ESPAÑOL DE UN DOCUMENTO QUE HE TENIDO A LA VISTA; QUE SOY
TRADUCTORA E INTERPRETE OFICIAL ACREDITADA POR RESOLUCIONES
0035 Y 530 DE LA UNIVERSIDAD NACIONAL DE COLOMBIA Y DEL
MINISTERIO DE LA JUSTICIA RESPECTIVAMENTE, Y QUE ESTOY
REGISTRADA ANTE EL MINISTERIO DE RELACIONES EXTERIORES DE
COLOMBIA

Bogotá, 1° de febrero de 2005

Nora Jiménez de Adam
NORA JIMÉNEZ DE ADAM
C.C. 20.335.247

Inductora e intérprete Oficial
Inglés-Español / Español-Inglés
Según Resolución 0530 del 11 de Junio de 2002
Emitida por el Ministerio de Justicia y del Derecho
Nora Jiménez de Adam
C.C. 20.335.247 Bst



REQUISITOS NECESARIOS PARA PRESENTACION Y AGILIZACION DEL TRAMITE

	USUARIO	RADICADOR
PATENTE DE INVENCION	()	()
MODELO DE UTILIDAD	()	()
Indicación de que se solicita la concesión de una patente.	()	()
Datos de identificación del solicitante o de la persona que se presenta la solicitud.	()	()
Descripción de la invención.	()	()
Dibujos de ser estos pertinentes.	()	()
Comprobante de Pago de las tasas establecidas.	()	()

☒ **COMPLETA** ()
 INCOMPLETA ()

DISEÑO INDUSTRIAL ()

- Indicación de que se solicita el registro de Diseño Industrial () ()
- Datos de identificación del solicitante o de la persona que presenta la solicitud. () ()
- Representación gráfica o fotográfica del Diseño Industrial o muestra del Material que incorpora el diseño. () ()
- Comprobante de pago de las tasas establecidas () ()

COMPLETA ()
 INCOMPLETA ()

ESQUEMA DE TRAZADO ()

- Indicación de que solicita el registro de un Esquema de trazado. () ()
- Datos de identificación del solicitante o de la persona que presenta la solicitud. () ()
- Representación gráfica del esquema de trazado () ()
- Comprobante de pago de las tasas establecidas. () ()

COMPLETA ()
 INCOMPLETA ()

Firma Responsable:

PEAR

Fecha:

mayo 25 de 2005

PETITORIO



SUPERINDUSTRIA Y COMERCIO
Radicación: 03030610 09000000 Folios: 165
Fecha (AMH): 2005-03-25 11:06:16
Trámite: 011 PCT-PATENT 378 FASE NACI 413 PRESENTAR
Dependencia: 3020 DIVISION DE NUEVAS CREACIONES

FORMULARIO UNICO DE SOLICITUD DE PATENTE
PCT
ENTRADA EN FASE NACIONAL

①

SOLICITUD DE:

☒ Patente de Invención	☐ Patente de Modelo de Utilidad
☒ Capítulo I	☐ Capítulo II

②

SOLICITANTE
(71)

Nombre: MICROSOFT CORPORATION
Dirección: One Microsoft Way Redmond, Washington 98052, Estados Unidos de America
Nacionalidad o Domicilio: REDMOND, WASHINGTON, ESTADOS UNIDOS DE AMERICA.
Lugar de Constitución: Estado de Washington, Estados Unidos de América
Teléfono: 3123600 Fax: 3122410
E-mail: general@cardenasvcardenas.com

IDENTIFICACION

C.C. ☐ NIT ☐
C.E. ☐ Otro ☒
Cual _____
Número _____

③

REPRESENTANTE
O APODERADO

Nombre: DARIO CARDENAS NAVAS
Dirección: Carrera 7 No 71-52 Torre B Piso 9
Teléfono: 3123600 Fax: 3122410
E-Mail: general@cardenasycardenas.com

IDENTIFICACION

C.C. ☒ NIT ☐
C.E. ☐ Otro ☐
Cual _____
No 17.066.629 TP 1.250
BTA C.S.J.

④

INVENTOR(ES)
(72)

- 1.Nombre:
1. Michael Edward Meulemans
 2. Aaron Averbuch
 3. Jason Roberts
 4. Ken Showman
 5. Mazhar Mohammed
 6. Joseph G. Dadzie
2. Dirección:
1. One Microsoft Way Redmond, Washington 98052, Estados Unidos de America
 2. One Microsoft Way Redmond, Washington 98052, Estados Unidos de America
 3. One Microsoft Way Redmond, Washington 98052, Estados Unidos de America
 4. One Microsoft Way Redmond, Washington 98052, Estados Unidos de America
 5. One Microsoft Way Redmond, Washington 98052, Estados Unidos de America
 6. One Microsoft Way Redmond, Washington 98052, Estados Unidos de America
- 3.Nacionalidad o Domicilio:
1. REDMOND, WASHINGTON, ESTADOS UNIDOS DE AMERICA
 2. REDMOND, WASHINGTON, ESTADOS UNIDOS DE AMERICA
 3. REDMOND, WASHINGTON, ESTADOS UNIDOS DE AMERICA
 4. REDMOND, WASHINGTON, ESTADOS UNIDOS DE AMERICA
 5. REDMOND, WASHINGTON, ESTADOS UNIDOS DE AMERICA
 6. REDMOND, WASHINGTON, ESTADOS UNIDOS DE AMERICA

