J.E.

Industria y Comercio

SUPERINTENDENCIA

SOLICITUD
PATENTE DE INVENCIÓN

EXPEDIENTE N° 04 9499 2

TITULO: INTERFACE DE PROGRAMACION PARA EL LICENCIAMIENTO

CLASIFICACION INTERNACIONAL: G06F 17/60

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
Radicacion : 04094972 00000000 Folios: 76
Fecha (AMD): 2004-09-23 17:00:27
Tramite : 002 PATENTES D 1 REGISTRO/ 411 PRESENTA
Dependencia: 2020 DIVISION DE NUEVAS CREACIONES

FORMULARIO UNICO DE SOLICITUD DE PATENTE
2000-3

1 SOLICITUD DE:

[X] Patente de Invención [ ] Patente de Modelo de Utilidad

2 SOLICITANTE (71)	Nombre:	**MICROSOFT CORPORATION**	**IDENTIFICACION**
	Dirección:	One Microsoft Way Redmond, Washington 98052, Estados Unidos de America	C.C. [ ] NIT [ ]
	Teléfono:	3123600 Fax: 3122410	C.E. [ ] Otro [X]
	E-Mail:	general@cardenasycardenas.com	
	Lugar de Constitución	Estado de Washington, Estados Unidos de América	Número
	Domicilio:	REDMOND, WASHINGTON, ESTADOS UNIDOS DE AMERICA.	
REPRESENTANTE O APODERADO (74)	Nombre:	**DARIO CARDENAS NAVAS**	**IDENTIFICACION**
	Dirección:	Carrera 7 No 71-52 Torre B Piso 9	C.C. [X] NIT [ ]
	Teléfono:	3123600 Fax: 3122410	C.E. [ ] Otro [ ]
	E-Mail:	general@cardenasycardenas.com	Número 17.066.629 BTA TP 1.250 C.S.J.
3 INVENTOR(ES) (72)	Nombre:	1. Caglar Gunyakti 2. Ning Zhang 3. Wen- Pin Scott Hsu	**IDENTIFICACION**
	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	C.C. [ ] NIT [ ]
	Teléfono:	3123600 Fax: 3123600	C.E. [ ] Otro [X]
	E-Mail:	general@cardenasycardenas.com	Número
	Domicilio:	1. WASHINGTON, ESTADOS UNIDOS DE AMERICA 2. WASHINGTON, ESTADOS UNIDOS DE AMERICA 3. WASHINGTON, ESTADOS UNIDOS DE AMERICA	

Título(54):

INTERFACE DE PROGRAMACION PARA EL LICENCIAMIENTO

6 Prioridad SI [X] NO [ ]	(33)País de Origen	(32)Fecha	(31)N° de Solicitud
	Estados Unidos de América	Octubre 24, 2003	10/692868

Para publicar a partir de la fecha de la presente solicitud a los:

[ ] 6 meses [ ] 12 meses [ ] 18 meses [ ] Otro

Calificación Internacional

G06F 17/60

AMR/JAG/aimg

Comprobante de Pago No.:	04-75,288 / 04-75,290	Fecha	Sept. 22, 2004

ANEXOS

- [X] Comprobante de pago de la tasa de presentación de la solicitud. (No 04-75,288)
- [X] Comprobante de pago de la tasa por concepto de excedente de palabras en la publicación.
- [X] Comprobante de pago de la tasa por concepto de reivindicación de prioridad. (No 04-75,290)
- [X] Poderes, si fuere el caso. (Protocolo No 16.069)
- [X] Certificado de existencia y representación legal cuando el solicitante sea persona jurídica. Protocolo No 16.069)
- [ ] Copia certificada de la primera solicitud, si se reivindica prioridad.
- [ ] Traducción simple de la primera solicitud, si se reivindica prioridad.
- [ ] Documento de cesión del inventor al solicitante o a su causante.
- [X] Descripción de la invención.
- [X] Una o más reivindicaciones.
- [X] Dibujos y/o planos necesarios.
- [ ] 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 la comunidad de las comunidades indígenas.
- [ ] De ser el caso, certificado de depósito del material biológico.
- [X] Arte final 12x12 y 6x6 cm.
- [X] Resumen y extracto para publicación.

NOMBRE: DARÍO CARDENAS NAVAS

FIRMA:

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

3

SUPERINTENDENCIA Y COMERCIO
Radicacion : 04094972 00000000 Folios: 76
Fecha (AMD): 2004-09-23 17:00:27
Tramite : 002 PATENTES D 1 REGISTRO/ 411 PRESENTA
Dependencia: 2020 DIVISION DE NUEVAS CREACIONES

NIT : 800176089-2

RECIBO OFICIAL DE CAJA : 04 - 75,298
FECHA : SEPTIEMBRE 22 DE 2004

C O N S I G N A C I O N

DEPOSITANTE	TIPO PAGO	BANCO	CUENTA	No. PAGO	FECHA PGO	Vr. PAGO
CARDENAS Y CARDENAS P.I.	CONSIGNACION	BANCO POPULAR	050-00110-6	24338231	22/09/2004	2,571,000.00

C O N C E P T O

CANT.	RENTISTICO	CONCEPTO	TOTAL CONCEPTO
1	50005-01-01 SOLICITUDES	1 TRAMITES DE SOL. DE PATENTE DE IN	422,000.00
		TOTAL : $	422,000.00

SON: CUATROCIENTOS VEINTIDOS MIL PESOS

RESPONSABLE : _____

RECIBO DE CAJA APLICADO AL EXPEDIENTE No._____

INTERFACE DE PROGRAMACION PARA EL LICENCIAMIENTO

CARDENAS & CARDENAS ABOGADOS

3

4

SUPERINTENDENCIA Y COMERCIO
Radicacion : 04094972 00000000 Folios: 76
Fecha (AMD): 2004-09-23 17:00:27
Tramite : 002 PATENTES B 1 REGISTRO/ 411 PRESENTA
Dependencia: 2020 DIVISION DE NUEVAS CREACIONES

NIT : 800176089-2

RECIBO OFICIAL DE CAJA : 04 - 75,290
FECHA : SEPTIEMBRE 22 DE 2004

***** C O N S I G N A C I O N *****

DEPOSITANTE	TIPO PAGO	BANCO	CUENTA	No. PAGO	FECHA PGO	Vr. PAGO
CARDENAS Y CARDENAS P.I.	CONSIGNACION	BANCO POPULAR	050-00110-6	24338251	22/09/2004	2,571,000.00

***** C O N C E P T O *****

CANT.	RENTISTICO	CONCEPTO	TOTAL CONCEPTO
1	50005-01-01 SOLICITUDES	92 INVOCACION DE UNA PRIORIDAD EN NU	118,000.00
		TOTAL : $	118,000.

SON: CIENTO DIEZ Y OCHO MIL PESOS

RESPONSABLE : _____

RECIBO DE CAJA APLICADO AL EXPEDIENTE No._____

INTERFACE DE PROGRAMACION PARA EL LICENCIAMIENTO

CARDENAS & CARDENAS
ABOGADOS

5

SITE PLAN

12x12

6

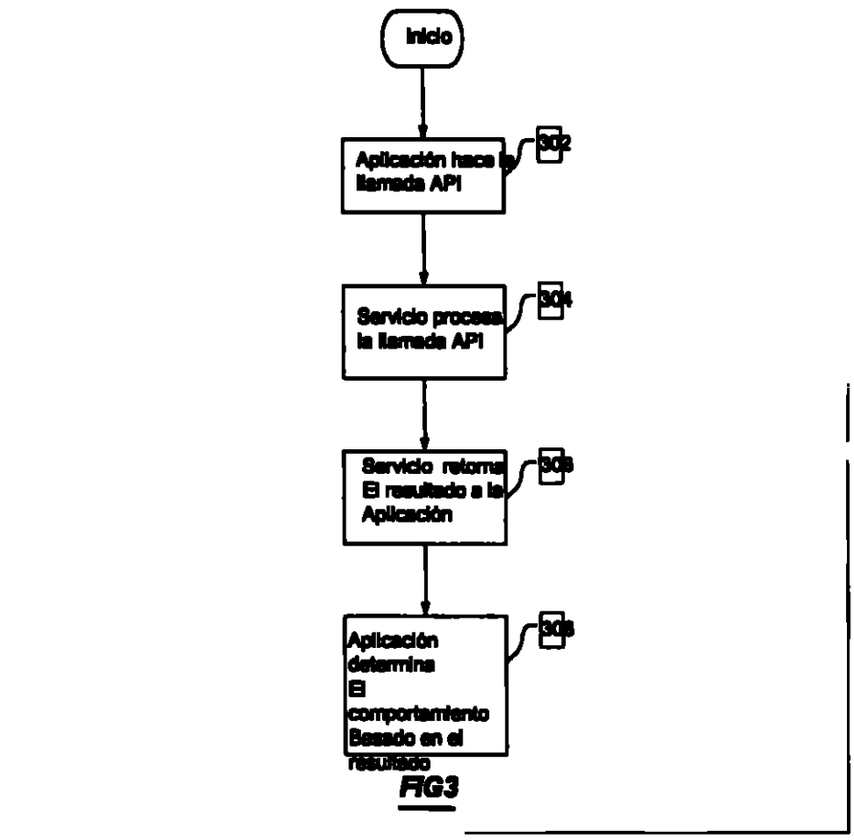
Inicio
Aplicación hace la llamada API
302
Servicio procesa la llamada API
304
Servicio retorna El resultado a la Aplicación
306
Aplicación determina El comportamiento Basado en el resultado
308

FIG3

7

INTERFASE DE PR GRAMACI N PARA EL LICENCIAMIENTO

CAMPO DE LA INVENCIÓN

[0001] la actual invención se relaciona generalmente con el campo del software de computadora, y, más particularmente, con un interfase de programación que apoya la aplicación de licencias electrónicas.

ANTECEDENTE DE LA INVENCION

[0002] El software producidos en el comercio de la tradicionalmente se ha hecho disponible bajo una licencia que define los términos permitidos del uso del software. Cuando la práctica del software licenciado comenzó, la licencia tomó generalmente la forma de un documento jurídico que definió los derechos del usuario con respecto al software. Tal documento confió en el sistema legislativo para la aplicación. Esto se ha vuelto deseable para la aplicación de las licencias electronicamente, es decir, es deseable que un programa de computadora contenga el código que desalienta activamente o previene el uso del software de una manera que sea contraria a la licencia.

[0003] La mayoría de software que preve la aplicación electrónica de la licencia proporciona su propia infraestructura para manejar el licenciamiento del software y el uso de las licencias. Así, un producto de software comercial típico puede incluir no solamente el código para realizar la función de la base del producto, pero puede también llevar con él el código para obtener, evaluar, proteger, y para manejar las licencias del software. Para que cada vendedor del software desarrolle e incorpore esta infraestructura en su software es a menudo una duplicación derrochadora de esfuerzo. Es por lo tanto deseable proporcionar un sistema que realice las funciones básicas relacionadas con el software que licencia, donde el sistema se puede utilizar por una amplia variedad de usos de software en una uniforme y definida manera.

[0004] En la vista precedente, hay una necesidad de un mecanismo que supera las desventajas del arte anterior.

El RESUMEN DE LA INVENCIÓN

[0005] La actual invención proporciona una licencia de software de la Interfase del Programa de Aplicación (API) que proporcione ciertas funciones de licencia para el uso de productos de software. Un servicio de licencia realiza funciones referentes al uso de licencias, y expone estas funciones a los productos de software con el API. El servicio realiza funciones tales como la obtención de las licencias, almacenamiento y el manejo, protegiendo la licencia de tratar de forzar, de evaluar la validez de una licencia, y de evaluar si una licencia está limitada correctamente al producto de la máquina y/o de software en el cual se utiliza. El software puede hacer uso esta funcionalidad llamando los métodos del API.

[0006] En un uso típico del API, un producto de software llama un método "abre" del API para obtener un manejo único que sea utilizado por el servicio de la licencia para identificar la aplicación. El producto de software entonces llama al método API "consumo corecto". "consumo," en este contexto, significa el ejercicio de un derecho especificada. La llamada al método "consumo corecto" es paremitrizada por el manejo del producto de software, y por el nombre del derecho a ser consumido. El servicio de la licencia entonces procura localizar una o más licencias válidas, correctamente nombradas que contienen el derecho nombrado. Si no existe ninguna licencia, entonces el producto de software es notificado de la falta. Si existen tales licencias, entonces el derecho está limitado a una de las licencias, y el producto de software llamado se notifica del enlazamiento. En tal caso, el producto de software sabe que existe el derecho, y puede realizarse cualesquiera funciones que se asocian a este derecho.

[0007] En una personalización preferida, el servicio de la licencia no define lo que puede o no puede bajo el derecho, o haga cumplir las obligaciones explícitos en el uso del software. Más allá, el servicio de la licencia maneja las licencias de una manera tal que un producto de software pueda determinarse

8

llamando el API si el derecho existe o no, de modo que el software pueda comportarse de acuerdo. Por ejemplo, un derecho se puede llamar "ejecución," indicando que el usuario tiene la derecho de ejecutar el producto de software. El producto de software puede utilizar el API para determinar si (correctamente enlazaado, y no-expirado) hay un derecho válido para ejecutar el software. Sin embargo, si la llamada del API vuelve con una indicación de que no hay derecho para ejecutar el software, es decisión del software cesar la operación o tomar otra acción basada en la no existencia de este derecho.

[0008] Una derecho puede ser asociado con la información, que llega a estar disponible después de que una llamada satisfactoria a el método "consumo correcto". Por ejemplo, un producto de software dado puede tener reglas individuales alrededor de cuando es permitido editar, imprimir, grabar, etc., y estas reglas se pueden almacenar en la licencia que contiene el derecho. El API proporciona un método "obtener información" que permite recuperar esta información de la licencia.

[0009] Otras características de la invención se describen abajo.

BREVE DESCRIPCIÓN DE LOS GRAFICAS

[0010] El resumen precedente, así como la descripción detallada siguiente de personalizaciones preferidas, se entiende mejor cuando está leída conjuntamente con los dibujos adjuntos. Con el fin de ilustrar la invención, se muestra en las construcciones ejemplares de los dibujos de la invención; sin embargo, la invención no se limita a los métodos y a los instrumentoss específicos divulgados. En los dibujos:

[0011] La fig. 1 es un diagrama de bloque de un ambiente de cómputo del ejemplo en el cual los aspectos de la invención puedan ser puestos en ejecucio'n;

[0012] La fig. 2 es un diagrama de bloque de una arquitectura en la cual un sistema realiza funciones de licenciamiento y expone un API para el uso por los productos de software;

[0013] La fig. 3 es un organigrama de un método con el cual un producto de software utiliza un API de licenciamiento;

[0014] La fig. 4 es un organigrama de un método por el cual un producto de software consume un derecho; y

[0015] La fig. 5 es un organigrama de un método por el cual un producto de software recupera la información referente a un derecho consumido.

DESCRIPCION DETALLADA DE LA INVENCIOM

Overview

[0016] el uso del software comercial es gobernado típicamente por una licencia, y ha llegado a ser cada vez más común que esta licencia sea incorporada a una forma electrónica que se puede hacer cumplir por el software sí mismo. Un desaflo de crear un sistema de licenciameinto electrónico es que una infraestructura es necesaria para manejar el uso de las licencias. Replicar esta infraestructura para cada producto de software es incómodo y derrochador. La actual invención proporciona un API que permite que diversos productos de software utilicen una infraestructura común que realiza varias funciones de licenciamiento.

ARREGLO DE COMPUTO EJEMPLAR

[0017] Fig. 1 muestar un ambiente de cómputo ejemplar en el cual los aspectos de la invención pueden ser implementados. El ambiente del sistema de cómputo 100 es solamente un ejemplo de un ambiente de cómputa conveniente y no se piensa sugerir ninguna limitación en cuanto al alcance del uso o la funcionalidad de la invención. No debería el ambiente de cómputo 100 ser interpretado como teniendo cualquier dependencia o requisito referente a cualquiera o a la combinación de componentes ilustrados en el ambiente de funcionamiento ejemplar 100.

