

(19) World Intellectual Property Organization
International Bureau



(43) International Publication Date
8 May 2003 (08.05.2003)

PCT

(10) International Publication Number
WO 03/038570 A2

- (51) International Patent Classification⁷: **G06F 1/00**
- (21) International Application Number: PCT/EP02/12190
- (22) International Filing Date: 10 October 2002 (10.10.2002)
- (25) Filing Language: English
- (26) Publication Language: English
- (30) Priority Data:
01480108.8 30 October 2001 (30.10.2001) EP
- (71) Applicant (for all designated States except MC, US):
INTERNATIONAL BUSINESS MACHINES CORPORATION [US/US]; New Orchard Road, Armonk, NY 10504 (US).

L'Aquila (IT). **DI GIULIO, Domenico** [IT/IT]; Via Alessio Baldovinetti, 85, I-00142 Roma (IT). **ROSSI, Riccardo** [IT/IT]; Via Leopoldo Ruspoli, 119, I-00149 Roma (IT).

(74) Agent: **ZERBI, Guido**; Compagnie IBM France, Direction de la Propriété Intellectuelle, F-06610 La Gaude (FR).

(81) Designated States (national): AE, AG, AL, AM, AT, AU, AZ, BA, BB, BG, BR, BY, BZ, CA, CH, CN, CO, CR, CU, CZ, DE, DK, DM, DZ, EC, EE, ES, FI, GB, GD, GE, GH, GM, HR, HU, ID, IL, IN, IS, JP, KE, KG, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, MA, MD, MG, MK, MN, MW, MX, MZ, NO, NZ, OM, PH, PL, PT, RO, RU, SD, SE, SG, SI, SK, SL, TJ, TM, TN, TR, TT, TZ, UA, UG, US, UZ, VN, YU, ZA, ZM, ZW.

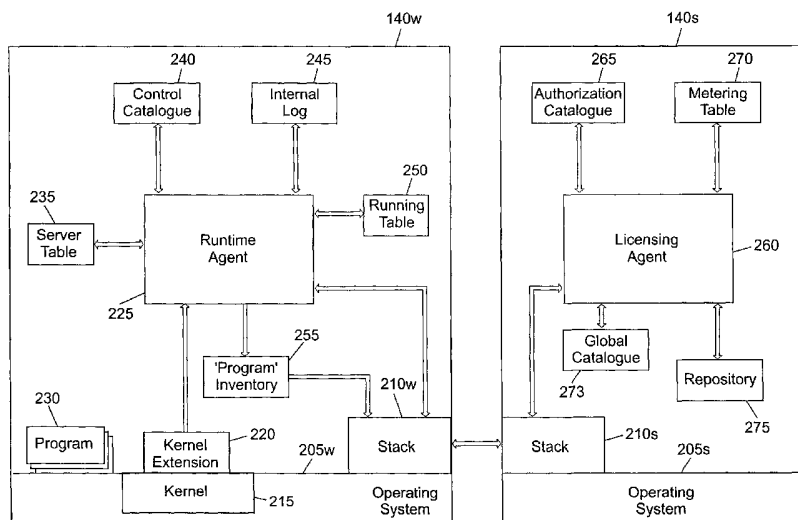
(71) Applicant (for MC only): **COMPAGNIE IBM FRANCE** [FR/FR]; Tour Descartes, 2, avenue Gambetta, F-92066 Paris La Défense Cedex (FR).

(84) Designated States (regional): ARIPO patent (GH, GM, KE, LS, MW, MZ, SD, SL, SZ, TZ, UG, ZM, ZW), Eurasian patent (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European patent (AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE, SK, TR), OAPI patent (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, ML, MR, NE, SN, TD, TG).

(72) Inventors; and
(75) Inventors/Applicants (for US only): **PASTORELLI, Bernardo** [IT/IT]; Via XX Settembre, 13/A, I-176100

[Continued on next page]

(54) Title: A COMPUTER-IMPLEMENTED METHOD AND SYSTEM FOR CONTROLLING USE OF DIGITALLY ENCODED PRODUCTS



(57) Abstract: A method (300) and system for controlling use of software programs on multiple workstations of a network. An agent running on each workstation detects (306) any new process (using a kernel hooking technique), which is started in response to the request of execution of a program (397). The process is suspended (308) and an authorisation to run the program is requested (334) to an associated metering server. The metering server compares (380) the current usage of the product with the respective authorized conditions of use, and answers (382) the agent granting or denying the authorisation to run. The process is accordingly resumed (312), in order to enable the program to start, or aborted (315), so as to prevent execution of the program. In any case, the program is not stopped (but it is prevented from starting); this avoids killing the program while it is performing potentially risky operations.