T

172

SISTEMA Y MÉTODO PARA ADMINISTRAR Y COMUNICAR ACTUALIZACIONES DE SOFTWARE

⑤ PCT (86)

Solicitud Internacional No. PCT/US2004/023975 Fecha Julio 23, 2004
Publicación Internacional No. Fecha

⑥ Título(54): SISTEMA Y MÉTODO PARA ADMINISTRAR Y COMUNICAR ACTUALIZACIONES DE SOFTWARE

⑦ Clasificación Internacional (51):

⑧ Prioridad

(33)País de Origen

(32)Fecha

(31)N° de Solicitud

SI ☒ NO ☐

U.S.A.

Diciembre 15, 2003

10/737.708

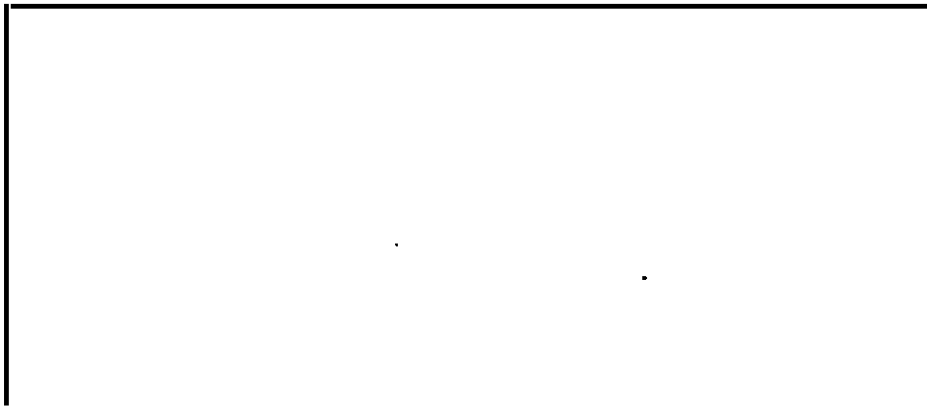
Comprobante de Pago No.: 05-37.299 Fecha Mayo 19, 2005

⑩

ANEXOS

- ☒ Comprobante de pago de la tasa de presentación de la solicitud.
- ☒ Documento que acredita la existencia y representación legal cuando el solicitante sea persona jurídica. (Protocolo 16.069)
- ☒ Poderes, si fuere el caso. (Protocolo 16.069)
- ☒ Documento de cesión del inventor al solicitante o a su causante.
- ☐ Declaraciones
- ☒ Copia de la solicitud internacional. (Resumen, descripción, reivindicaciones, dibujos)
- ☒ Traducción de la solicitud internacional al castellano. (Resumen, descripción, reivindicaciones, dibujos)
- ☐ Copia del examen preliminar internacional efectuado por la Autoridad Internacional de Examen Preliminar (IPEA).
- ☐ Traducción del examen preliminar internacional efectuado por la IPEA.
- ☐ De ser el caso, copia del contrato de acceso.
- ☐ De ser el caso, documento que acredite la licencia o autorización de uso de conocimientos tradicionales de las comunidades indígenas.
- ☐ De ser el caso, certificado de depósito del material biológico.
- ☒ Arte final 12x12.
- ☐ De ser el caso, información sobre otras solicitudes de patente o títulos obtenidos en el extranjero por el mismo titular o su causante, relacionadas parcial o totalmente con la invención de esta solicitud.
- ☒ De ser el caso, modificaciones efectuadas a la solicitud internacional.
- ☐ Otros.

11 FIGURA CARACTERISTICA



12

Solicito la Concesión de la patente,

NOMBRE: DARIO GARDENAS NAVAS

FIRMA:

C.C. 17.066.629 BTA TP. 1.250 CSJ

173

La implementación del invento en las cuales se reivindica propiedad exclusiva o privilegios se definen de la siguiente forma:

1. En un sistema de computación que incluye el dispositivo de cómputo de un cliente que está en comunicación con un servicio de actualización de software, un método para comunicar actualizaciones de software de un servicio de actualización de software al dispositivo de cómputo del cliente, método que consta de:

(a) obtener una solicitud de sincronización del dispositivo de cómputo del cliente al servicio de actualización de software, donde la solicitud de sincronización incluye un identificador de actualizaciones de software instaladas si el dispositivo de cómputo del cliente almacena el software de actualización instalado;

(b) determinar si la solicitud de sincronización tiene un identificador de por lo menos una actualización de software instalada;

(c) si determina que la solicitud de sincronización tiene un identificador de por lo menos una actualización de software instalada, seleccionar actualizaciones adicionales de software para comunicar al dispositivo de cómputo del cliente, donde la selección de actualizaciones de software adicionales depende del cumplimiento de prerequisites definidos en la actualización de software adicional, donde los prerequisites requieren que la solicitud de sincronización incluya un identificador de por lo menos una actualización de software instalada;