[0018] La invención es operacional con numeroso otros fines generales o ambientes o configuraciones especiales del sistema del propósito de cómputo. Ejemplos de los sistemas de cómputo bien conocidos, ambientes, y/o las configuraciones que pueden ser conveniente para el uso con la invención incluyen, pero no se limitan a, ordenadores personales, servidores, ayudantes de mano o los dispositivos de computadora portátil, sistemas basados en multiprocesador,

sistemas basados microprocesadores, las cajas superiores, la electrónica de consumidor programable, las PC de red, los ordenadores centralizados, computadores fijos, sistemas nativos, ambientes de cómputo distribuidos que incluyen cualesquiera de los sistemas o de los dispositivos anteriormente dichos, y los similares.

[0019] La invención se puede describir en el contexto general de instrucciones ejecutables de cómputo, tales como módulos de programa, siendo ejecutado por una computadora. Generalmente, los módulos de programa incluyen las rutinas, los programas, los objetos, los componentes, las estructuras de datos, etc. que realizan tareas particulares o implementan tipos de datos abstractos particulares. La invención se puede también practicar en los ambientes cómputo distribuidos donde las tareas son realizadas por los dispositivos de procesamiento remoto que se ligan con la red de comunicaciones u otro medio de transmisión de datos. En un ambiente de cómputo distribuido (Distributed Computing Environment), los módulos del programa y otros datos se pueden situar en medios locales y remotos de la memoria interna incluyendo los dispositivos de almacenamiwento de la memoria.

[0020] Con referencia a la Fig. 1, un sistema ejemplar para poner la invención en ejecución incluye un dispositivo de cómputo para fines generales en forma de una computadora 110. Los componentes de la computadora 110 no se pueden incluir, sino limitar a, una unidad de proceso 120, una memoria de sistema 130, y un bus de sistema 121 que conformavarios componentes de ese sistema incluyendo la memoria del sistema para la unidad de proceso 120. La unidad de proceso 120 puede representar unidades de proceso lógicas múltiples tales como estas apoyadas en un procesador multi-hilos. El bus 121 del sistema puede ser cualesquiera de varios tipos de estructuras de bus incluyendo un bus de memoria o un controlador de memoria, un bus periférico, y un bus local usando cualquiera de una variedad de arquitecturas de bus. A modo de ejemplo, y no sin limitación, tales arquitecturas incluyen el autobús estándar de la arquitectura de la industria (ISA), el bus microarquitectura de canal (MCA), el bus extyendido ISA (EISA), val asociación de estándares electrónicos de video (VESA), y el bus de interconección de dispositivo periférico (PCI) (también conocido como mezzzanine (entresuelo). El autobús 121 del sistema se puede también implementar como una conexión punto a punto, fábrica de switcheo, o similares, entre los dispositivos de comunicación.

[0021] La computadora 110 incluye típicamente una variedad de medios legibles por computador. Los medios legibles por computador pueden ser cualquier medio disponible que se pueda accesar por la computadora 110 e incluye medios volátiles y permanentes, removibles y no removibles. A modo de ejemplo, y sin limitación, los medios legibles por computador puede abarcar medios de la memoria interna y medios de comunicación. Los medios de la memoria interna incluyen los medios volátiles y permanentes, removibles y no removibles implementados en cualquier método o tecnología para el almacenamiento de información tal como instrucciones legibles por computador, estructuras de datos, módulos del programa u otros datos. Los medios de la memoria interna incluyen, pero no se limitan a, RAM, ROM, EEPROM, la Memoria FLASH u otra tecnología de memoria, CDROM, discos versátiles digitales (DVD) u otro almacenamiento en discos óptico, casetes magnéticos, cinta magnética, almacenamiento en discos magnético u otros dispositivos de almacenamiento magnéticos, o cualquier otro medio que se pueda utilizar para almacenar la información deseada y que pueda ser accedido por la computadora 110. Los medios de comunicación incorporan típicamente las instrucciones legibles por computador, estructuras de datos, los módulos de programa u otros datos en señales de datos moduladas por ejemplo la onda de portador u otro mecanismo de transporte e incluyen cualquier medio de entrega de información. El término "señal modulada de datos" significa una señal en cuanto tenga una o más de sus características fijas o cambiantes de tal manera para codificar la información en la señal. A modo de ejemplo, y sin limitación, los medios de comunicación incluyen medios conectados tales como una red cableada o una conexión directa-cableada, y medios inalámbricos tales como acústico, RF, infrarrojo y otros medios inalámbricos. Las combinaciones de cualquiera de los anteriores deben también ser incluidos dentro del alcance de medios legibles por computador.

[0022] La memoria de sistema 130 incluye medios de almacenamiento en forma de memoria volátil y/o permanente tal como memoria sólo para leer (ROM) 131 y memoria de acceso aleatorio (RAM) 132. Un sistema básico 133 (BIOS) de entrada-salida, conteniendo las rutinas básicas que ayudan a transferir la información entre los elementos dentro de la computadora 110, por ejemplo durante el inicio (start-up), se almacena típicamente en ROM 131. La RAM 132 contiene típicamente los módulos de datos y/o del programa a los cuales sea inmediatamente accesible y/o actualmente siendo encendido por la unidad de proceso 120. A modo de ejemplo, y sin limitación, Fig. 1 ilustra el sistema operativo 134, los programas de aplicación 135, otros módulos del programa 136, y los datos del programa 137.

[0023] La computadora 110 puede también incluir otro medio de almacenamiento removible/no-removible, volátil/no volátil. A modo de ejemplo solamente, la Fig. 1 ilustra una unidad de disco duro 140 que lee desde o escribe en medios no removibles, permanentes, una unidad de disco magnético 151 que lee o escribe en un disco magnético removible, permanente 152, y a una unidad de disco óptico 155 que lee desde o escribe en un disco óptico removible, permanente 156, tal como una ROM de CD u otros medios ópticos. Otros medios de memoria removible/no-removible, volátil/no volátil, que se pueden utilizar en el ambiente de funcionamiento ejemplar incluyen, pero no se limitan a, los casetes de cinta magnética, las tarjetas de memoria flash, los discos versátiles digitales (DVDs), cinta video digital, RAM de estado sólido, ROM de estado sólido, y similares. La unidad de disco duro 141 está conectada típicamente con el bus 121 del sistema a través de una interfase fija de memoria tal como la interfase 140, y la unidad de disco magnético 151 y la unidad de disco óptico 155 son conectados típicamente con el bus 121 del sistema por una interfase removible de la memoria, tal como la interfase 150.

[0024] Las unidades y sus medios asociados de la memoria interna discutidos arriba e ilustrados en Fig. 1, proporcionan el almacenamiento de instrucciones legibles por computador, de estructuras de datos, de módulos de programa y de otros datos para la computadora 110. En la Fig. 1, por ejemplo, la unidad de disco duro 141 se ilustra como almacena el sistema operativo 144, los programas de aplicación 145, otros módulos 146 del programa, y los datos del programa 147. Observe que estos componentes pueden ser iguales o diferentes del sistema operativo 134, de los programas de aplicación 135, de otros módulos del programa 136, y de los datos del programa 137. El sistema operativo 144, los programas de aplicación 145, otros módulos del programa 146, y los datos del programa 147 son dados diversos números aquí para ilustrar que, en un mínimo, ellos son copias diferentes. Un usuario puede incorporar comandos y la información en los dispositivos de entrada directos de la computadora 20 tales como un teclado 162 y el dispositivo apuntador 161, designado comúnmente como ratón, un Trackball (Mouse de Esfera) o un touchpad (Mouse de cojín de Tacto). Otros dispositivos de entrada (no mostrados) pueden incluir un micrófono, una palanca de mando, un cojín de juego (gamepad), un plato de satélite, un digitalizador (scanner), o similares. Éstos y otros dispositivos de entrada están conectados a menudo con la unidad de proceso 120 a través de la interfase 160 de entrada del usuario que se junta al bus del sistema, pero se pueden conectar por otras estructuras de interfase de bus, tales como un puerto paralelo, un puerto de juego, o un bus serial universal (USB). Un tipo de monitor 191 u otro dispositivo de pantalla también está conectado con el bus 121 del sistema vía una interfase, tal como una interfase de video 190. Además del monitor, las computadoras pueden también incluir otros dispositivos de salida periféricos tales como parlantes 197 e impresora 196, que pueden ser conectados a través de una interfase periférica 195 de salida.

[0025] La computadora 110 puede funcionar en un ambiente de red usando conexiones lógicas a unas o más computadoras remotas, tales como una computadora remota 180. La computadora remota 180 puede ser el ordenador personal, el servidor, el enrutador, la PC de la red, el dispositivo par u otro nodo de red común, e incluye típicamente muchos o todos los elementos descritos arriba relacinados a la computadora 110, aunque solamente un dispositivo de almacenamiento de memoria 181 se ha ilustrado en Fig. 1. Las conexiones lógicas

representadas en la Fig. 1 incluyen una red de área local (LAN) 171 y una red de área amplia (WAN) 173, pero pueden también incluir otras redes. Tales ambientes de red son comunes en oficinas, redes de ordenadores de empresas grandes, intranets e Internet.

[0026] Cuando está utilizada en un ambiente de red LAN, la computadora 110 está conectada con la LAN 171 a través de una interfase o de un adaptador de red 170. Cuando se utiliza en un ambiente de red WAN, la computadora 110 incluye típicamente un módem 172 u otro para establecer comunicaciones sobre la WAN 173, tal como Internet. El módem 172, que puede ser interno o externo, se puede conectar con el bus 121 del sistema vía la interfase 160 de entrada de usuario, u otro mecanismo apropiado. En un ambiente interconectado, los módulos de programa representados relacionados con la computadora 110, o partes de ella, se pueden almacenar en el dispositivo de almacenamiento remoto de la memoria. A modo de ejemplo, y sin limitación, la Fig. 1 ilustra los programas de aplicación remota 185 como residente en el dispositivo de memoria 181. Es importante mostrar que las conexiones de red son ejemplos y que pueden ser utilizados otros medios de establecer un puente de comunicaciones entre las computadoras.

Servicio de Licenciamiento de Software

[0027] La Fig. 2 muestra un sistema de ejemplo que proporciona un servicio que licenciamiento del software 202. El servicio de licenciamiento de software 202 funciona dentro de la computadora 110 (mostrada en la Fig. 1). En un ejemplo, el servicio de licenciamiento de software 202 es parte de un sistema operativo que se ejecuta en la computadora 110. El servicio de licenciamiento del software mantiene un almacén de la licencia 204 en el cual los archivos de la licencia para el software se almacenen. Los archivos de la licencia pueden, por ejemplo, ser los archivos de derechos del lenguaje enriquecido (XrML – eXtensible Rights Markup Language) que especifican los derechos al software, y que también pueden especificar varios tipos de condiciones en el ejercicio de esos derechos. El software 204 también mantiene un almacén de confianza 206. El almacén de confianza 206 almacena datos dinámicos autenticables, de una manera de forzar-resistencia; el almacén de confianza 206 almacena los datos que se utilizan en el proceso de la validación de la licencia. Por ejemplo, ciertas licencias pueden tener fechas de vencimiento, y, para evitar que los datos de expiración sean evitados con la restauración no actualizada del reloj, el tiempo actual (y el tiempo transcurrido) se pueden almacenar periódicamente en el almacén de confianza 206 para asegurarse de que el reloj se está moviendo siempre hacia adelante.

[0028] El servicio de licenciamiento de software 202 maneja el almacén de licencia 204 y el almacén de confianza 206, y también realiza varias funciones relacionadas con licenciamiento de software. Por ejemplo, el servicio licenciamiento de software 202 puede contener los módulos que analizan los archivos de la licencia, los módulos que hacen cumplir la obligación de una licencia a una máquina en particular y/o a un caso particular de un producto de software, y un contador de tiempo (timer/counter) seguro (que usa el almacén de confianza 206 en la manera descrita arriba).

[0029] El servicio de licenciamiento de software 202 expone una interfase de programación de aplicación (API) 208 que permite al software de aplicación (tal como la aplicación 135) el servicio de licenciamiento de software 202. La aplicación 135 puede invocar las características del servicio de licenciamiento de software 208 haciendo la llamada local del procedimiento (LPC) a los métodos del API 208. Un sistema de ejemplo de los métodos del API que se puede exponer por el servicio de licenciamiento de software 202 se describe abajo.

[0030] La manera en qué el API 208 es utilizado por una aplicación se describe con referencia a la Fig. 3. Inicialmente, la aplicación hace una llamada del API (302). El servicio 202 procesa la llamada del API (304), y retorna el resultado de la llamada del API a la aplicación 306. Por ejemplo, una llamada del API puede solicitar ejercitar ("consuma") un derecho concedida en una licencia, o para recuperar la información de una licencia. La aplicación entonces recibe el

resultado de la llamada del API, y se determina, basado en ese resultado, qué el comportamiento del la aplicación debe ser (308). Es decir el servicio de licenciamiento de software 202, en una personalización preferida, no se obliga una licencia directamente, sino que proporciona la infraestructura a través de la cual las licencias pueden ser manejadas y utilizadas. Por ejemplo, si una aplicación hace una llamada del API para consumir una derecho y el servicio 202 determina que no hay licencia válida que conceda este derecho, el servicio 202, en una personalización preferida, no evita que la aplicación funcione, sino que por el contrario informa a la aplicación que el derecho no está disponible. Así, la aplicación puede utilizar sus propios mecanismos para determinar qué hacer en respuesta a la indisponibilidad del derecho. Esta faceta del API proporciona a vendedores del software la flexibilidad de decidir cómo la infraestructura de licenciamiento proporcionada por el servicio 202 debe ser utilizada. En otra implementación, la aplicación se puede limitar a la funcionalidad del servicio de licenciamiento

Ejemplo de Licenciamiento de Software API

[0031] El siguiente ejemplo es un conjunto de métodos de API que se pueden exponer por un servicio de licenciamiento de software:

SLOpen

[0032] La función de SLOpen abre una SL manija de contexto del cliente que se debe utilizar para todas las llamadas subsecuentes del SL API. (a través de las descripciones del API del ejemplo, el "SL" referirá al servicio de licenciamiento de software. Así, la manija del contexto del cliente SL es la manija usada por un cliente para comunicarse con el servicio de licenciamiento de software.)

HRESULT
SLOpen(
CONST GUID * pguidApp,
HSLC * phSLC
);
Parameters
pguidApp

[0033] [Entrada] El apuntadors para la apliación GUID que identifica únicamente una aplicación. Si esta argumento es NULO, se vuelve un error de E_INVALIDARG.

phSLC
[0034] [Salida] El manejo del contexto del cliente SL o INVALID_HANDLE_VALUE si falla.

Observaciones (Remarks)
[0035] Utilizan la función SLClose para cerrar una manija del contexto retornada por SLOpen.
[0036] La aplicación GUID: Identificación (ID) única del la aplicación. Para la versión de WINDOWS del conjunto de aplicación de MICROSOFT OFFICE, WinWord tiene una aplicación GUID la cual es diferente de la aplicación GUID de Excel. Para Windows, Windows en sí mismo es una aplicación, aunque esta compuesto de muchos programas.
[0037] En el caso de la oficina, el usuario puede instalar ambos, el conjunto de Office y WinWord como producto independiente. Desde el punto de vista SL, WinWord tiene la misma aplicación GUID asociada a dos productos GUID. Es decir WinWord puede utilizar la licencia de conjunto del producto o la del producto de WinWord.
[0038] Cuando SLOpen tiene éxito:
[0039] Se ha establecido un atascamiento RPC (Remote Procedure Call).
[0040] Una memoria del contexto se crea en servicio SL. El contexto se utiliza para guardar el estado de la información para el cliente solicitante.
[0041] El manejo de SLC es como una archivo de administración. Un proceso puede abrir manijas múltiples de contexto del SL pero las manijas son válidas dentro del proceso del solicitante.

Returns

[0042] Éxito o falla

SLClose

[0043] La función SLClose cierra una manija abierta del contexto de cliente SL. Cualquier información en el contexto se libera automáticamente.

HRESULT

SLClose(

HSLC) *hSLC*

);

Parámetros

hSLC

[0044] [Entrada] La administración al cliente actual SL que maneja el contexto

Observaciones

[0045] Cuando se hace SLClose, el enlace del RPC ese liberado, y se destruye el contexto.

Returns (Retorno)

[0046] Éxito o fallta.

SLInstall

[0047] La función de SLInstall instala las licencias de aplicación y registra la información de las aplicaciones.