WO 03/038570 A2



Declaration under Rule 4.17:

— *of inventorship (Rule 4.17(iv)) for US only*

Published:

— *without international search report and to be republished upon receipt of that report*

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

**A COMPUTER-IMPLEMENTED METHOD AND SYSTEM FOR CONTROLLING USE
OF DIGITALLY ENCODED PRODUCTS**

The present invention relates to a method and system for controlling use of digitally encoded products.

5 Digitally encoded products, such as software programs, can be perfectly reproduced in an infinite number of copies. However, unauthorised copying of programs generally involves infringement of intellectual property rights of their publishers. For this reason, it is of the utmost importance to
10 verify compliance of the employed programs with corresponding authorised conditions of use. This problem is particularly acute in big organisations having a large number of computers, and it has been exacerbated in the last years by the widespread diffusion of the INTERNET, which further
15 facilitates the uncontrolled distribution of this kind of products.

 The above-mentioned verification is usually performed auditing all the computers manually. However, this procedure is time consuming (particularly in large sites), and prone to
20 errors.

 Some automatic tools have been also proposed for evaluating compliance of the programs with their authorised conditions of use. Typically, these tools exploit an agent installed on each computer. The agent periodically checks the
25 programs running on the computer. The result of the

verification is logged, in order to generate program usage reports. Moreover, if a program is running violating its authorised conditions of use the tool may kill the respective process (so as to stop execution of the program), and may make
5 the program unable to start again.

However, the solution described above does not monitor usage of the programs in real-time, so that it is not possible to perform an actual licensing validation. In addition, when execution of a program is killed to enforce compliance with
10 its authorised conditions of use, damage to the corresponding data may be suffered; this also forces an end-user to re-install the program on the computer for a next (authorised) usage.

A different solution proposed in the art, known as
15 technical licensing, consists of modifying each program to include a call to a licensing management system. Every time the end-user starts the program a corresponding request is transferred to the licensing management system. The licensing management system verifies whether the running of the program
20 falls within the limits set out by the authorised conditions of use embedded in a respective digital certificate, and provides a corresponding return code to the program. If the result of the verification is positive the program can continue its execution; otherwise, the program is forced to
25 stop.

However, the solution described above requires each program to be modified by the publisher. In addition, it is quite rigid and cannot be readily adapted to different requirements. The technical licensing technology applies only
30 to programs supporting the licensing management system, and it is completely ineffective to control usage of different products.

It is an object of the present invention to overcome the above-mentioned drawbacks. In order to achieve this object, a

method as set out in the first claim is proposed.

Briefly, the present invention provides a computer-implemented method of controlling use of digitally encoded products including the steps of: requesting execution
5 of a digitally encoded product on a computer, intercepting the execution request before starting of the product, verifying compliance of the execution request with authorised conditions of use of the product, and enabling or preventing starting of the product according to the result of the verification.

10 Moreover, the present invention also provides a computer program application for performing the method, a program product storing the application, and a corresponding data processing system for controlling use of digitally encoded products.

15 Further features and the advantages of the solution according to the present invention will be made clear by the following description of a preferred embodiment thereof, given purely by way of a non-restrictive indication, with reference to the attached figures, in which:

20 Fig.1 is a basic block diagram of a data processing system in which the method of the invention can be used;

Fig.2 shows a partial content of a working memory of a workstation and of a metering server included in the system;

25 Figg.3a-3c are a flowchart describing the logic of a method for controlling use of software programs on the workstation.

With reference in particular to Fig. 1, a data processing system 100 in which the method of the invention can be used is depicted. The system 100 includes multiple workstations 105w.
30 The workstations 105w are grouped into a series of sub-sets; a metering server 105s is associated with the workstations 105w of each sub-set. The metering server 105s and the corresponding workstations 105w are connected to each other through a network 110 (for example an INTRANET). The different

metering servers 105s communicate with an administrative server 115 (installed at a remote location) through a different network 120 (for example the INTERNET). Moreover, the administrative server 115 is connected to an assets management system 125 (through a further network 127), which stores information relating to the workstations and their users.