(d) si determina que la petición de sincronización no tiene un identificador para por lo menos una actualización de software instalada, seleccionar el primer nivel de actualizaciones de software para comunicar al dispositivo de cómputo del cliente, donde el primer nivel de actualizaciones de software no incluye un prerequisite;

(e) comunicar un componente de instrucciones de actualización del software seleccionado al dispositivo de cómputo del cliente de un servicio de actualización de software; y

56

124

(f) guardar el componente de instrucciones de la actualización de software seleccionada en el dispositivo de cómputo del cliente como una actualización de software instalada si el dispositivo de cómputo del cliente contiene por lo menos un componente que cumple una de las condiciones por de una regla de aplicabilidad guardada en la actualización de software seleccionada.

2. El método como se describió en la reivindicación 1, donde el método incluye además:

Determinar si por lo menos un componente de instrucción de la actualización de software seleccionada provee una indicación de que el servicio de actualizaciones de software tiene actualizaciones de software adicionales disponibles para el dispositivo de cómputo del cliente; y

Repitiendo los elementos entre (a)-(f) si se determina que por lo menos un componente de instrucciones del software de actualización seleccionado provee una indicación de que el servicio de actualización de software tiene actualizaciones de software adicionales disponibles para el dispositivo de cómputo del cliente.

3. El método como se describe en la reivindicación 1, donde el método consta además:

si el componente de instrucciones no da una indicación de que el servicio de actualización de software tiene actualizaciones de software adicionales disponibles para el dispositivo de cómputo del cliente:

comunicar un componente de datos localizado en la actualización de software seleccionada del servicio de actualización de software al dispositivo de cómputo del cliente; y;

generar una vista que contiene texto derivado del componente de datos localizado y comunicado, donde el texto da una descripción de la actualización de software seleccionada; y

en respuesta a haber recibido la orden, comunicar el archivo de datos asociado con la actualización de software seleccionada del servicio de actualización de software al dispositivo de cómputo del cliente.

34

La Implementación del Invento en las cuales se reivindica propiedad exclusiva o privilegios se definen de la siguiente forma:

1. En un sistema de computación que incluye el dispositivo de cómputo de un cliente que está en comunicación con un servicio de actualización de software, un método para comunicar actualizaciones de software de un servicio de actualización de software al dispositivo de cómputo del cliente, método que consta de:

(a) obtener una solicitud de sincronización del dispositivo de cómputo del cliente al servicio de actualización de software, donde la solicitud de sincronización incluye un identificador de actualizaciones de software instaladas si el dispositivo de cómputo del cliente almacena el software de actualización instalado;

(b) determinar si la solicitud de sincronización tiene un identificador de por lo menos una actualización de software instalada;

(c) si determina que la solicitud de sincronización tiene un identificador de por lo menos una actualización de software instalada, seleccionar actualizaciones adicionales de software para comunicar al dispositivo de cómputo del cliente, donde la selección de actualizaciones de software adicionales depende del cumplimiento de prerequisites definidos en la actualización de software adicional, donde los prerequisites requieren que la solicitud de sincronización incluya un identificador de por lo menos una actualización de software instalada;

(d) si determina que la petición de sincronización no tiene un identificador para por lo menos una actualización de software instalada, seleccionar el primer nivel de actualizaciones de software para comunicar al dispositivo de cómputo del cliente, donde el primer nivel de actualizaciones de software no incluye un prerequisite;

(e) comunicar un componente de instrucciones de actualización del software seleccionado al dispositivo de cómputo del cliente de un servicio de actualización de software; y

176

(f) guardar el componente de instrucciones de la actualización de software seleccionada en el dispositivo de cómputo del cliente como una actualización de software instalada si el dispositivo de cómputo del cliente contiene por lo menos un componente que cumple una de las condiciones por de una regla de aplicabilidad guardada en la actualización de software seleccionada.

2. El método como se describió en la reivindicación 1, donde el método incluye además:

Determinar si por lo menos un componente de instrucción de la actualización de software seleccionada provee una indicación de que el servicio de actualizaciones de software tiene actualizaciones de software adicionales disponibles para el dispositivo de cómputo del cliente; y

Repetiendo los elementos entre (a)-(f) si se determina que por lo menos un componente de instrucciones del software de actualización seleccionado provee una indicación de que el servicio de actualización de software tiene actualizaciones de software adicionales disponibles para el dispositivo de cómputo del cliente.

3. El método como se describe en la reivindicación 1, donde el método consta además:

si el componente de instrucciones no da una indicación de que el servicio de actualización de software tiene actualizaciones de software adicionales disponibles para el dispositivo de cómputo del cliente:

comunicar un componente de datos localizado en la actualización de software seleccionada del servicio de actualización de software al dispositivo de cómputo del cliente; y;

generar una vista que contiene texto derivado del componente de datos localizado y comunicado, donde el texto da una descripción de la actualización de software seleccionada; y

en respuesta a haber recibido la orden, comunicar el archivo de datos asociado con la actualización de software seleccionada del servicio de actualización de software al dispositivo de cómputo del cliente.

59