HRESULT SLInstall(

HSLC *Hslc,*

(CONST SL_PRODKEY * *pAppPrdKey,*

DWORD *dwNumOfApps,*

CONST GUID * pguidApps,

DWORD *dwNumOfLicFiles*

PCWSTR *ppszLicFiles[ ]*

BOOL *bVerify*

);

Parameters

hSLC

[0048] [in - Entrada] Administración del actual manejo del contexto del cliente SL

pAppPrdKey

[0049] [in - Entrada] Estructura de clave del producto de la aplicación. La

clave del producto puede ser de formato clave del producto de Microsoft o del formato clave del producto de la aplicaión.

typedef struct _ tag**SL_PRODKEY**

{

DWORD cbSize; //tamaño del SL_PRODKEY

Structure //(Estructura)

DWORD dw Versión; //versión de la estructura SL_PRODKEY

WCHAR szProdKey[**MAX_PRODKEYSTR_SIZE+1**];

SL_PRODKEY_TYPE eProdKeyType; //tipo de clave del Producto

SL_CUSTOM_PRODKEY_INFO CustomPrdKeyInfo; //tipo de clave del Producto

} SL_PRODKEY;

El eProdKeyType puede ser uno de valores siguientes:

SL_PRODKEY_CUSTOM

SL_PRODKEY_MS2002

SL_PRODKEY_MS2003

[0050] Si el tipo de clave de producto no es un producto calve MS (MS Product Key) (es decir eProdKeyType == SL_PRODKEY_CUSTOM), el solicitante tiene que completar su información propia de la clave del producto. Si el producto utiliza la clave de producto MS (MS Product Key), entonces la información del producto clave personal (el CustomPrdKeyInfo) puede ser no ignorada.

typedef del struct _ tag**SL_CUSTOM_PRODKEY_INFO**

DWORD dwSKUID //identificación única para SKU específico, por ejemplo identificación de grupo en MS PID.

DWORD dwSerialNumber; //número de serie único, ej. número del canal + de serie en el MS PID.

} SL_CUSTOM_PRODKEY_INFO;

[0051] El número de versión actual de SL_PRODKEY es 1. El solicitante puede utilizar SL_CURRENT_PRODKEY_VERSION en el campo del dwVersion.

dwNumOfApps

[0052] [in - entrada] El número de la aplicación GUID en pguidApps.

pguidApps

[0053] [in - entrada] Una lista de aplicación de GUID. La aplicación representa la aplicación de para que siendo instalada la licencia. Por ejemplo, el programa de inicio de Office puede llamar esta función para instalar las licencias de Word, Excell especificando cada aplicación GUID en pguidApps. pguidApp no puede ser NULO aquí.

dwNumOfApps

[0054] [in - entrada] Número de archivos de licencia archiva.

ppszLicFile

[0055] [in – entrada] Nombres de archivo en arreglos de cadenas (array of strings).

Returns

[0056] Éxito o falla

SLUninstall

[0057] La función SLUninstall desinstala una licencia de producto de una aplicación

HRESULT SLUninstall (

HSLC *hSLC,*

CONST SL_PRODKEY * *pAppPrdKey*

);

Parameters (Parámetros)

hSLC

[0058] [in - entrada] Administración del actual manejo del contexto del cliente SL

pAppPrdKey

[0059] [in - entrada] Vea arriba la defininición en la conexión con SLInstall.

Observaciones

[0060] Una aplicación podría tener licencias de más de un producto. Por ejemplo, cuando la un usuario desinstala la suit de Office (Conjunto de Programas de Office), la asociación entre la licencia de la suit de Office y WinWord deben ser removidos, pero la licencia de WinWord independiente no debe ser removida.

[0061] Cuando tiene éxito SLUninstall:

[0062] La información acociada con la clave del producto es removida. (véase SLInstall, arriba, para la información asociada).

[0063] La asociación de claves de producto con la aplicación GUID es retirada.

[0064] Los archivos de la licencia asociados al producto GUID preferiblemente todavía se guardan.

Returns

[0065] Éxito o falla

SLConsumeRight

[0066] La función de derecho de consumo SLConsumeRight deja a una aplicación examinar o ejercitar los derechos en una licencia local-almacenada. Llamar esta función ata una licencia al derecho mencionado en pszRightName. Si este derecho no se puede ejercitar por el solicitante actual, entonces la aplicación falla. Si la función tiene éxito, la acción

asociada al derecho puede ser ejecutado (como dsiminuyendo un contador de uso que disminuye, la cuenta de tiempo, o nada)

HRESULT SLConsumeRight(

HSLC *hSLC,*

PCWSTR *pszRightName,*

SL_ASYNC_CONTEXT * pAsyncContext

);

Parameters

hSLC

[0067] [in - entrada] Administración del actual manejo del contexto del cliente SL

pszRightName

[0068] [In - Entrada] El nombre de los derechos necesita ser evaluado. En el diseño actual, el nombre de derecho es definido por las aplicaciones. El SL abre la licencia y evalúa la condición basada en el nombre de derecho.

pAsyncContext

[0069] [in/out – entrada/salida] Si el *pAsyncContext* es NULO, esta función trabaja en modo síncrono, sino, la función trabaja en modo asincrónico. SL_ASYNC_CONTEXT es transparente al solicitante y es manejado por SLC.

Observaciones

[0070] Todas las licencias asociadas con la aplicación GUID (especificado en SLOpen) serán combinadas conceptualmente en una licencia lógica.

[0071] Si son garantías consumibles múltiples del derecho, entonces la licencia con una prioridad más alta será consumida primero.

Returns - Retorna

[0072] Éxito o falla

SLInitializeAsyncContext

[0073] La función SLInitializeAsyncContext inicializa el contexto asincrónico para que las funciones de SLC hagan llamada asincrónica.

HRESULT SLInitializeAsyncContext(

SL_ASYNC_CONTEXT * *pAsyncContext* // Contexto asincrónico

HANDLE *hEvent,* // Manejo de Evento

PVOID *pvReserved* // reservado, NULO

);

Parameters - Parámetros

pAsyncContext

[0074] [in/out – entrada/salida] Apuntador al contexto asincrónico que contiene la información asincrónica de la llamada.

hEvent

[0075] [In - Entrada] El objeto del evento usado para la sincronización.

pvReserved

[0076] [in -entrada] Reservado para la extensión.

Returns - Retorna

[0077] Éxito o falla.

SLCancelAsyncCall

[0078] La función de SLCancelAsyncCall se utiliza para cancelar una llamada asincrónica.

HRESULT SLCancelAsyncCall(

SL_ASYNC_CONTEXT * *pAsyncContext,* // Contexto asincrónico

BOOL *fAbortCall* // cancela inmediatamente

);

Parameters - Parámetros

pAsyncContext

[0079] [In -entrada] Contexto asincrónico para la llamada asincrónica del SL.

fAbortCall

[0080] [In –Entrada] Si es VERDADERO, la llamada es cancelada inmediatamente. Si es FALSO, espera el SL para terminar la llamada.

Observaciones

[0081] Hay dos maneras para que el solitante requiera la cancelación de una llamada-abortiva y nonabortiva asincrónica. En una cancelación abortiva (fAbortCall es VERDADERO), la función de SLCancelAsyncCall envía una notificación de cancelación al SLC y la llamada asincrónica es cancelada inmediatamente, sin esperar una respuesta del SLC. En una cancelación no-abortiva (fAbortCall es FALSO) la función de SLCancelAsyncCall notifica al SLC de la cancelación y el solicitante espera SLC para terminar la llamada.

Returns - Retorna

[0082] Éxito o falla

SLCompleteAsyncCall

[0083] La función SLCompleteAsyncCall se utiliza para terminar una llamada asincrónica de SLC.

HRESULT SLCompleteAsyncCall(

SL_ASYNC_CONTEXT * *pAsyncContext* // Contexto

asincrónico

HRESULT * *phrAsyncCall* // código de error de la llamada asincrónica sometida

};

Parameters - Parámetros

pAsyncContext

[0084] [In – entrada] Contexto asincrónico para la llamada asincrónica del SL.

phrAsyncCall

[0085] [Salida] El código de error de la llamada asincrónica sometida.

Observaciones

[0086] Si el solicitante llama esta función antes de que haya llegado la respuesta, la llamada retorna **E_SLC_ASYNC_CALL_PENDING**. El almacenador intermediario debe ser válido y debe ser bastante grande recibir el valor de vuelta. Si la llamada no vuelve **E_SLC_ASYNC_CALL_PENDING**, esta invocación de **SLCompleteAsyncCall** es final para la llamada asincrónica. Después de esta llamada de función, sin importar éxito o falla, todos los recursos asignados para esta llamada asincrónica se liberan. (las llamadas subsecuentes a las funciones de SLCompleteAsyncCall o de SLCancelAsyncCall teiene resultados indefinidos hasta que una nueva llamada sobre la estructura de SL_ASYNC_CONTEXT sea iniciada).

Returns – Retorna

Valor	**Significado**
S_OK	La llamada fue terminado con éxito
E_SLC_INVALID_ASYNC_CONTEXT	El contexto asincrónico de la llamada No es inválido
E_SLC_ASYNC_CALL_PENDING	La llamada todavía no ha terminado
E_SLC_CALL_CANCELLED	La llamada fue cancelada

SLGetInformation

[0087] La función de SLGetLicenseInfo se utiliza para conseguir una variedad de información.

HSLC de HRESULT SLGetInfomation(

HSLC *HSLC,* *// Manejo del contexto del cliente SL*

DWORD *dwCategory;* // La categoría de la información a recuperar

PCWSTR *pszKeyName,* // Nombre de la clave

DWORD* *pdwType,* //.Tipo de valor

SIZE_T* *pcbValue*, ,// Tamaño del valor

PBYTE * *ppbValue*, // Indicador del almacenador intermedio del valor

);

Parameters – Parámetros

hSLC

[0088] [In - Entrada] Administración del actual manejo del contexto del cliente SL

dwCategory

[0089] [In - Entrada] La categoría de la información.

Categoría	**Significado**
SL_CAT_RIGHTDATA	Obtiene la información del derecha atado. La licencia tiene que ser consumida con éxito antes de conseguir estos datos de derecho.
SL_CAT_DYNAMICPROPERTY	Obtiene la información que no está en la licencia pero calculado en el tiempo de ejecución. Por ejemplo, (Período de días de Gracia Restantes - RemainingGracePeriodDays. El derecho tiene que ser consumido antes de la llamada.
	Name
	Significado
	RemainingGracePeriodDays: DWORD
	El período de gracia se define en una licencia fuera-de-caja (out-of-box). Una vez que la aplicación es instalada, el tiempo está contando hacia tras. Las aplicacione pueden comprobar períodos de gracia restantes después de que hayan consumido la licencia.
	ActivationStatus: DWORD
	Después de que las aplicaciones consumieron la licencia, puede conseguir el tipo de licencia consumido. El valor de retorno podría ser:
	SL_LIC_OOB
	La licencia consumida es licencia fuera-de-caja (out-of-box).
	SL_LIC_ACQUIRED
	La licencia consumida es licencia adquirida.

	SL_LIC_NONE No hay licencia disponible.
SL_CAT_SERVICEINFO	Obtiene la información que no depende de la licencia. El solicitante puede conseguir esta categoría de a información sin consumir la. **Nombre** **Significado** SLVersion: DWORD La versión del formato SL.1.2.3.4. HWID:BINARY HWID Actual
SL_CAT_WINDOWSINFO	Obtiene la información que es una característica del derecha atado en la licencia de Windows. Esto es para compensación. La licencia de Windows ha sido consumida por el servicio SL y el servicio SL guarda las características de derechos atados.
SL_CAT_ENUMLICINFO	Cuando el SLEnumLicenseis es llamado y tiene éxito, el solicitante puede preguntar la información de la licencia enumerada usando esta categoría.

pszKeyName
[0090] [in - Entrada] El nombre de la Clave. E.g. BuildNumber (Construye Número)

pdwType
[0091] [out - salida] Tipo de Datos

Value	**Meaning**
SL_DATATYPE_SZ	Cadena única
SL_DATATYPE_DWORD	DWORD
SL_DATATYPE_BINARY	Binary (Binario)

pcbValue
[0092] [out - salida] Tamaño de la memoria intermedia asignada (en bytes).
ppbValue
[0093] [out - salida] Si tiene éxito, el dato es retornado en la memoria intermedia asignada por el SLC. El solicitante no tiene que llamar a SLFreeMemory para liberar memoria.

Returns - Retorna
[0094] Éxito o Falla

SLAcquireLicense

[0095] La función de SLAcquireLicense se utiliza para adquirir la licencia en línea para el usuario. SLC enumera las claves del producto asociadas a la aplicació y toma la clave del producto con la prioridad más alta del producto (véase SLInstall, información del registro). Después el SL consigue el URL de la cámara de compensación de licencia fuera-de-caja (out-of-box) y conecta con la cámara de compensación para conseguir una licencia.

[0096] SLAcquireLicense podría ser un proceso muy largo. Las aplicaciones pueden llamar esta función en modo asincrónico especificando el pAsyncContext (NULO significa modo síncrono).

HRESULT SLAcquireLicense(
HSLC *hSLC,* // Manejo del contexto cliente SL
PCWSTR *pszProdKeyHash* // Clave del producto desmunaza

PCWSTR *pszPublishLicense* // Cadena de la licencia de publicación

SL_ASYNC_CONTEXT * *pAsyncContext*

);

Parameters – Parámetros

hSLC

[0097] [In - entrada] Administración del actual manejo de contexto del cliente SL

pszProdKeyHash

[0098] [In - entrada] Cadena de la clave del producto separada. La separación de la clave es creada cuando SLInstall es llamado y mantenido por el servicio de licencia.

PszPublishingLicense

[0099] [In - Entrada] Secuencia de la licencia de publicación.

pAsyncContext

[0100] [In - Entrada] Contexto asincrónico para la llamada asincrónica del SL.

Observaciones

[0101] La licencia adquirida será almacenada en almacén de la licencias por consiguiente y la información de la licencia será registrada, también (véase SLInstall)

[0102] Las aplicaciones podrían necesitar agregar más información del cliente a la licencia. La aplicación puede poner la información al pbAppData en la llamada y estos datos serán enviados a la cámara de compensación.

[0103] Cuando esta función tiene éxito:

[0104] Envió de la información obligatoria necesaria al servidor especificado de la licencia.

[0105] Recibe la licencia del servidor de licencia.

[0106] Almacenó la licencia en almacén de la licencia. Vea la descripción de SLInstall, arriba, de cómo se almacena el archivo de la licencia.

Returns - Retorna

[0107] Éxito o falla.

SLGenerateTextChallenge

[0108] Genera e instala el texto requerido para que se encamine a un emisor de la licencia fuera de la forma de banda (teléfono, email, archivo compartido, etc).

HRESULT SLGenerateTextChallenge(
HSLC *hSLC,* // Manejo del contexto cliente SL
PCWSTR *pszProdKeyHash* // Clave del producto detallada

BOOL *fSingleSession,* // Sesión sencilla solamente?

PWSTR **ppszChallenge* // Apuntador de memoria intermedia para contener el texto exigido
);

Parameters – Parámetros

hSLC

[0001] [in - entrada] Administración del actual manejo de contexto del cliente SL

pszProdKeyHash

[0002] [in - entrada] Cadena de la clave del producto detallada. La clave del producto detallada es creada cuando SLInstall es llamada y mantenida por servicio de licencia.

bSingleSession

[0003] [in - entrada] Especifica si la respuesta de texto correspondiente del elemnto de la licencia será válido o no solo para el tiempo de vida del manejo de la sesión SLC.

ppszChallenge

[0004] [out - salida] Si tiene éxito, el texto exigido es retornado en la memoria intermedia asignada por el SLC. El solicitante necesita llamar a SLFreeMemory para liberar la memoria asignada.

Returns - Retorna

[0005] Éxito o Falla.

SLDepositTextResponse

[0114] Deposita la respuesta a un texto exigido de la instalación en el sistema que licencia. Utilizado para activar una licencia con un código de acceso condicional. Solamente válido si hay una exigencia excepcional del texto que se ha publicado para una licencia. Si la licencia original especificaba que el desafío era solamente válido para una sola sesión, este API se debe llamar con la respuesta antes de que el manejo de SLC sea cerrado o la entrega de la respuesta fallará.