Each workstation 105w, for example consisting of a Personal Computer (PC), is formed by several units that are connected in parallel to a communication bus 130w. In detail, a microprocessor (μ P) 135w controls operation of the workstation 105w, a RAM 140w is directly used as a working memory by the microprocessor 135w, and a ROM 145w stores basic programs for a bootstrap of the workstation 105w. Several peripheral units are further connected to the bus 130w (by means of respective interfaces). Particularly, a bulk memory consists of a magnetic hard-disk 150w and a driver 155w for reading CD-ROMs 160w. Moreover, the workstation 105w includes input devices 165w (for example consisting of a keyboard and a mouse), and output devices 170w (for example consisting of a monitor and a printer). A Network Interface Card (NIC) 175w is used to plug the workstation 105w into the network 110.

The metering server 105s (for example consisting of a mid-range computer) is likewise formed by a bus 130s, multiple microprocessors 135s, a RAM 140s, and a ROM 145s; the metering server 105s further includes a hard-disk 150s, a driver 155s for CD-ROMs 160s, input devices 165s, output devices 170s, and a NIC 175s (for accessing the networks 110 and 120).

Similar considerations apply if the system has a different architecture, if the workstations, the metering servers and the administrative server are connected to each other in a different manner, if other networks are employed, if each workstation or metering server has a different structure or includes different components, if no assets

management system is provided, and the like.

Considering now Fig.2, a partial content of the working memories 140w and 140s of the workstation and of the metering server, respectively, is shown. The information (programs and data) is typically stored on the respective hard-disks and loaded (at least partially) into the working memories when the programs are running; the programs are initially installed onto the hard disks from CD-ROMs.

An operating system 205w and an operating system 205s provide a software platform for the workstation and the metering server, respectively. A stack 210w (on the workstation) and a stack 210s (on the metering server) process a set of protocol layers working together for defining network communications. In order to ensure effective transmission of information across the INTERNET, the information is encrypted (so as to ensure security), and it is transmitted using protocols (such as the HTTP) able to cross firewalls.

Considering in particular the workstation, the operating system 205w includes a main module 215 (known as kernel), which provides all the essential services required by other parts of the operating system. A kernel extension module 220 adds functionality to the operating system, using publicly available, standard interfaces.

A runtime agent 225 is running in the background, for controlling execution of application programs 230 on the workstation. The runtime agent 225 interfaces with the stack 210w, in order to exchange information with the metering servers. The runtime agent is silently installed on the workstation during a registration process, in which an end-user of the workstation registers himself/herself by connecting to a web page hosted on the associated metering server.

The runtime agent 225 controls several memory structures. Particularly, a server table 235 includes the IP addresses of

possible alternative metering servers associated with the workstation (in addition to the main one directly connected to the workstation, and identified in a configuration table of the runtime agent). A control catalogue 240 lists the programs
5 whose execution on the workstation must be controlled; on the other hand, an internal log 245 lists the programs not included in the control catalogue 240, but which have been nevertheless executed on the workstation. Moreover, the runtime agent 225 controls entering of information into a
10 running table 250; the running table 250 lists the programs currently in execution on the workstation and for which a corresponding authorisation to run has been granted (together with an identifier of the metering server that has granted the authorisation). The runtime agent 225 also generates a program
15 inventory 255 (for transmission to the metering server), which includes information about the programs installed on the workstation.

Moving now to the metering server, a licensing agent 260 running in the background controls the associated
20 workstations. The licensing agent 260 interfaces with the stack 210s, in order to exchange information with the workstations and the administrative server.

The licensing agent 260 manages an authorisation catalogue 265, which contains information about the authorised
25 conditions of use of each program; for example, the authorisation catalogue 265 specifies the maximum processing power or the maximum capacity of the working memory of the workstations on which the program can run, the number of licenses for the program (defining the maximum number of
30 instances of the program allowed to run concurrently), and the like. In addition, the licensing agent controls entering of information into a metering table 270, which lists the programs currently in execution on the workstations, and whose authorisation to run has been granted by the metering server.

The licensing agent 260 also manages a global catalogue 273 including known programs that could be generally allowed to run on the associated workstations.

Moreover, the licensing agent 260 manages a repository
5 275 that contains different information to be distributed to the associated workstations (such as the up-to-date version of the runtime agent code, or the list of the alternative metering servers and of the programs to be controlled) or collected from the workstations (such as real-time reporting
10 of software usage and inventory information of programs installed on the workstations).

Similar considerations apply if the programs and data are structured in a different manner, if other modules or functions are provided, if the data are stored in equivalent
15 memory structures, if different conditions of use are envisaged, if the information is transmitted using different protocols, if the licensing agent is embedded in a web server, and so on. Alternatively, the runtime agent is installed by a logon script stored on the workstation, or it is deployed by
20 means of a software distribution system.

As shown in Figg.3a-3c, a series of routines, which together make up a method 300, are performed on the workstation and the metering server for controlling use of programs installed on the workstation.

Particularly, the kernel extension module is loaded
25 during the bootstrap of the workstation. The routine implemented by the kernel extension module starts at block 302, and then cycles in an idle loop at block 304 waiting for an event to occur. If a request of execution of a program is detected the blocks 306-308 are executed, if a corresponding
30 response message is returned by the runtime agent the blocks 310-315 are executed, whereas if the termination of a program is detected the block 318 is executed; in every case, the routine then returns to block 304 waiting for a new event to

occur. Conversely, if the workstation has been shut down, the routine ends at the final block 322.

The bootstrap of the workstation also involves the loading of the runtime agent, which implements a routine starting at block 324. The routine cycles in an idle loop at block 326 waiting for an event to occur. If an execution request is notified the blocks 328-334 are executed, if a corresponding response message is received from the metering server the blocks 336-346 are executed, if a program termination is notified the blocks 348-350 are executed, whereas if a time-out has expired the blocks 364-374 are executed; in every case, the routine then returns to block 326 waiting for a new event to occur. Conversely, if the workstation has been shut down, the routine ends at the final block 375.

At the same time, the licensing agent is loaded during the bootstrap of the metering server. The licensing agent implements a routine that starts at block 376, and then cycles in an idle loop at block 378 waiting for an event to occur. If an execution request message is received the blocks 380-386 are executed, if a termination message is received the block 387 is executed, if information is received from a workstation the blocks 388-390 are executed, if a checking message is not received from each associated workstation for a pre-set period of time the blocks 392-393 are executed, whereas if a management message is received from the administrative server the block 394 is executed; in every case, the routine then returns to block 378 waiting for a new event to occur. Conversely, if the metering server has been shut down, the routine ends at the final block 396.

Whenever the end-user requests execution of a program on the workstation (block 397), for example by double clicking with the mouse on a respective icon, the kernel starts a new process. The starting of the process is notified to the kernel

extension module (using a so-called kernel hooking technique), which reports the information to the runtime agent at block 306. The kernel extension module then suspends the process at block 308.

5 In response to the notification, the runtime agent verifies whether the program is included in the control catalogue at block 328. If not, the method passes to block 330, wherein the runtime agent notifies the kernel extension module that the program may be started; the program is then
10 added to the internal log at block 332. In response thereto, the kernel extension module resumes the process associated with the program at block 312, in order to enable the program to start its execution. Conversely, if the program is included in the control catalogue the runtime agent sends a
15 corresponding execution request message to the metering server at block 334 (including configuration information defining the execution environment of the program, such as the processing power or the working memory capacity of the workstation).

 When the licensing agent running on the metering server
20 receives the execution request message, it verifies at block 380 whether execution of the program would comply with the authorised conditions of use stored in the authorisation catalogue (for example whether the execution environment meets the authorised one and the number of instances of the program
25 already running does not reach the maximum value allowed). The licensing agent returns a corresponding response message to the workstation at block 382. A test is then performed at block 384 to determine whether the authorisation to run has been granted. If so, the licensing agent adds the program to
30 the metering table at block 385, and then passes to block 386; conversely, the method descends into block 386 directly. Considering now block 386, the metering server sends an e-mail of notification to a system administrator when a pre-set percentage (such as 80%) of the available licenses for the

product have been granted, and a further e-mail of notification when the metering server has run out of licenses for the product.

As soon as the runtime agent receives the response message, a test is made at block 336 to determine whether the authorisation to run has been granted. If so, the method descends into block 338 (described in the following). On the contrary, the runtime agent verifies at block 340 whether alternative metering servers are listed in the server table. If one or more alternative metering servers are available, the runtime agent sends the execution request message to each alternative metering server in turn, so as to perform the same verification described above; the method then descends into block 338. Conversely, if no alternative metering server is available the method proceeds to block 338 directly.

Considering now block 338, the runtime agent forwards the response message to the kernel extension module and then passes to decision block 344. If the authorisation to run has been granted, the program is added to the running table at block 346 (conversely, the method returns to block 326 directly).