HRESULT SLDepositTextResponse(

HSLC *hSLC* // Manejo del contexto del cliente SL

PCWSTR *pszProdKeyHas,*

PCWSTR *pszResponse* // Memoria Intermedia que contiene el texto de respuesta
);

[0115] [in - entrada] Administración del actual manejo de contexto del cliente SL

pszProdKeyHash

[0116] Cadena de la clave del producto detallada.

pszChallenge

[0117] [in - entrada]

Returns – Retorna

[0118] Éxito o Falla.

SLEnumLicense

[0006] El SLEnumLicensefunction se utiliza para enumerar licencias instaladas y para conseguir la información de la licencia.

HRESULT SLEnumLicense(
HSLC *hSLC,* // Manejo del contexto del cliente SL
CONST GUID* *pguidApp,* // Aplicación GUID
DWORD *dwIndex* // Número Índice
);

Parameters - Parámetros
hSLC

[0120] [in - entrada] Administración del actual manejo de contexto del cliente SL

pguidApp

[0121] [in - entrada] Vea SLInstall. Si el pguidApp no es NULO, entonces las licencias asociadas con este GUID se enumeran. Si el GUID es NULO, entonces todas las licencias se enumeran.

dwIndex

[0122] [in - entrada] El número de índice de la licencia a ser recuperado. Este valor debe ser cero para la primera llamada a la función de SLEnumLicense y después incrementada para las llamadas subsecuentes.

[0123] La función puede retornar licencias en cualquier orden.

Returns - Retorna

[0124] **E_SL_N _M RE_DATA** - Ninguna licencia en el índice especificado.

Observaciones

[0125] Si SLEnumLicenses tuvo éxito, la información seleccionada de la licencia puede ser alcanzada llamando SLGetInformation y la categoría es SL_CAT_ENUMLICINFO

Ejemplo:

```
DWORD i=0;

PBYTE PbProductPriority = NULL; // NULO
PBYTE PbRemainingGracePeriodDays = NULL; // NULO

For (i=0; ; i++)
{
EXIT_ON_ERROR(SLEnumLicense(hSLC, NULL, dwIndex));

if (E_SL_NO_MORE_DATA==SCODE(hr))

hr = S_OK;

brake;

}

EXIT_ON_ERROR(SLGetInformation(hSLC, SL_CAT_ENUMLIC,
"ProductPriority", &dwType, &cbProductPriority, &pbProductPriority));

   EXIT_ON_ERROR(SLGetInformation(hSLC, SL_CAT_ENUMLIC,
"RemainingGracePeriodDays", &dwType, &cbRemainingGracePeriodDays,
&pbRemainingGracePeriodDays));

Exit:
   SLFreeMemory(pbProductPriority);
   SLFreeMemory(pbRemainingGracePeriodDays);

}
```

SLFreeMemory

[0007] La función de SLFreeMemory se utiliza para liberar la memoria asignada por SLC.

```
VOID SLFreeMemory(
   PVOID          pvMemblock,      // Apuntador a la memoria
);
```

Parameters - Parámetros

pvMemBlock

[0008] [in - entrada] Bloque de memoria previamente asignado para ser liberado.

Returns - Retorna

[0009] Ninguna

El uso del Licenciamiento de Software API para controlar el uso del software

[0129] Un producto de software utiliza el API de la actual invención para varios propósitos relacionados con licenciamiento, incluyendo la consumición de los derechos en una licencia, y la recuperación de datos de la licencia. Según lo observado arriba, el API permite que el software determine los derechas que están presentes en la licencia, pero, preferiblemente, está el software para determinar qué hacer con esa información - e.g., la concesión o negación del acceso a una

característica, cesa la operación en conjunto, etc. La descripción siguiente de las figs. 4 y 5 demuestran cómo el API de la actual invención es utilizado por un producto de software.

[0130] La fig. 4 muestra un proceso de ejemplo por el cual una aplicación "consume" un derecho. La aplicación llama el método SLConsumeRight (402). Según lo discutido arriba, los argumentos de la función SLConsumeRight incluyen el manejo del cliente asignado por el servicio de licenciameinto, y el nombre del derecha (que es asignado por el vendedor del software al cual pertenece el derecha). El servicio de licenciamiento (servicio 202, mostrado en la fig. 2) recibe la llamada (404). El servicio entonces localiza las licencias que contienen el derecho, y comprueba los enlaces y la validez de las licencias. Según lo observado arriba, la licencia está situada en el almacén de licencia; si hay más de una licencia que pertenece al software de aplciación al cual pertenece la llamada SLConsumeRight, entonces una regla de prioridad se puede utilizar para seleccionar una de las licencias aplicables. La comprobación de los enlaces significa que: (1) la licencia está limitada a la clave del producto de la aplicación identificada por el manejo del cliente; y (2) la licencia está limitada a la máquina en la cual el software está funcionando (o al grupo máquinas de las cuales la máquina actual es un miembro). La comprobación de validez puede incluir la determinación de que no ha expirado el derecho (en la caja de licencias que especifican una fecha de vencimiento), y de que el número máximo de derecho de las aplicaciones no está excedido (en el caso donde la licencia especifica un número máximo de veces que se puede utilizar el derecho (es decir, "consumido")).

[0131] Si se encuentran la licencia y/o el derecho de ser limitados correctamente y válidos (408), después la licencia está limitada al derecho solicitado en la llamada del API (412). (debe ser observado que "atar" una licencia a una máquina, al ambiente, y a una clave del producto significa que la licencia especifica con qué calve de la máquina(s) y del producto puede ser utilizada; "atar" una licencia a un derecho significa que la función de consumo ha sido acertada, y el derecho de una licencia particular se está consumiendo. A través de esta descripción, estará claro el contexto de que el significado de "atar" aplica.) La llamada del API después retorna a la aplicación que llama e indica que la llamada era acertada (414). Si se ha encontrado la licencia y/o el derecho para ser inválidos, o no correctamente atado a la máquina, al ambiente, o a la identificación del producto, entonces el retorno de la llamada de SLConsumeRight a la aplicación que llama e indican que la operación falló (410).

[0132] Si la llamada de SLConsumeRight retorna con una falla, después el derecho especificado en la llamada de una licencia no se puede consumir, y no hay información sobre ese derecho disponible para la aplicación que llama. Sin embargo, si el derecho se consume con éxito, después la aplicación puede utilizar el enlace del derecho de la licencia para conseguir la información de la licencia sobre el derecho. Por ejemplo, una licencia puede contener un derecho general llamado "funcionamiento," que indica que la aplicación puede ser ejecutada. Sin embargo, por derecho de "funcionamiento", la licencia puede contener parámetros más específicos sobre el uso la aplicación - e.g., la licencia puede especificar si las características particulares de una aplicación (e.g., la impresión, edición, grabado, etc.) deberían estar encendidas o apagadas, y puede dar los parámetros específicos para el uso de estas características (e.g., el documento se puede grabar solamente en las máquinas que están funcionando en un dominio particular, o la característica de la impresión se puede utilizar solamente para treinta días, etc.). El API SL no requiere ningún tipo particular de información a ser asociado a un derecho, sino que proporciona algún mecanismo por el que un vendedor de aplicación puede asociar cualquier tipo de información a un derecho, que se puede después recuperar e interpretar por la aplicación.

[0133] Si se asume que una derecha se ha consumido con éxito según lo descrito en fig. 4, la aplicación puede entonces recuperar la información asociada al derecho. El proceso de recuperar esta información se describe en la fig. 5.

[0134] Primero, la aplicación llama el método SLGetInformation en el derecho enlazado (502). Varios tipos de información que pueden ser recuperados se describen arriba en la conexión con la descripción del método SLGetInformation. El servicio de licenciamiento entonces recibe la llamada (504). El servicio recupera la información solicitada del archivo de la licencia que contiene la derecha encuadernada (506). El servicio de licenciamiento entonces pone esta información en una memoria intermedia (508), y retorna a la apiciación que llama (510). La aplicación que llama entonces lee el contenido de la memoria intermeida, y realiza cualesquiera acciones si se juzga necesario basado en la información recuperada.

[0135] Debe ser observado que el servicio de licenciamiento puede no estar enterado del significado de la información que está dirigiendo como parte de una llamada de SLGetInformation. Según lo discutido arriba, el sistema de licenciamiento proporciona un mecanismo por el cual un vendedor de software puede crear los derechos, y puede asociar la información a los derechos. La invención no se limita tampoco a ningún tipo particular de información que se puede asociar al derecha. Cuando la información se recupera de la licencia, es pasada simplemente por el servicio de licenciamiento a la aplicación en una memoria intermedia (Buffer). La aplicación entonces interpreta la información recuperada, decide qué acciones se tomarán basadas en esa información, y utiliza sus propias características de seguridad para hacer cumplir la decisión de la aplicación. (e.g., si, basado en la información recuperada, la aplicación decide inhabilitar la característica de impresión, la aplicación contiene el código que inhabilita realmente esta característica y, posiblemente, el código que evita que un hacker trate de forzar inhabilitar la característica de impresión.)

[0136] Se observa que los ejemplos precedentes se han proporcionado simplemente con el fin de explicación y no es de manera alguna para ser interpretado como limitación de la actual invención. Mientras que la invención se ha descrito referente a varias personalizaciones, se entiende que las palabras que se han utilizado aquí son palabras de descripción y de ilustración, más bien que palabras de limitación. Además, aunque la invención que se ha descrito aquí referente a medios, a materiales y a personalizaciones particulares, la invención no se piensa para ser limitada a los detalles divulgados aquí; más aún, la invención se extiende a todas las estructuras funcionalmente equivalentes, métodos y aplicaciones, por ejemplo están dentro del alcance de las demandas añadidas. Los expertos en la materia, teniendo la ventaja de las enseñanzas de esta especificación, pueden efectuar modificaciones numerosas y los cambios pueden ser realizados sin salir del alcance y del espíritu de la invención en sus aspectos.

Qué es Demandado: → Cap. Dewd

1. Un sistema para apoyar la obligación de una licencia para un programa de computadora, el sistema comprende:

un componente de licenciamiento que mantiene un almacén de licencia en el cual se almacena la licencia, la licencia que comprende un derecho en el software y un sistema de datos se asocian al derech dicho, el componente de licenciamiento que exponía una interfaz accesible al programa de computadora, dicha llamada interfaz comprende:

un método de derecho de consumo que recibe un identificador de dicho derecho del programa de computadora y se determina si el derecho puede ser ejercitado; y un método de recuperación de información que recibe un identificador de dicho derecho del programa de computadora y proporciona datos al sistema dicho, o información basada en el sistema dicho de datos, al programa de computadora.

2. El sistema de la demanda 1, en donde el componente de licencia dicho es usable por una pluralidad de programas de computadora, el programa de

computadora que es incluido entre la pluralidad dicha de programas de computadora, en donde dicha llamada interfase abarca:

Un método de apertura de manejo que proporciona al programa de computadora; en donde el método de consumo de derechos recibe el manejo del programa de computadora y utiliza el manejo para identificar el programa de computadora del cual una llamada al método de consumo de derechos se recibe.

3. El sistema de la demanda 1, en donde la licencia es una de una pluralidad de licencias que se almacenen en dicho almacén de licencia, y en donde el método del derecho de consumo hace que el componente de licenciamiento seleccione la licencia basada en unos o más factores que comprende:

si el almacén de la licencia está asociado al programa de computadora; y una regla del conflicto que determina qué licencia seleccionar entre una pluralidad de licencias que se asocian al programa de computadora.

4. El sistema de demanda 1, en donde dicha llamda interfase comprende: un método iniciador de contexto asincrónico que establece un contexto asincrónico que procesa y proporciona un identificador de contexto de dicho programa de computadora; en donde el método dicho del consumo de derecho recibe el identificador del contexto dicho del programa de computadora dicho y procesa una petición del consumo de derecho asincrónicamente en respuesta al recibo del identificador del contexto dicho.

5. El sistema de demanda 1, en donde el método de los derechos de consumo determinan si el derecho puede ser ejercido basado en si el derecho está identificado en la licencia.

6. El sistema de demanda 1, en donde el programa de computadora y el componente de licenciamiento se ejecutan en una máquina, y en donde el método del derecho de consuo determina si el derecho puede ser ejercido basado en si la licencia está limitada a dicha máquina.

7. El sistema de demanda 1, en donde el programa de computadora se asocia a un identificador del producto, y en donde el método del derecho de consumo determina si el derecho puede ser ejercido basado en si la licencia está limitada a dicha máquina o a una clase de máquinas de las cuales dicha máquina es un miembro.

8. Un método de restringir el uso de un programa de computadora asociado con una licencia, la licencia que especifica un derecha en el programa de computadora, el método comprende:

invocando un servicio de licenciamiento haciendo una primera llamada a un primer método de interfase del servicio de licencia mencionado, dicha primera llamada siendo parametrizada por un identificador asociado con el derecho mencionado;

en respuesta a la dicha primera llamada que recibe una indicación como si el derecho es posible; y

ajustando a un primer comportamiento o a un segundo comportamiento según la indicación.

9. El método de la demanda 8, en donde el primer comportamiento dicho abarca permitir que el programa de computadora se ejecute, y en donde el segundo comportamiento dicho abarca la interrupción de la ejecución del programa de computadora.

10. El método de la demanda 8, en donde el primer comportamiento dicho abarca permitir al programa de computadora realizar un primer sistema de funciones, y en donde el segundo comportamiento dicho abarca permitir que el programa de computadora realice un segundo sistema de funciones que no sea idéntico al primer sistema dicho de funciones.

11. El método de la demanda 8, en donde el derecha se asocia a un sistema de datos, en donde el método además abarca:

haciendo una segunda llamada a un segundo método de la interfase dicho, el segundo método dicho siendo parametrizado por una indicación del derecho; y

en respuesta a la segunda llamada dicha, recibiendo el sistema dicho sistema de datos.

12. El método de la demanda 11, además comprende:

la dirección de la operación del programa de computadora basado en el mencionado sistema de datos.

13. El método de la demanda 8, además comprende:

haciendo una segunda llamada a un segundo método de la mencionada interfase; y

en respuesta a la segunda llamada dicha, recibiendo un manejo;

en donde la segunda llamada dicha se hace antes de la primera llamada dicha, y en donde la primera llamada dicha es además parametrizada por el mencionado manejo.

14. El método de la demanda 8, además comprende:

haciendo una segunda llamada a un segundo método de la mencionada interfase; y

en respuesta a la segunda llamada dicha, recibiendo un contexto asincrónico; en donde la segunda llamada dicha se hace antes de la primera llamada dicha, en donde la primera llamada dicha es además parametrizada por el contexto asincrónico dicho, y en donde el programa de computadora realiza por lo menos una acción mientras que la primera llamada es manejada asincrónicamente.

15. El método de la demanda 8, en donde el primer método dicho determina si el derecho es ejercible basado en uno o más factores que comprenden:

si la licencia está limitada a una máquina o a un ambiente en los cuales el programa de computadora se esté ejecutando;

si la licencia o el derecho está limitado a un identificador del producto asociado al programa de computadora;

si ha expirado la licencia o el derecha; y

si el derecho se ha consumido un número de veces en exceso del derecho especificado en la licencia.

16. El medio legible por computador que se codifica sobre las instrucciones ejecutables de la computadora para realizar que un método permita la aplicación de una licencia a un programa de computadora, el método comprende:

recepción de una primera llamada del método del programa de computadora, la primera llamada del método que identifica un derecho en el programa de computadora;

determinando que el derecha está contenida en la licencia y es ejercible; y volviendo al programa de computadora una indicación de que el derecha es ejercible.

17. El medio legible por computador de la demanda 16, en donde la indicación abarca un enlace del derecho a la licencia.

18. El medio legible por computador de la demanda 16, en donde el acto de determinación dicho se basa encendido si la derecha está especificada en la licencia.

19. El medio legible por computador de la demanda 16, en donde el acto de determinación dicho basado en si la licencia está limitada a una máquina en la cual el programa de computadora se esté ejecutando.

20. El medio legible por computador de la demanda 16, en donde el acto de determinación dicho basado en si la licencia o el derecha está limitado al programa de computadora.

21. El medio legible por computador de la demanda 16, en donde el acto de determinación dicho basado en si no está expirada la licencia o el derecho.

22. El medio legible por computador de la demanda 16, en donde el acto de determinación dicho basado en si la licencia se ha consumido un número de veces que excede un límite.