As soon as the kernel extension module receives the response message, a test is carried out at block 310 to determine whether the authorisation to run has been granted. If so, the kernel extension module resumes the process associated with the program at block 312, in order to enable the program to start its execution; conversely, if the authorisation to run has been denied the kernel extension module aborts the process at block 315, so as to prevent starting of the program.

When execution of the program is terminated (block 398), the event is notified to the kernel extension module, which reports the information to the runtime agent at block 318. In response to this notification, the runtime agent checks at

block 348 whether the program is included in the running table. If so, the runtime agent sends a corresponding termination message to the metering server at block 349; passing to block 350, the program is deleted from the running
5 table (conversely, the method returns to block 326 directly). When the licensing agent (on the metering server) receives the termination message, the program is deleted from the metering table at block 387, in order to release the respective license.

10 Whenever the time-out, for example of a few minutes, has expired (block 364), the runtime agent sends a checking message to each metering server that has granted an authorisation to run a program on the workstation (as indicated in the running table). The method then passes to
15 block 366, wherein the runtime agent sends the whole internal log to the associated metering server.

A test is made in decision block 368 to determine whether a longer time-out (for example of some tens of minutes) has expired. If so, the runtime agent scans the hard-disk of the
20 workstation at block 370, in order to collect information about the programs installed thereon. Continuing to block 372, the installation information so collected (stored in the program inventory) is sent to the metering server. The runtime agent then downloads (from the metering server) any upgrade to
25 the server table, to the control catalogue and/or to the code of the runtime agent at block 374, without requiring any user intervention.

Moving now to the licensing agent running on the metering server, any information (such as the internal log or the
30 program inventory) received from an associated workstation is stored at block 388. When the metering server receives an internal log, the method checks at block 389 whether each program listed in the internal log is included in the global catalogue. If so, the method passes to block 390, wherein the

program is added to the control catalogue to be distributed to the workstation and to the authorisation catalogue (with no limit on the corresponding authorised conditions of use); in this way, any next request of execution of the program will be
5 always enabled, so as to allow the usage of the program to be tracked for analysis by the system administrator. Conversely, the method returns to block 378 directly.

10 Considering now block 392, if a checking message is not received from a generic associated workstation for a pre-set period of time (for example ten minutes), all the programs running on this workstation are deleted from the metering table at block 392 (in order to release the respective licenses); a corresponding e-mail of notification is sent to the system administrator at block 393.

15 Moreover, whenever a management message is received from the administrative server, the corresponding actions are performed at block 394. For example, the authorisation catalogue for each metering server may be extracted from a master catalogue stored on the administrative server; each
20 authorisation catalogue is then deployed to the corresponding metering server (thereby reducing the amount of information to be transferred on the network). Moreover, the administrative server may collect information from the metering server (such as about the programs installed and/or running on the
25 corresponding workstations). The administrative server may then create different reports. In detail, the system administrator may request reports on usage of products (filtered by start and end time), on detailed usage of a specific product, on comparative views of installed, used and
30 licensed products. Security profiles are also defined on the administrative server, so that each end-user of the workstations can only access information relating to programs he/she is using. In this way, the administrative server implements a single repository where usage, procurement and

inventory data are collected and managed.

Similar considerations apply if an equivalent method is performed (for example with several concurrent processes that execute the above described operations in parallel), if other
5 functions are provided, if the execution of the program is requested by another program, if the execution request message includes different information, if the time-out values are different or may be customised by the system administrator, if equivalent checking messages are sent to the metering servers
10 periodically (for example including the corresponding records of the running table), if no e-mail of notification is sent to the system administrator, and so on. Alternatively, the proposed method is employed to control use of equivalent
15 digitally encoded products, such as audio and video recordings, electronic books, or multimedia works.

More generally, the present invention provides a computer-implemented method of controlling use of digitally encoded products. The method starts with the step of requesting execution of a digitally encoded product on a
20 computer; the execution request is intercepted before starting of the product. Compliance of the execution request with authorised conditions of use of the product is verified, and starting of the product is enabled or prevented according to the result of the verification.

25 The solution of the invention allows programs running on the computer to be controlled in real-time. In this way, a very effective licensing validation is carried out at low cost.

30 Operation of the proposed method is completely safe, because the program is prevented from starting and it is not killed while already processing data and performing potentially risky operations. Moreover, the integrity of the program is not affected by the denying of the authorisation, so that execution of the program may be readily enabled later

on (without having to re-install its code).