23. El medio legible por computador de la demanda 16, en donde el método además comprende:

recibiendo una segunda llamada del método del programa de computadora; y en respuesta a la segunda llamada del método, retornando un manejo al programa de computadora que identifica el programa de computadora;

en donde la primera llamada dicha del método es subsecuentemente realizada a la segunda llamada dicha del método, y en donde la primera llamada dicha del método además identifica dicho manejo.

24. El medio legible por computador de la demanda 16, en donde el método además comprende:

recibiendo una segunda llamada del método del programa de computadora; en respuesta a la segunda llamada del método, retornando un contexto asincrónico al programa de computadora, en donde la primera llamada del método es subsecuentemente ejecutada a la segunda llamada del método e identifica el contexto asincrónico dicho; y

ejecutando la primera llamada del método asincrónicamente mientras que el programa de computadora realiza una acción.

25. El medio legible por computador de la demanda 16, en donde el derecho se asocia a un sistema de datos, y en donde el método además comprende:

recibiendo una segunda llamada del método que indica el derecho; y

en respuesta a la segunda llamada dicha del método, proporcionando el sistema de datos al programa de computadora.

RESUMEN DEL DESCUBRIMIENTO

Un licenciameinto de software API (Application Program Interface – Interfase del Programa de Aplicación) que permite que los productos de software utilicen la funcionalidad de la administración de licencia de un servicio común. Una licencia especifica los derechos en un producto de software. El producto de software llama un método de consumo en el API para consumir un derecho. Si existe el derecho, el servicio enlaza el derecho a la licencia en la cual se encuentra el derecho. El producto de software hace cumplir los términos de la licencia concediendo, o negando, el acceso a alguna o todas las características dependiendo de si uan instancia válida del derecho es encontarda. Los datos arbitrarios se pueden asociar a un derecho. El API incluye un método para recuperar datos de un derecho que ha sido limitado previamente por el método de consumo.

Ambiente de Cómputo 100

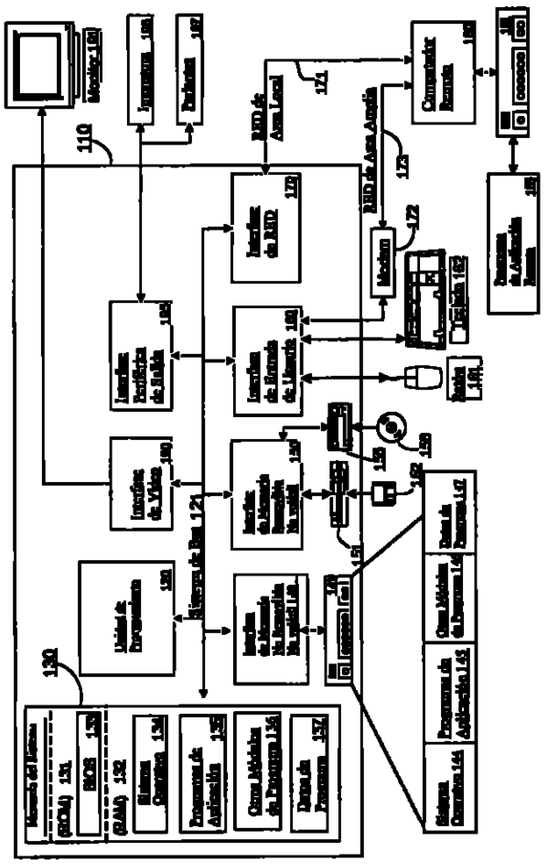

FIG. 1

25

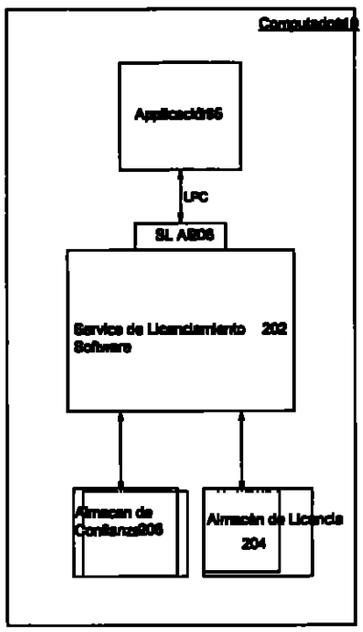
Computadora
Aplicación
LPC
SL ABS
Servicio de Licenciamiento 202
Software
Almacen de Confianza
Almacén de Licencia
204

FIG2

35


Inicio
Aplicación hace la llamada API
302
Servicio procesa la llamada API
304
Servicio retorna El resultado a la Aplicación
306
Aplicación determina El comportamiento Basado en el resultado
308

FIG3

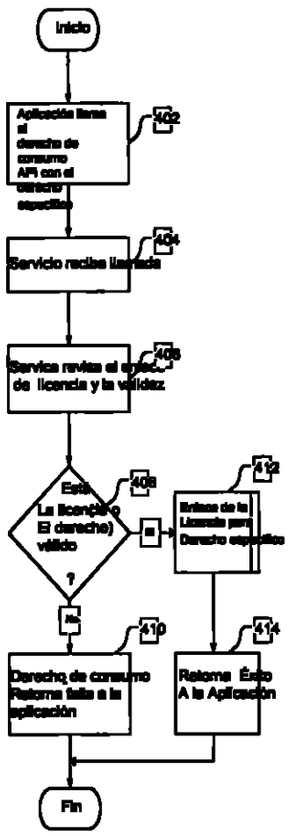
Inicio
Aplicación llama al derecho de consumo API con el derecho específico
402
Servicio recibe llamada
404
Servicio revisa el objeto de licencia y la validez
406
Está La licencia o El derecho) válido ?
408
Si
No
Enlace de la Licencia para Derecho específico
412
Retorna Éxito A la Aplicación
414
Derecho de consumo Retorna falla a la aplicación
410
Fin

FIG4

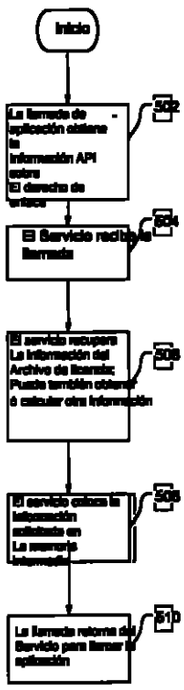
Inicio
La llamada de aplicación obtiene la información API sobre El derecho de enlace
502
El Servicio recibe la llamada
504
El servicio recupera La información del Archivo de licencia; Puede también obtener ó calcular otra información
508
El servicio coloca la información solicitada en La memoria intermedia
508
La llamada retorna del Servicio para llenar la aplicación
510

FIG5

PROGRAMMING INTERFACE FOR LICENSING

FIELD OF THE INVENTION

[0001] The present invention relates generally to the field of computer software, and, more particularly, to a programming interface that supports the enforcement of electronic licenses.

BACKGROUND OF THE INVENTION

[0002] Commercially-produced software has traditionally been made available under a license that defines the permissible terms of the software's use. When the practice of software licensing first began, the license generally took the form of a legal document that defined the user's rights with respect to the software. Such a document relied upon the legal system for enforcement. It has since become desirable for licenses to be enforced electronically – i.e., it is desirable that a computer program contain code that actively discourages or prevents use of the software in a manner that is contrary to the license.

[0003] Most software that provides for electronic license enforcement provides its own infrastructure to manage the licensing of the software and the use of the licenses. Thus, a typical commercial software product may include not only the code to perform the product's core function, but may also carry with it the code to obtain, evaluate, protect, and manage licenses for the software. For each software vendor to develop and incorporate this infrastructure into its software is often a wasteful duplication of effort. It is therefore desirable to provide a system that performs the basic functions related to software licensing, where the system can be used by a broad variety of software applications in a uniform and defined way.

[0004] In view of the foregoing, there is a need for a mechanism that overcomes the drawbacks of the prior art.

SUMMARY OF THE INVENTION

[0005] The present invention provides a software licensing Application Programming Interface (API) that provides certain licensing functions for use by software products. A license service performs functions relating to the use of licenses, and exposes these functions to software

products through the API. The service performs functions such as obtaining licenses, storing and managing licenses, protecting licenses from tampering, evaluating a license's validity, and evaluating whether a license is correctly bound to the machine and/or software product on which it is used. The software is able to make use of this functionality by calling the methods of the API.

[0006] In a typical use of the API, a software product calls an "open" API method in order to obtain a unique handle that is used by the license service to identify the application. The software product then calls a "consume right" API method. "Consume," in this context, means the exercise of a specified right. The call to the "consume right" method is parameterized by the software product's handle, and by the name of the right to be consumed. The license service then attempts to locate one or more valid, correctly bound licenses that contains the named right. If no such license exists, then the software product is notified of the failure. If such licenses exist, then the right is bound to one of the licenses, and the calling software product is notified of the binding. In such a case, the software product knows that the right exists, and can perform whatever functions are associated with this right.

[0007] In a preferred embodiment, the license service does not define what the software can or cannot do under the right, or enforce substantive constraints on the use of the software. Rather, the license service manages the licenses in such a way that a software product can determine by calling the API whether a right does, or does not, exist, so that the software can behave accordingly. For example, a right may be called "run," indicating that the user has the right to run the software product. The software product can use the API to determine whether there is a valid (and correctly bound, and non-expired) right to run the software. However, if the API call returns with an indication that there is no right to run the software, it is up to the software to cease operation or take some other action based on the non-existence of this right.

[0008] A right may be associated with information, which becomes available after a successful call to the "consume right" method. For example, a given software product may have individual rules about when it is permissible to edit, print, save, etc., and these rules can be stored in the license that contains the right. The API provides a "get information" method that allows this information to be retrieved from the license.

[0009] Other features of the invention are described below.

BRIEF DESCRIPTION OF THE DRAWINGS

[0010] The foregoing summary, as well as the following detailed description of preferred embodiments, is better understood when read in conjunction with the appended drawings. For the purpose of illustrating the invention, there is shown in the drawings exemplary constructions of the invention; however, the invention is not limited to the specific methods and instrumentalities disclosed. In the drawings:

[0011] FIG. 1 is a block diagram of an example computing environment in which aspects of the invention may be implemented;

[0012] FIG. 2 is a block diagram of an architecture in which a system performs licensing functions and exposes an API for use by software products;

[0013] FIG. 3 is a flow diagram of a method through which a software product uses a licensing API;

[0014] FIG. 4 is a flow diagram of a method by which a software product consumes a right; and

[0015] FIG. 5 is a flow diagram of a method by which a software product retrieves information relating to a consumed right.

DETAILED DESCRIPTION OF THE INVENTION

Overview

[0016] The use of commercial software is typically governed by a license, and it has become increasingly common for this license to be embodied in an electronic form that can be enforced by the software itself. One challenge in creating an electronic licensing system is that an infrastructure is needed to manage the use of the licenses. Replicating this infrastructure for every software product is cumbersome and wasteful. The present invention provides an API that allows different software products to use a common infrastructure that performs various licensing functions.

Exemplary Computing Arrangement

[0017] FIG. 1 shows an exemplary computing environment in which aspects of the invention may be implemented. The computing system environment 100 is only one example of a

suitable computing environment and is not intended to suggest any limitation as to the scope of use or functionality of the invention. Neither should the computing environment 100 be interpreted as having any dependency or requirement relating to any one or combination of components illustrated in the exemplary operating environment 100.

[0018] The invention is operational with numerous other general purpose or special purpose computing system environments or configurations. Examples of well known computing systems, environments, and/or configurations that may be suitable for use with the invention include, but are not limited to, personal computers, server computers, hand-held or laptop devices, multiprocessor systems, microprocessor-based systems, set top boxes, programmable consumer electronics, network PCs, minicomputers, mainframe computers, embedded systems, distributed computing environments that include any of the above systems or devices, and the like.

[0019] The invention may be described in the general context of computer-executable instructions, such as program modules, being executed by a computer. Generally, program modules include routines, programs, objects, components, data structures, etc. that perform particular tasks or implement particular abstract data types. The invention may also be practiced in distributed computing environments where tasks are performed by remote processing devices that are linked through a communications network or other data transmission medium. In a distributed computing environment, program modules and other data may be located in both local and remote computer storage media including memory storage devices.

[0020] With reference to FIG. 1, an exemplary system for implementing the invention includes a general purpose computing device in the form of a computer 110. Components of computer 110 may include, but are not limited to, a processing unit 120, a system memory 130, and a system bus 121 that couples various system components including the system memory to the processing unit 120. The processing unit 120 may represent multiple logical processing units such as those supported on a multi-threaded processor. The system bus 121 may be any of several types of bus structures including a memory bus or memory controller, a peripheral bus, and a local bus using any of a variety of bus architectures. By way of example, and not limitation, such architectures include Industry Standard Architecture (ISA) bus, Micro Channel Architecture (MCA) bus, Enhanced ISA (EISA) bus, Video Electronics Standards Association (VESA) local bus, and Peripheral Component Interconnect (PCI) bus (also known as Mezzanine bus). The system bus 121

may also be implemented as a point-to-point connection, switching fabric, or the like, among the communicating devices.

[0021] Computer 110 typically includes a variety of computer readable media. Computer readable media can be any available media that can be accessed by computer 110 and includes both volatile and nonvolatile media, removable and non-removable media. By way of example, and not limitation, computer readable media may comprise computer storage media and communication media. Computer storage media includes both volatile and nonvolatile, removable and non-removable media implemented in any method or technology for storage of information such as computer readable instructions, data structures, program modules or other data. Computer storage media includes, but is not limited to, RAM, ROM, EEPROM, flash memory or other memory technology, CDROM, digital versatile disks (DVD) or other optical disk storage, magnetic cassettes, magnetic tape, magnetic disk storage or other magnetic storage devices, or any other medium which can be used to store the desired information and which can accessed by computer 110. Communication media typically embodies computer readable instructions, data structures, program modules or other data in a modulated data signal such as a carrier wave or other transport mechanism and includes any information delivery media. The term "modulated data signal" means a signal that has one or more of its characteristics set or changed in such a manner as to encode information in the signal. By way of example, and not limitation, communication media includes wired media such as a wired network or direct-wired connection, and wireless media such as acoustic, RF, infrared and other wireless media. Combinations of any of the above should also be included within the scope of computer readable media.

[0022] The system memory 130 includes computer storage media in the form of volatile and/or nonvolatile memory such as read only memory (ROM) 131 and random access memory (RAM) 132. A basic input/output system 133 (BIOS), containing the basic routines that help to transfer information between elements within computer 110, such as during start-up, is typically stored in ROM 131. RAM 132 typically contains data and/or program modules that are immediately accessible to and/or presently being operated on by processing unit 120. By way of example, and not limitation, FIG. 1 illustrates operating system 134, application programs 135, other program modules 136, and program data 137.

[0023] The computer 110 may also include other removable/non-removable, volatile/nonvolatile computer storage media. By way of example only, FIG. 1 illustrates a hard disk drive 140 that reads from or writes to non-removable, nonvolatile magnetic media, a magnetic disk drive 151 that reads from or writes to a removable, nonvolatile magnetic disk 152, and an optical disk drive 155 that reads from or writes to a removable, nonvolatile optical disk 156, such as a CD ROM or other optical media. Other removable/non-removable, volatile/nonvolatile computer storage media that can be used in the exemplary operating environment include, but are not limited to, magnetic tape cassettes, flash memory cards, digital versatile disks, digital video tape, solid state RAM, solid state ROM, and the like. The hard disk drive 141 is typically connected to the system bus 121 through a non-removable memory interface such as interface 140, and magnetic disk drive 151 and optical disk drive 155 are typically connected to the system bus 121 by a removable memory interface, such as interface 150.