The solution of the present invention provides an integrated solution for controlling use of any type of programs, either technically licensed ones and non-technically
5 licensed ones. Therefore, even programs that do not contain any special logic for enforcing authorised conditions of use may be controlled; at the same time, the proposed method also supports existing solutions exploiting licensing management systems.

10 The preferred embodiment of the invention described above offers further advantages.

Particularly, the execution request is intercepted by detecting the starting of the respective process; the process is suspended, and it is then resumed or aborted later on
15 according to the result of the verification.

This procedure is very simple, but at the same time quite effective.

Advantageously, the operating system notifies the starting of the process to the kernel extension module, which
20 in turn reports the execution request to the runtime agent running on the workstation; at the same time, the kernel extension module suspends the process (waiting for a response to the execution request).

The provision of the kernel extension module and the
25 runtime agent makes the implementation of the solution of the invention platform-independent, and readily portable to any kind of operating system.

In a preferred embodiment of the invention, the execution request is processed by a metering server, which controls
30 multiple workstations associated therewith.

The proposed architecture allows a very flexible management of the licenses. For example, concurrent use of a maximum number of instances of each program on the workstations may be controlled from a single point in a very

simple manner.

Advantageously, if the associated server has already granted all the available licenses, the workstation is allowed to contact alternative metering servers.

5 Therefore, the total number of licenses available for each product may be shared among all the workstations of the organisation (even if they are controlled by different metering servers).

10 However, the solution of the invention leads itself to be carried out also intercepting the execution request in a different manner (for example monitoring any end-user action), employing only the kernel extension module or the runtime agent, processing the execution request by the workstation directly (without any metering server), and even not allowing
15 the workstations to contact any alternative metering server.

Preferably, the control catalogue lists the programs to be metered (execution of the other programs is not affected by the envisaged method).

20 This feature makes it possible to implement customised licensing policies by the system administrator.

The programs not included in the control catalogue are always allowed to run; however, they are added to the internal log, which is periodically sent to the metering server (for updating the authorisation catalogue and the control catalogue
25 for the workstation).

The internal logs provided by the different workstations allow usage of any new product to be readily tracked (with unlicensed programs that are reported to the metering server as soon as they are used).

30 Alternatively, no control catalogue is provided (with all the programs that are always controlled), the running of programs not included in the control catalogue is notified to the metering server immediately, the authorisation and control catalogues are updated in a different manner according to the

internal log (even without using any global catalogue), or only the programs listed in the control catalogue may be executed on the workstation (provided that they are authorised to).

5 In addition, when the program ends its execution a termination message is sent from the workstation to the metering server that has granted the respective authorisation to run; the program is then deleted from the running table (on the workstation) and the metering table (on the metering
10 server).

 This procedure ensures direct release of the corresponding license, as soon as the program ends its execution; the license is then immediately available for use by other workstations.

15 Advantageously, each workstation periodically sends a checking message to the respective metering servers that have granted the authorisations to run; if the checking message from a workstation is not received for a pre-set period of time, the metering server automatically releases the licences
20 used by the workstation.

 In this way, if a workstation cannot release a license, for example because of a network problem, the license is released automatically on the metering server.

 However, the solution of the invention is also suitable
25 to be implemented detecting the termination of each program in a different manner (for example with a polling procedure under the control of each metering server), and even without any periodic transmission of the checking message (with the licenses that are automatically released after a pre-set
30 period of time, such as every night).

 Advantageously, each workstation periodically collects installation information, which is then sent to the associated metering server.

 The envisaged feature makes it possible to track and keep

watch of products installed on each workstation (even if they are not used).

In a preferred embodiment of the invention, a single administrative server centrally controls multiple metering
5 servers.

The proposed three-tier architecture (workstations, metering servers and administrative server) enables a very flexible management of the licenses from a centralised point.

For example, the administrative server collects
10 information from all the metering servers, in order to provide historical reporting concerning program usage and inventory information for all the workstations. In addition, it provides an administrative interface to maintain licensing information from a centralised point.

Therefore, it is possible to save money by detecting over
15 compliance conditions (wherein too many licenses are available for unused products), or it is possible to detect potential infringement conditions in advance (in order to take corrective actions before running out of licenses).

The proposed architecture features a high scalability.
20 For example, in a small company only one metering server and one administrative server are needed (even installed on the same computer). Conversely, in a large company, multiple metering servers (controlled by a single administrative
25 server) may be provided in one or more networks. On the other hand, the administrative server may be installed in a remote location and may control metering servers at different facilities (for example in an organisation that offers software management services to other companies).