[0024] The drives and their associated computer storage media discussed above and illustrated in FIG. 1, provide storage of computer readable instructions, data structures, program modules and other data for the computer 110. In FIG. 1, for example, hard disk drive 141 is illustrated as storing operating system 144, application programs 145, other program modules 146, and program data 147. Note that these components can either be the same as or different from operating system 134, application programs 135, other program modules 136, and program data 137. Operating system 144, application programs 145, other program modules 146, and program data 147 are given different numbers here to illustrate that, at a minimum, they are different copies. A user may enter commands and information into the computer 20 through input devices such as a keyboard 162 and pointing device 161, commonly referred to as a mouse, trackball or touch pad. Other input devices (not shown) may include a microphone, joystick, game pad, satellite dish, scanner, or the like. These and other input devices are often connected to the processing unit 120 through a user input interface 160 that is coupled to the system bus, but may be connected by other interface and bus structures, such as a parallel port, game port or a universal serial bus (USB). A monitor 191 or other type of display device is also connected to the system bus 121 via an interface, such as a video interface 190. In addition to the monitor, computers may also include other peripheral output devices such as speakers 197 and printer 196, which may be connected through an output peripheral interface 195

[0025] The computer 110 may operate in a networked environment using logical connections to one or more remote computers, such as a remote computer 180. The remote computer 180 may be a personal computer, a server, a router, a network PC, a peer device or other common network node, and typically includes many or all of the elements described above relative to the computer 110, although only a memory storage device 181 has been illustrated in FIG. 1. The logical connections depicted in FIG. 1 include a local area network (LAN) 171 and a wide area network (WAN) 173, but may also include other networks. Such networking environments are commonplace in offices, enterprise-wide computer networks, intranets and the Internet.

[0026] When used in a LAN networking environment, the computer 110 is connected to the LAN 171 through a network interface or adapter 170. When used in a WAN networking environment, the computer 110 typically includes a modem 172 or other means for establishing communications over the WAN 173, such as the Internet. The modem 172, which may be internal or external, may be connected to the system bus 121 via the user input interface 160, or other appropriate mechanism. In a networked environment, program modules depicted relative to the computer 110, or portions thereof, may be stored in the remote memory storage device. By way of example, and not limitation, FIG. 1 illustrates remote application programs 185 as residing on memory device 181. It will be appreciated that the network connections shown are exemplary and other means of establishing a communications link between the computers may be used.

Software Licensing Service

[0027] FIG. 2 shows an example system that provides a software licensing service 202. Software licensing service 202 operates inside of computer 110 (shown in FIG. 1). In one example, software licensing service 202 is part of an operating system that executes on computer 110. Software licensing service maintains a license store 204 in which license files for software are stored. License files may, for example, be eXtensible Rights Markup Language (XrML) files that specify rights to software, and that also may specify various types of conditions on the exercise of those rights. Software 204 also maintains a trust store 206. Trust store 206 stores un-authenticatable, dynamic data in a tamper-resistant manner; trust store 206 stores data that is used in the license validation process. For example, certain licenses may have expiration dates, and, in order to prevent the expiration data from being circumvented through clock rollback, the current time (and elapsed

time) may be periodically stored in trust store 206 to ensure that the clock is always moving forward.

[0028] Software licensing service 202 manages license store 204 and trust store 206, and also performs various functions relating to the licensing of software. For example, software licensing service 202 may contain modules that parse license files, modules that enforce the binding of a license to a particular machine and/or to a particular instance of a software product, and a secure timer/counter (that uses trust store 206 in the manner described above).

[0029] Software licensing service 202 exposes an application programming interface (API) 208 that allows application software (such as application 135) to use software licensing service 202. Application 135 may invoke the features of software licensing service 208 by making local procedure call (LPC) to the methods of API 208. An example set of API methods that may be exposed by software licensing service 202 is described below.

[0030] The manner in which API 208 is used by an application is described with reference to FIG. 3. Initially, the application makes an API call (302). Service 202 then processes the API call (304), and returns the result of the API call to the application 306. For example, an API call may request to exercise ("consume") a right granted in a license, or to retrieve information from a license. The application then receives the result of the API call, and determines, based on that result, what the application's behavior should be (308). In other words, software licensing service 202, in a preferred embodiment, does not enforce a license directly, but rather provides the infrastructure through which licenses can be managed and used. For example, if an application makes an API call to consume a right and service 202 determines that there is no valid license granting this right, service 202, in a preferred embodiment, does not prevent the application from running, but instead informs the application that the right is not available. Thus, the application can use its own mechanisms to determine what to do in response to the unavailability of the right. This facet of the API provides software vendors the flexibility to decide how the licensing infrastructure provided by service 202 should be used. In another implementation, application can be bound to licensing service functionality.

Example Software Licensing API

[0031] The following is an example set of API methods that may be exposed by a software licensing service:

SLOpen

[0032] The SLOpen function opens a SL client context handle that must be used for all subsequent SL API calls. (Throughout the example API descriptions, "SL" shall refer to the software licensing service. Thus, the SL client context handle is the handle used by a client in communicating with the software licensing service.)

HRESULT

SLOpen(

CONST GUID* *pguidApp,*

HSLC* *phSLC*

);

Parameters

pguidApp

[0033] [in] Pointer to application GUID that uniquely identifies an application. If this argument is NULL, an E_INVALIDARG error is returned.

phSLC

[0034] [out] SL client context Handle or INVALID_HANDLE_VALUE if failed.

Remarks

[0035] Use the SLClose function to close an context handle returned by SLOpen.

[0036] Application GUID: Unique ID of application. For the WINDOWS version of the MICROSOFT OFFICE application suite, WinWord has an Application GUID which is different from Excel's Application GUID. For Windows, Windows itself is an application, although it is a composite of many programs.

[0037] In Office case, A user can install both Office Suite and WinWord standalone products on the machine. From SL point of view, the WinWord in both products has same

Application GUID. WinWord's Application GUID associated to two Product GUID. In other words, WinWord can use either Office Suite's product license or WinWord's product license.

[0038] When SLOpen succeeds:

[0039] A RPC (remote procedure call) binding has been established.

[0040] A context memory is created on SL service. The context is used to keep the status information for the caller of the client.

[0041] The SLC handle is like a file handle. A process can open multiple SL context handles but handles are valid within the caller process.

Returns

[0042] Success or failure

SLClose

[0043] The SLClose function closes an opened SL client context handle. Any information in the context is released automatically.

HRESULT

SLClose(

HSLC *hSLC*

);

Parameters

hSLC

[0044] [in] Handle to current SL client context handle

Remarks

[0045] When SLClose is done, the RPC binding is released, and the context is destroyed.

Returns

[0046] Success or failure.

SLInstall

[0047] The SLInstall function installs applications' licenses and registers applications' information.

HRESULT SLInstall(

HSLC *hSLC,*

CONST SL_PRODKEY* *pAppPrdKey,*

DWORD *dwNumOfApps,*

CONST GUID* *pguidApps,*

DWORD *dwNumOfLicFiles,*

PCWSTR *ppszLicFiles[],*

BOOL *bVerify*

);

Parameters

hSLC

[0048] [in] Handle to current SL client context handle

pAppPrdKey

[0049] [in] Application product key structure. The product key can be Microsoft product key format or Application's product key format.

```
typedef struct _tagSL_PRODKEY
{
  DWORD cbSize;                                    // Size of SL_PRODKEY structure
  DWORD dwVersion;                  // Version of SL_PRODKEY structure
  WCHAR szProdKey[MAX_PRODKEYSTR_SIZE+1];
  SL_PRODKEY_TYPE eProdKeyType;          // Type of Product key
  SL_CUSTOM_PRODKEY_INFO CustomPrdKeyInfo;   // Customer product key info
} SL_PRODKEY;
```

The eProdKeyType can be one of following values:

SL_PRODKEY_CUSTOM

SL_PRODKEY_MS2002

SL_PRODKEY_MS2003

[0050] If the Product Key type is non-MS Product Key (i.e. eProdKeyType == SL_PRODKEY_CUSTOM), the caller has to fill in its custom product key information. If the product uses MS Product Key, then CustomPrdKeyInfo can be ignored.

```
typedef struct _tagSL_CUSTOM_PRODKEY_INFO
  DWORD dwSKUID;                    // Unique ID for specific SKU, for example
Group ID in MS PID.
  DWORD dwSerialNumber;             // unique serial number, e.g. channel + sequence
number in MS PID.
} SL_CUSTOM_PRODKEY_INFO;
```

[0051] Current version number of SL_PRODKEY is 1. The caller can use SL_CURRENT_PRODKEY_VERSION in dwVersion field.

dwNumOfApps

[0052] [in] The number of application GUID in pguidApps.

pguidApps

[0053] [in] A list of application of GUID. The application GUID represents the application that the license is being installed for. For example, Office Setup program can call this function to install the license(s) for Word, Excel by specifying each application GUID in pguidApps. *pguidApp* cannot be NULL here.

dwNumOfApps

[0054] [in] The number of license files.

ppszLicFile

[0055] [in] File names in array of strings .

Returns

[0056] Success or failure

SLUninstall

[0057] The SLUninstall function uninstalls a product's license from an application.

HRESULT SLUninstall (

HSLC *hSLC,*

CONST SL_PRODKEY* *pAppPrdKey*

);

Parameters

hSLC

[0058] [in] Handle to current SL client context handle

pAppPrdKey

[0059] [in] See above defininition in connection with SLInstall.

Remarks

[0060] An application could have more than one product licenses. For example, when the user uninstalls Office suite, the association between Office Suite license and WinWord should be removed, but the license from WinWord Standalone product should not be removed.

[0061] When SLUninstall succeeds:

[0062] The information associated with this Product Key is removed. (see SLInstall, above, for the associated information)

[0063] The product keys associated with the Application GUID is removed.

[0064] The license files associated with the product GUID are preferably still kept.

Returns

[0065] Success or failure

SLConsumeRight

[0066] The SLConsumeRight function lets an application to examine or exercise the rights on a locally-stored license. Calling this function binds a license to the right mentioned in *pszRightName*. If this right cannot be exercised by the current caller, then the application fails. If the function succeeds, the action associated with the right can be executed (like decreasing usage count, decreasing time quota, or nothing)

HRESULT SLConsumeRight(
HSLC *hSLC,*
PCWSTR *pszRightName,*
SL_ASYNC_CONTEXT* *pAsyncContext*
);

Parameters

hSLC

[0067] [in] Handle to current SL client context handle

pszRightName

[0068] [in] The name of right needs to be evaluated. In current design, the right name is defined by applications. SL opens the license and evaluates the condition based on the right name.

pAsyncContext

[0069] [in/out] If *pAsyncContext* is NULL, this function works in synchronous mode, otherwise, the function works in asynchronous mode. **SL_ASYNC_CONTEXT** is opaque to caller and managed by SLC.

Remarks

[0070] All licenses associated with the application GUID (specified in SLOpen) will be conceptually combined in one logic license.

[0071] If their are multiple consumable grants of the right, then the license with higher priority will be consumed first.

Returns

[0072] Success or failure

SLInitializeAsyncContext

[0073] The SLInitializeAsyncContext function initializes the asynchronous context for SLC functions to make asynchronous call.

HRESULT SLInitializeAsyncContext(

SL_ASYNC_CONTEXT* ***pAsyncContext,*** // asynchronous context

HANDLE ***hEvent,*** // event handle

PVOID ***pvReserved*** // reserved, NULL

);

Parameters

pAsyncContext

[0074] [in/out] pointer to asynchronous context that contains asynchronous call information..

hEvent

[0075] [in] The event object used for synchronization.

pvReserved

[0076] [in] reserved for extension.

Returns

[0077] Success or failure.

SLCancelAsyncCall

[0078] The SLCancelAsyncCall function is used to cancel an asynchronous call.

HRESULT SLCancelAsyncCall(

SL_ASYNC_CONTEXT* ***pAsyncContext,*** // asynchronous context

SL

BOOL *fAbortCall* // cancel immediately

};

Parameters

pAsyncContext

[0079] [in] asynchronous context for SL asynchronous call.

fAbortCall

[0080] [in] If TRUE, the call is cancelled immediately. If FALSE, wait for the SL to complete the call.

Remarks

[0081] There are two ways for the caller to request cancellation of an asynchronous call—abortive and nonabortive. In an abortive cancel (fAbortCall is TRUE), the SLCancelAsyncCall function sends a cancel notification to the SLC and the asynchronous call is canceled immediately, without waiting for a response from the SLC. In a nonabortive cancel (fAbortCall is FALSE) the SLCancelAsyncCall function notifies SLC of the cancel and the caller waits for SLC to complete the call.

Returns

[0082] Success or failure.

SLCompleteAsyncCall

[0083] The SLCompleteAsyncCall function is used to complete an SLC asynchronous call.

HRESULT SLCompleteAsyncCall(

SL_ASYNC_CONTEXT* *pAsyncContext,* // asynchronous context

HRESULT* *phrAsyncCall* // error code of the submitted asynchronous call

};

Parameters

pAsyncContext

[0084] [in] asynchronous context for SL asynchronous call.

phrAsyncCall

[0085] [out] the error code of the submitted asynchronous call.

Remarks

[0086] If the caller calls this function before the reply has arrived, the call returns E_SLC_ASYNC_CALL_PENDING. The buffer must be valid and it must be big enough to receive the return value. If the call does not return E_SLC_ASYNC_CALL_PENDING, this SLCompleteAsyncCall invocation is final for the asynchronous call. After this function call, regardless of success or failure, all resources allocated for this asynchronous call are freed. (Subsequent calls to the SLCompleteAsyncCall or SLCancelAsyncCall functions have undefined results until a new call on the SL_ASYNC_CONTEXT structure is initiated).

Returns

Value	Meaning
S_OK	The call was completed successfully.
E_SLC_INVALID_ASYNC_CONTEXT	The asynchronous call context is not valid.
E_SLC_ ASYNC_CALL_PENDING	The call has not yet completed.
E_SLC_CALL_CANCELLED	The call was cancelled.

SLGetInformation

[0087] The SLGetLicenseInfo function is used to get a variety of information.

HRESULT SLGetInfomation(

HSLC *hSLC,* // SL client context handle

DWORD *dwCategory,* // The category of information to retrieve

```
    PCWSTR      pszKeyName,     // Name of the Key
    DWORD*      pdwType,        // Type of value
    SIZE_T*     pcbValue,       // Size of value
    PBYTE*      ppbValue        // Pointer to buffer of value
);
```

Parameters

hSLC

[0088] [in] Handle to current SL client context handle

dwCategory

[0089] [in] The category of information.

Category	Meaning
SL_CAT_RIGHTDATA	Get the information from bound right. The license has to be consumed successfully before getting these right data.
SL_CAT_DYNAMICPROPERTY	Get the information that is not in the license but calculating in the run-time. For example, RemainingGracePeriodDays. The right has to be consumed before calling.

Name	Meaning
RemainingGracePeriodDays : DWORD	The grace period is defined in out-of-box license. Once the application is installed, the time is counting down. Applications can check remaining grace period after they have consumed license.

<table>
<tr><td>ActivationStatus : DWORD</td><td>After applications consumed license, it can get consumed license type. The return value could be:
<table>
<tr><td>SL_LIC_OOB</td><td>The consumed license is out-of-box license.</td></tr>
<tr><td>SL_LIC_ACQUIRED</td><td>The consumed license is acquired license.</td></tr>
<tr><td>SL_LIC_NONE</td><td>No license is available.</td></tr>
</table></td></tr>
</table>

SL_CAT_SERVICEINFO

Get the information that is not dependent on license. The caller can get this category of information without consuming license.

Name	Meaning
SLVersion: DWORD	The version of SL. 1.2.3.4 format.
HWID:BINARY	Current HWID

SL_CAT_WINDOWSINFO

Get information that is bound right property in Windows license. This is for Componentization. Windows License

	has been consumed by SL service and SL service keeps the bound right properties.
SL_CAT_ENUMLICINFO	When SLEnumLicenseis called and is succeed, the caller can query the information of enumerated license by using this category.

pszKeyName

[0090] [in] the name of the key. E.g. BuildNumber

pdwType

[0091] [out] type of data

Value	Meaning
SL_DATATYPE_SZ	Unicode string
SL_DATATYPE_DWORD	DWORD
SL_DATATYPE_BINARY	Binary

pcbValue

[0092] [out] Size of the buffer allocated (in bytes).

ppbValue

[0093] [out] If successful, the data is returned in the buffer allocated by SLC. The caller has to call SLFreeMemory to free the memory.

Returns

[0094] Success or failure

SLAcquireLicense

[0095] The SLAcquireLicense function is used to acquire on-line license for the user. SLC enumerates the product keys associated with the Application and picks up the product key with highest product priority (see SLInstall, registration information). Then SL gets the clearing house URL from out-of-box license and connects to clearing house to get a license.

[0096] SLAcquireLicense could be a lengthy process. Applications can call this function in asynchronous mode by specifying *pAsyncContext (NULL means synchronous mode)*.

```
HRESULT SLAcquireLicense(
  HSLC      hSLC,          // SL client context handle
  PCWSTR    pszProdKeyHash,    // hash of product key
  PCWSTR    pszPublishLicense,      // string of publishing license.
  SL_ASYNC_CONTEXT*  pAsyncContext
);
```

Parameters

hSLC

[0097] [in] Handle to current SL client context handle

pszProdKeyHash

[0098] [in] string of product key hash. The product key hash is created when SLInstall is called and is maintained by the licensing service.

pszPublishingLicense

[0099] [in] string of publishing license.

pAsyncContext

[0100] [in] asynchronous context for SL asynchronous call.

Remarks

[0101] The acquired license will be stored in license store accordingly and the license information will be registered, too. (see SLInstall)

[0102] Applications might need to add more client information to license. The application can put the information to pbAppData in the call and this data will be sent to clearing house.

[0103] When this function succeeds:

[0104] It sent necessary binding information to the specified license server.

[0105] It receives the license from license server.

[0106] It stored the license in license store. See description of SLInstall, above, for how the license file is stored.

Returns

[0107] Success or failure.

SLGenerateTextChallenge

[0108] Generates and installation challenge text to be routed to a license issuer in an out of band fashion (telephone, email, file share, etc).

```
HRESULT SLGenerateTextChallenge(
  HSLC      hSLC,          // SL client context handle
  PCWSTR    pszProdKeyHash,        // string of product key hash
  BOOL      fSingleSession,  // Single session only?
  PWSTR     *ppszChallenge    // Pointer to buffer to hold challenge text
);
```

Parameters

hSLC

[0109] [in] Handle to current SL client context handle

pszProdKeyHash

[0110] [in] String of product key hash. The product key hash is created when SLInstall is called and maintained by the licensing service.

bSingleSession

[0111] [in] Specifies whether or not the corresponding text response from the license issue will be valid only for the lifetime of this SLC session handle.

ppszChallenge

[0112] [out] If successful, the text challenge is returned in the buffer allocated by SLC. The caller needs to call SLFreeMemory to free the allocated memory.

Returns

[0113] Success or failure.

SLDepositTextResponse

[0114] Deposits the response to an installation challenge text in the licensing system. Used to activate a license with a conditional access code. Only valid if there is an outstanding text challenge which has been issued for a license. If the original license specified that the challenge was only valid for a single session, this API must be called with the response before the SLC handle is closed or depositing the response will fail.

```
HRESULT SLDepositTextResponse(
  HSLC      hSLC,          // SL client context handle
  PCWSTR    pszProdKeyHash,
  PWSTR     pszResponse    // Buffer containing response text
);
```

Parameters

hSLC

[0115] [in] Handle to current SL client context handle

pszProdKeyHash

[0116] string of the product key hash

pszChallenge

[0117] [in] Response text

Returns

[0118] Success or failure.

SLEnumLicense

[0119] The SLEnumLicensefunction is used to enumerate installed licenses and get information from the license.

```
HRESULT SLEnumLicense(
  HSLC        hSLC,          // SL client context handle
  CONST GUID* pguidApp,      // application GUID
  DWORD       dwIndex        // index number
);
```

Parameters

hSLC

[0120] [in] Handle to current SL client context handle

pguidApp

[0121] [in] See SLInstall. If *pguidApp* is not NULL, then licenses associated with this GUID are enumerated. If the GUID is NULL, then all licenses are enumerated.

dwIndex

[0122] [in] The index number of the license to be retrieved. This value should be zero for the first call to the SLEnumLicense function and then incremented for subsequent calls.

[0123] The function may return licenses in any order.

Returns

60

[0124] E_SL_NO_MORE_DATA – No license at the specified index.

Remarks

[0125] If SLEnumLicenses succeeded, the selected license information can be accessed by calling SLGetInformation and the category is SL_CAT_ENUMLICINFO

Sample:

```
DWORD i=0;

PBYTE pbProductPriority = NULL;
PBYTE pbRemainingGracePeriodDays = NULL;

for (i=0; ; i++)
{
  EXIT_ON_ERROR(SLEnumLicense(hSLC, NULL, dwIndex));

  if (E_SL_NO_MORE_DATA == SCODE(hr))
  {
    hr = S_OK;
    break;
  }

  EXIT_ON_ERROR(SLGetInformation(hSLC, SL_CAT_ENUMLIC, "ProductPriority",
&dwType, &cbProductPriority, &pbProductPriority));

  EXIT_ON_ERROR(SLGetInformation(hSLC, SL_CAT_ENUMLIC,
"RemainingGracePeriodDays", &dwType, &cbRemainingGracePeriodDays,
&pbRemainingGracePeriodDays));
```

```
Exit:
  SLFreeMemory(pbProductPriority);
  SLFreeMemory(pbRemainingGracePeriodDays);

}
```

SLFreeMemory

[0126] The SLFreeMemory function is used to free the memory allocated by SLC.

```
VOID SLFreeMemory(
  PVOID          pvMemblock,     // pointer to memory
);
```

Parameters

pvMemBlock

[0127] [in] Previously allocated memory block to be freed

Returns

[0128] None

Use of Software Licensing API to Control Use of Software

[0129] A software product uses the API of the present invention for various purposes related to licensing, including the consumption of rights in a license, and the retrieval of data from the license. As noted above, the API allows the software to determine what rights are present in the license, but, preferably, it is up to the software to determine what to do with that information – e.g., grant or deny access to a feature, cease operation altogether, etc. The following description of FIGS. 4 and 5 show how the API of the present invention is used by a software product.

[0130] FIG. 4 shows an example process by which an application "consumes" a right. The application calls the SLConsumeRight method (402). As discussed above, the arguments to the SLConsumeRight function include the client handle assigned by the licensing service, and the name

of the right (which is assigned by the vendor of the software to which the right pertains). The licensing service (service 202, shown in FIG. 2) receives the call (404). The service then locates licenses that contains the right, and checks the licenses bindings and validity. As noted above, the license is located in the license store; if there is more than one license that pertains to the application software to which the SLConsumeRight call pertains, then a priority rule may be used to select one of the applicable licenses. Checking the binding means determining that: (1) the license is bound to the product key of the application identified by the client handle; and (2) the license is bound to the machine on which the software is running (or to the group of machines of which the current machine is a member). Checking validity may include determining that the right has not expired (in the case of licenses that specify an expiration date), and that the maximum number of uses of the right is not exceeded (in the case where the license specifies a maximum number of times that the right may be used (i.e., "consumed")).

[0131] If the license and/or right are found to be correctly bound and valid (408), then the license is bound to the right requested in the API call (412). (It should be noted that "binding" a license to a machine, environment, and a product key means that the license specifies which machine(s) and product key it can be used with; "binding" a license to a right means that the consume function has been successful, and the right is being consumed from a particular license. Throughout this description, it will be clear from context which meaning of "binding" applies.) The API call then returns to the calling application and indicates that the call was successful (414). If the license and/or right has been found to be invalid, or not correctly bound to the machine, environment, or product ID, then the SLConsumeRight call returns to the calling application and indicates that the operation failed (410).

[0132] If the SLConsumeRight call returns with a failure, then the right specified in the call cannot be consumed from a license, and no information about that right will be available to the calling application. However, if the right is successfully consumed, then the application can use the binding of the right to the license to get information from the license about the right. For example, a license may contain a general right called "run," which indicates that the application may be run. However, for the "run" right, the license may contain more specific parameters about the usage of the application – e.g., the license may specify whether particular features of an application (e.g., print, edit, save, etc.) should be turned on or off, and may give specific parameters for the use of

these features (e.g., the document can be saved only on machines that are running in a particular domain, or the print feature can only be used for thirty days, etc.). The SL API does not require any particular type of information to be associated with a right, but rather provides a mechanism whereby an application vendor can associate any type of information with a right, which can then be retrieved and interpreted by the application.

[0133] Assuming that a right has been successfully consumed as described in FIG. 4, the application may then retrieve the information associated with the right. The process of retrieving this information is described in FIG. 5.

[0134] First, the application calls the SLGetInformation method on the bound right (502). The various types of information that can be retrieved are described above in connection with the description of the SLGetInformation method. The licensing service then receives the call (504). The service retrieves the requested information from the license file that contains the bound right (506). The licensing service then places this information in a buffer (508), and returns to the calling application (510). The calling application then reads the contents of the buffer, and performs whatever actions it deems necessary based on the retrieved information.

[0135] It should be noted that the licensing service may not be aware of the meaning of the information that it is handling as part of an SLGetInformation call. As discussed above, the licensing framework provides a mechanism whereby a software vendor can create rights, and can associate information with the rights. The invention is not limited either to any particular type of information that can be associated with the right. When the information is retrieved from the license, it is simply passed by the licensing service to the application in a buffer. The application then interprets the retrieved information, decides what actions to be taken based on that information, and uses its own security features to enforce the application's decision. (E.g., if, based on the retrieved information, the application decides to disable the print feature, the application contains the code that actually disables this features and, possibly, code that prevents a hacker from tampering with the disabling of the print feature.)

[0136] It is noted that the foregoing examples have been provided merely for the purpose of explanation and are in no way to be construed as limiting of the present invention. While the invention has been described with reference to various embodiments, it is understood that the words which have been used herein are words of description and illustration, rather than words of

limitations. Further, although the invention has been described herein with reference to particular means, materials and embodiments, the invention is not intended to be limited to the particulars disclosed herein; rather, the invention extends to all functionally equivalent structures, methods and uses, such as are within the scope of the appended claims. Those skilled in the art, having the benefit of the teachings of this specification, may effect numerous modifications thereto and changes may be made without departing from the scope and spirit of the invention in its aspects.

What is Claimed:

1. A system for supporting the enforcement of a license for a computer program, the system comprising:

a licensing component that maintains a license store in which the license is stored, the license comprising a right in the software and a set of data associated with said right, the licensing component exposing a callable interface to the computer program, said callable interface comprising:

a right-consumption method which receives an identifier of said right from the computer program and determines whether the right can be exercised; and

an information-retrieval method which receives an identifier of said right from the computer program and provides said set of data, or information based on said set of data, to the computer program.

2. The system of claim 1, wherein said licensing component is usable by a plurality of computer programs, the computer program being included among said plurality of computer programs, wherein said callable interface further comprises:

a handle-opening method that provides a handle to the computer program; wherein the rights-consumption method receives the handle from the computer program and uses the handle to identify the computer program from which a call to the rights-consumption method is received.

3. The system of claim 1, wherein the license is one of a plurality of licenses that are stored in said license store, and wherein the rights-consumption method causes the licensing component to select the license based on one or more factors comprising:

whether the license store is associated with the computer program; and

a conflict rule that determines which license to select from among a plurality of licenses that are associated with the computer program.

4. The system of claim 1, wherein said callable interface further comprises:

an asynchronous-context-initiator method that establishes a context for asynchronous

processing and provides an identifier of said context to the computer program;
wherein said rights-consumption method receives the identifier of said context from said computer program and processes a right-consumption request asynchronously in response to receipt of the identifier of said context.

5. The system of claim 1, wherein the rights-consumption method determines whether the right can be exercised based on whether the right is identified in the license.

6. The system of claim 1, wherein the computer program and the licensing component execute on a machine, and wherein the rights-consumption method determines whether the right can be exercised based on whether the license is bound to said machine.

7. The system of claim 1, wherein the computer program is associated with a product identifier, and wherein the rights-consumption method determines whether the right can be exercised based on whether the license is bound to said machine or to a class of machines of which said machine is a member.

8. A method of restricting the use of a computer program associated with a license, the license specifying a right in the computer program, the method comprising:

invoking a licensing service by making a first call to a first method of an interface of said licensing service, said first call being parameterized by an identifier associated with said right;

in response to said first call receiving an indication as to whether the right is exercisable; and

engaging in either a first behavior or a second behavior according to the indication.

9. The method of claim 8, wherein said first behavior comprises allowing the computer program to execute, and wherein said second behavior comprises discontinuing execution of the computer program.

10. The method of claim 8, wherein said first behavior comprises allowing the computer program to perform a first set of functions, and wherein said second behavior comprises allowing

the computer program to perform a second set of functions that is non-identical to said first set of functions.

11. The method of claim 8, wherein the right is associated with a set of data, wherein the method further comprises:

making a second call to a second method of said interface, said second method being parameterized by an indication of the right; and

in response to said second call, receiving said set of data.

12. The method of claim 11, further comprising:

directing the operation of the computer program based on said set of data.

13. The method of claim 8, further comprising:

making a second call to a second method of said interface; and

in response to said second call, receiving a handle;

wherein said second call is made prior to said first call, and wherein said first call is further parameterized by said handle.

14. The method of claim 8, further comprising:

making a second call to a second method of said interface; and

in response to said second call, receiving an asynchronous context;

wherein said second call is made prior to said first call, wherein said first call is further parameterized by said asynchronous context, and wherein the computer program performs at least one action while the first call is handled asynchronously.

15. The method of claim 8, wherein said first method determines whether the right is exercisable based on one or more factors comprising:

whether the license is bound to a machine or environment on which the computer program is executing;

whether the license or right is bound to a product identifier associated with the

computer program;

whether the license or right has expired; and

whether the right has been consumed a number of times in excess of a right specified in the license.

16. A computer-readable medium having encoded thereon computer-executable instructions to perform a method of enabling the enforcement of a license to a computer program, the method comprising:

receiving a first method call from the computer program, the first method call identifying a right in the computer program;

determining that the right is contained in the license and is exercisable; and

returning to the computer program an indication that the right is exercisable.

17. The computer-readable medium of claim 16, wherein the indication comprises a binding of the right to the license.

18. The computer-readable medium of claim 16, wherein said determining act is based on whether the right is specified in the license.

19. The computer-readable medium of claim 16, wherein said determining act is based on whether the license is bound to a machine on which the computer program is executing.

20. The computer-readable medium of claim 16, wherein said determining act is based on whether the license or right is bound to the computer program.

21. The computer-readable medium of claim 16, wherein said determining act is based on whether the license or right is non-expired.

22. The computer-readable medium of claim 16, wherein said determining act is based on whether the license has been consumed a number of times that exceeds a limit.

23. The computer-readable medium of claim 16, wherein the method further comprises:

receiving a second method call from the computer program; and

in response to the second method call, returning a handle to the computer program that identifies the computer program;

wherein said first method call is performed subsequent to said second method call, and wherein said first method call further identifies said handle.

24. The computer-readable medium of claim 16, wherein the method further comprises:

receiving a second method call from the computer program;

in response to the second method call, returning an asynchronous context to the computer program, wherein the first method call is executed subsequent to the second method call and identifies said asynchronous context; and

executing the first method call asynchronously while the computer program performs an action.

25. The computer-readable medium of claim 16, wherein the right is associated with a set of data, and wherein the method further comprises:

receiving a second method call which indicates the right; and

in response to said second method call, providing the set of data to the computer program.

ABSTRACT OF THE DISCLOSURE

A software licensing Application Programming Interface (API) that allows software products to use the license management functionality of a common service. A license specifies rights in a software product. The software product calls a consume method on the API in order to consume a right. If the right exists, the service binds the right to the license in which the right is found. The software product enforces the terms of the license by granting, or denying, access to some or all features depending on whether a valid instance of the right is found. Arbitrary data can be associated with a right. The API includes a method to retrieve data from a right that has been previously bound by the consume method.