Alternatively, the installation information is collected
30 on request, the proposed method only controls execution of the programs (and not their installation), or the system has a different architecture (even without any administrative server).

Advantageously, the solution according to the present invention is implemented with a computer program application, which is provided on CD-ROM. The application consists of programs installed on each workstation and on the associated metering sever.

Alternatively, the programs are provided on floppy-disks, are pre-loaded onto the hard-disks, or are stored on any other computer readable medium, are sent to the workstation and the metering server through a network (typically the INTERNET), are broadcast, or more generally are provided in any other form directly loadable into their working memories. However, the method according to the present invention leads itself to be carried out even with a hardware structure, for example integrated in chips of semiconductor material.

Naturally, in order to satisfy local and specific requirements, a person skilled in the art may apply to the solution described above many modifications and alterations all of which, however, are included within the scope of protection of the invention as defined by the following claims.

CLAIMS

1. A computer-implemented method (300) of controlling use of digitally encoded products including the steps of:

5 requesting (397) execution of a digitally encoded product on a computer,
intercepting (306) the execution request before starting of the product,
10 verifying (334,380-385,336-342) compliance of the execution request with authorised conditions of use of the product, and
enabling or preventing (310-315) starting of the product according to the result of the verification.

2. The method (300) according to claim 1, further including the steps of:

15 detecting (306) the starting of a process associated with the execution request,
suspending (308) the process, and
resuming or aborting (310-315) the process according to the result of the verification.

20 3. The method (300) according to claim 2, further including the steps of:

notifying (397) the starting of the process to a kernel extension module of an operating system controlling operation of the computer,
25 the kernel extension module reporting (306) the execution request to an agent running on the computer and suspending (308) the process in response to the notification, and
the agent causing (338) the kernel extension module to resume or abort the process according to the result of the
30 verification.

4. The method (300) according to any claim from 1 to 3, wherein the step of verifying (334,380-385,336-342) compliance

of the execution request includes:

sending (334) a request message indicative of the execution request from the computer to a metering server associated with a plurality of the computers,

5 granting or denying (380) an authorisation to run the product by the metering server according to a comparison between licensing information indicative of the authorised conditions of use for all the associated computers and a metering memory structure indicative of the authorisations to run currently granted by the metering server,

10 sending (382) a corresponding response message from the metering server to the computer, and

adding (384-385) the product to the metering memory structure when the authorisation to run has been granted.

15 5. The method (300) according to claim 4, further including the steps of:

sending (336,340,342) the request message from the computer to at least one further metering server in response to the denying of the authorisation to run.

20 6. The method (300) according to any claim from 1 to 5, wherein the computer stores a control memory structure indicative of the products to be controlled, the method further including the step of controlling (328) inclusion of the product in the control memory structure, the steps of
25 verifying (334,380-385,336-342) compliance of the execution request and of enabling or preventing (310-315) starting of the product being executed only if the product is included in the control memory structure.

30 7. The method (300) according to claim 6, further including the steps of:

enabling (330) starting of a product not included in the control memory structure,

adding (332) an indication of the product not included in the control memory structure to a log stored on the computer,

periodically (366) transmitting the log to the associated metering server, and

updating (389,390) the control memory structure and the licensing information according to the log.

5 8. The method (300) according to any claim from 4 to 7, further including the steps of:

terminating (398) execution of the product on the computer,

intercepting (318) the execution termination,

10 sending (349) a termination message indicative of the execution termination from the computer to a current metering server which has granted the authorisation to run, and

removing (387) the product from the metering memory structure in response to the termination message.

15 9. The method (300) according to claim 8, further including the steps of:

periodically (364) sending a checking message from the computer to each corresponding current metering server, and

20 removing (392) each product running on the computer from the metering memory structure stored on the corresponding current metering server when the checking message is not received for a pre-set period of time.

10. The method (300) according to any claim from 4 to 9, further including the steps of:

25 periodically collecting (368,370) installation information indicative of the products installed on the computer, and

sending (372) the installation information from the computer to the corresponding metering server.

30 11. The method (300) according to any claim from 4 to 10, further including the step of controlling (394) a plurality of metering servers from a single central administrative server.

12. A computer program application (220,225,260) directly loadable into a working memory of a data processing system

(100) for performing the method of any claim from 1 to 11 when the program application is run on the data processing system.