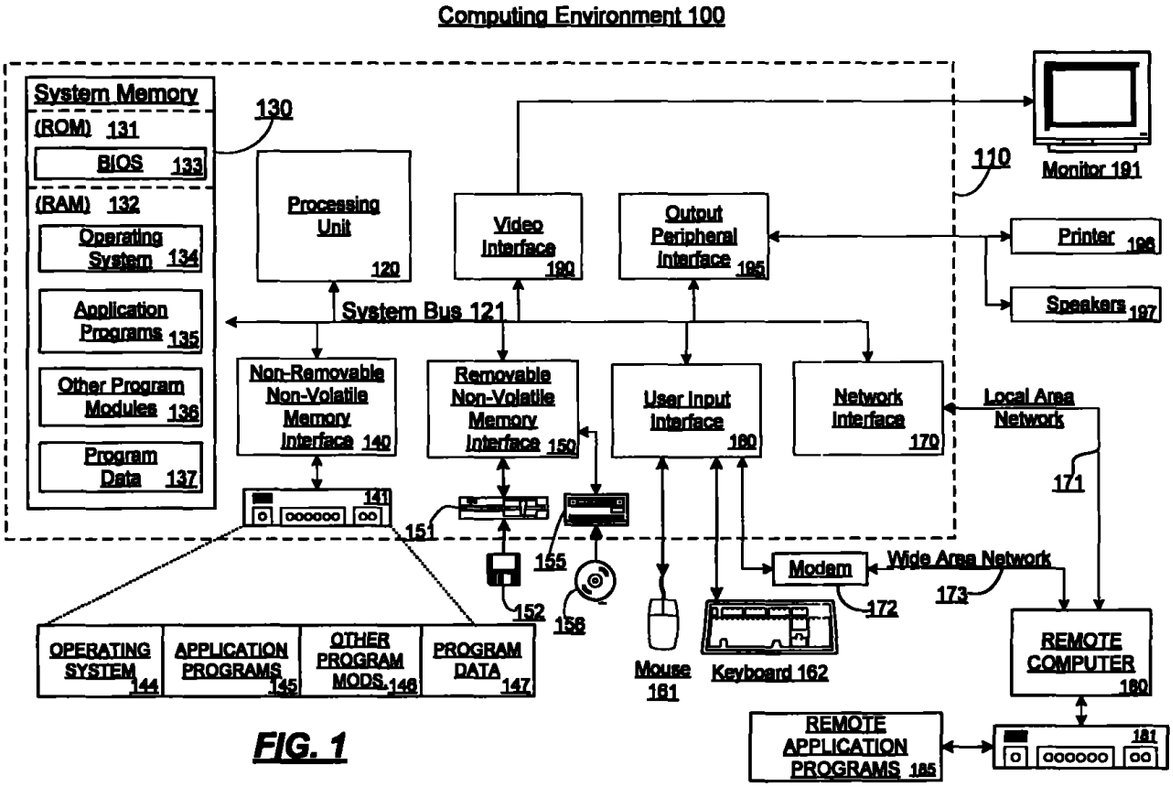
Computing Environment 100
System Memory
(ROM) 131
BIOS 133
(RAM) 132
Operating System 134
Application Programs 135
Other Program Modules 136
Program Data 137
130
Processing Unit 120
Video Interface 190
Output Peripheral Interface 195
Monitor 191
110
Printer 196
Speakers 197
System Bus 121
Non-Removable Non-Volatile Memory Interface 140
Removable Non-Volatile Memory Interface 150
User Input Interface 160
Network Interface 170
Local Area Network
171
141
151
155
152
156
Modem
Wide Area Network
172
173
OPERATING SYSTEM 144
APPLICATION PROGRAMS 145
OTHER PROGRAM MODS. 146
PROGRAM DATA 147
Mouse 161
Keyboard 162
REMOTE COMPUTER 180
REMOTE APPLICATION PROGRAMS 185
181

FIG. 1

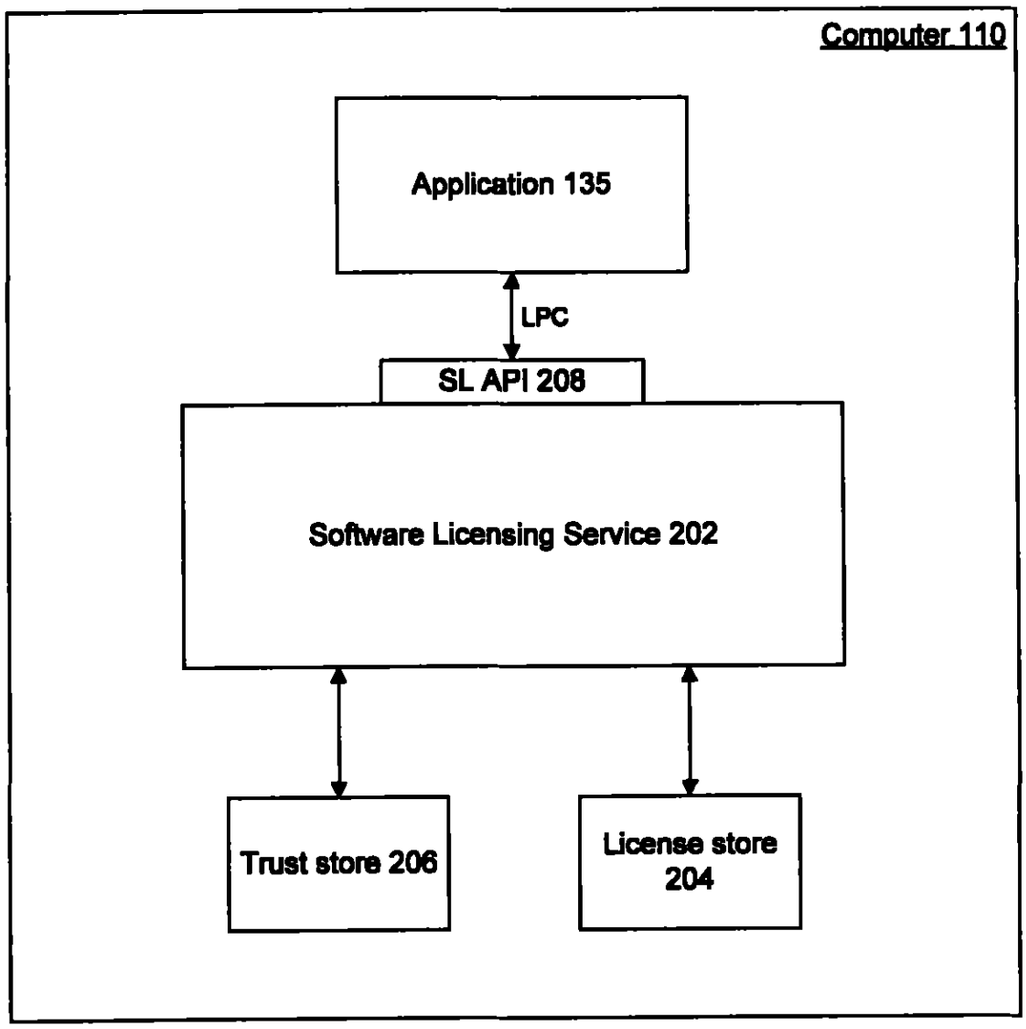
Computer 110
Application 135
LPC
SL API 208
Software Licensing Service 202
Trust store 206
License store 204

FIG. 2

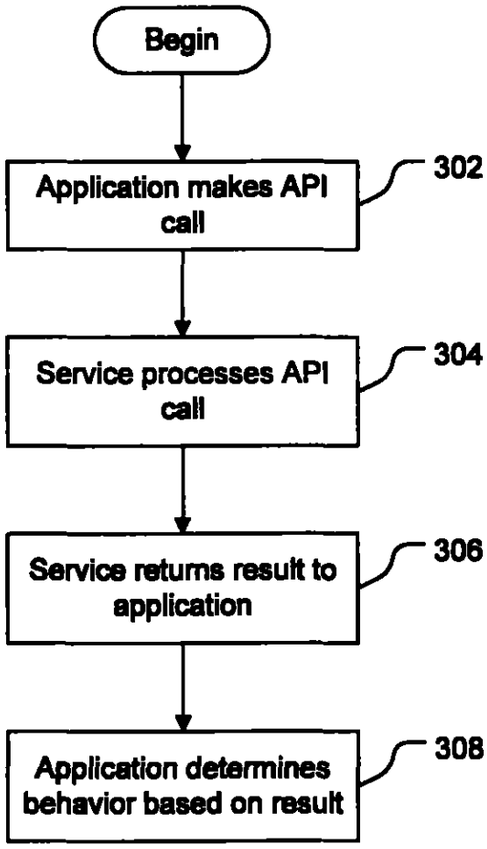
Begin
Application makes API call
302
Service processes API call
304
Service returns result to application
306
Application determines behavior based on result
308

FIG. 3

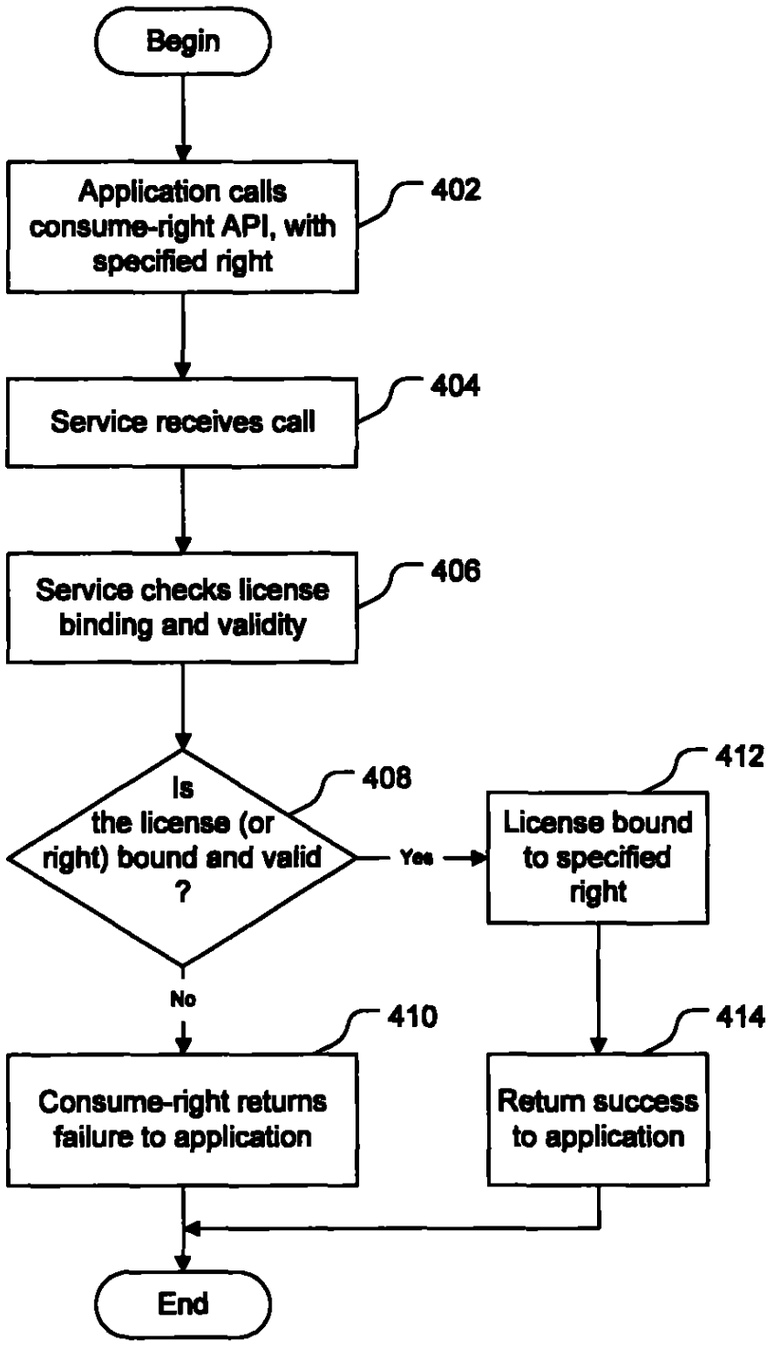
Begin
Application calls consume-right API, with specified right
402
Service receives call
404
Service checks license binding and validity
406
Is the license (or right) bound and valid ?
408
Yes
License bound to specified right
412
No
Consume-right returns failure to application
410
Return success to application
414
End

FIG. 4

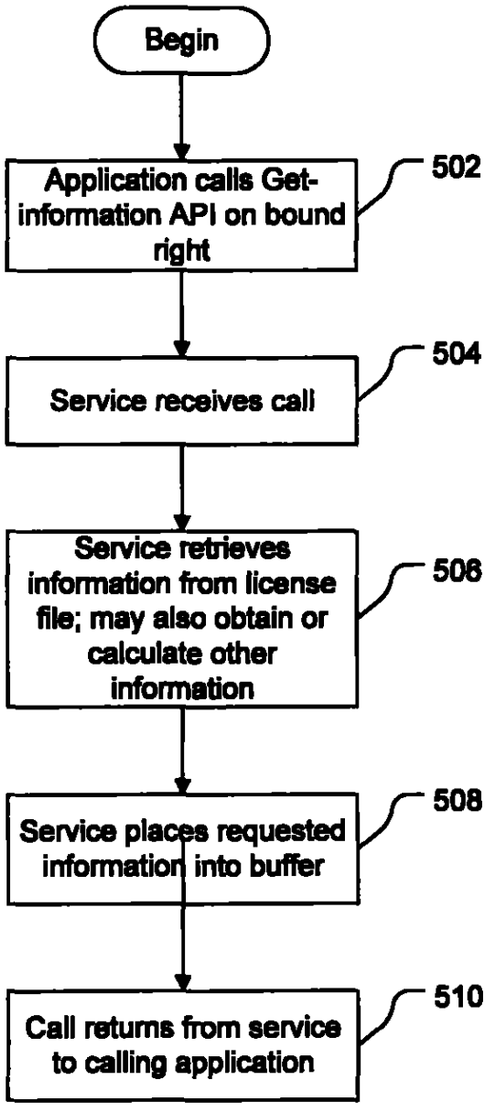
Begin
Application calls Get-information API on bound right
502
Service receives call
504
Service retrieves information from license file; may also obtain or calculate other information
506
Service places requested information into buffer
508
Call returns from service to calling application
510

FIG. 5

50 76
77

REQUISITOS NECESARIOS PARA PRESENTACION Y AGILIZACION DEL TRAMITE

	USUARIO	RADICADOR
PATENTE DE INVENCIÓN 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 identificacion 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 solcitante 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

Fecha

SUPERINDUSTRIA Y COMERCIO
Radicacion 04094972 00000000
Fecha (AMD): 2004-09-23 17:00:27 Folios 76
Tramite 002 PATENTES D 1 REGISTRO/ 411 PRESENTA
Dependencia: 2020 DIVISION DE NUEVAS CREACIONES

77 378

PROTOCOLO

Expediente 04-94972

SUPERINDUSTRIA Y COMERCIO
SISTEMA DE REGISTRO

MANTENIMIENTO DE PROTOCOLOS

Numero : 16069
F cha : 04/08/1995
Conferido: MICROSOFT CORPORATION

Pais : US ESTADOS UNIDOS DE AMERICA

Ciudad : 234 REDMOND, WASHINGTON

Represent: S Sustituir : S Desistir : S
No. Radicacion : 95 34920 Clase: 0 Seev: 0 Seac: 0

scncia	Codigo	Ctr	Nombre
1	77	A	CARDENAS NAVAS DARIO
2	3653		CARDENAS CABALLERO EDUARDO
3	5048		CARDENAS MARTINEZ BERNARDO

Encontrados (1/1) registro(s)

Carrera 13 No. 27-00 Pisos 5, 7 y 10
Conmutador: 382 0840
Fax: 350 5220
E-mail: info sic.gov.co
www.sic.gov.co
Bogotá, D.C., Colombia

2020
Bogotá, D. C.

Doctor(a)
DARIO CARDENAS NAVAS
Apoderado(a)

Oficio No. 10015

REFERENCIA:

Radicación	04-94972
Trámite	002
Evento	001
Actuación	430
Folios	001

Como resultado del examen de forma, realizado con fundamento en el artículo 38 de la Decisión 486 de la Comisión de la Comunidad Andina, se le comunica que:

- Allegar copia del documento en que conste la cesión del derecho a la patente otorgado por los inventores al solicitante o a su causante. (art. 26-k,Dec 486/2000).

En cumplimiento del artículo 39 de la Decisión 486, se le requiere para que dentro del término de dos (2) meses siguientes a la fecha de notificación del presente oficio, complete los requisitos anteriormente enunciados. En caso de no dar cumplimiento al presente requerimiento, la solicitud se considerará abandonada y perderá su prelación.

Notifíquese el presente oficio conforme a lo dispuesto en el numeral 5.2, literal e), del capítulo quinto del título primero de la Circular Unica No.10 de 2001.

ALIX CESPEDES DE VERGEL
Jefe de la División de Nuevas Creaciones

El anterior requerimiento fue notificado por fijación en lista de fecha 28 OCT 2004

Se recuerda al interesado que debe presentar, dentro de los diecisés meses siguientes contados a partir de la fecha de presentación de la solicitud cuya prioridad se invoca, copia de la misma certificada por la autoridad que la expidió y, en los casos en que haya lugar, la traducción simple del capítulo reivindicatorio. El incumplimiento de este plazo, acarreará la pérdida de la prioridad invocada.

202027