13. A program product (160w,160s) comprising a computer readable medium on which the program application of claim 12 is stored.

14. A data processing system (100) for controlling use of digitally encoded products including means (205w) for requesting execution of a digitally encoded product on a computer (105w), means (220,225) for intercepting the execution request before starting of the product, means (225,260) for verifying compliance of the execution request with authorised conditions of use of the product, and means (225,220) for enabling or preventing starting of the product according to the result of the verification.

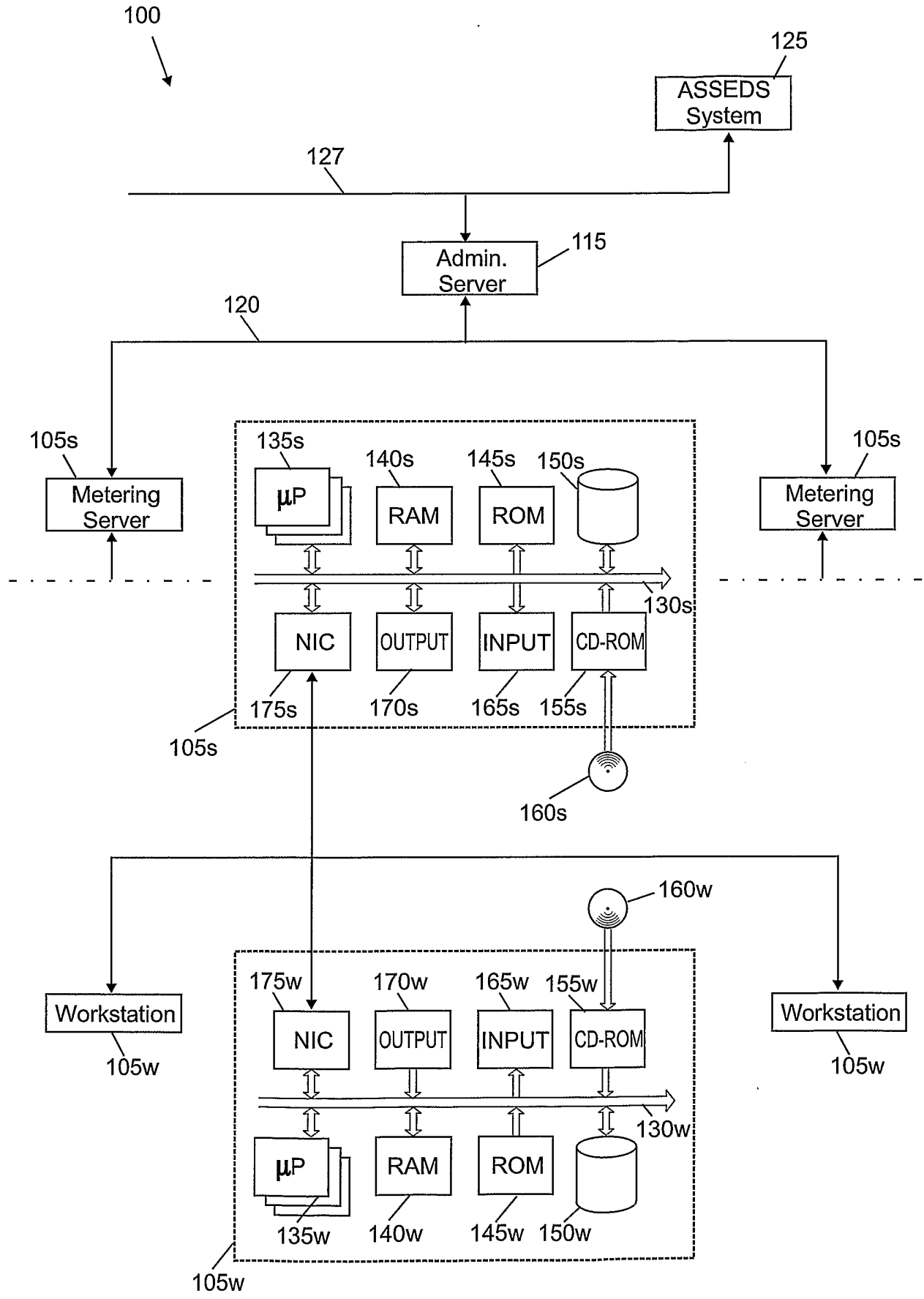


FIG. 1

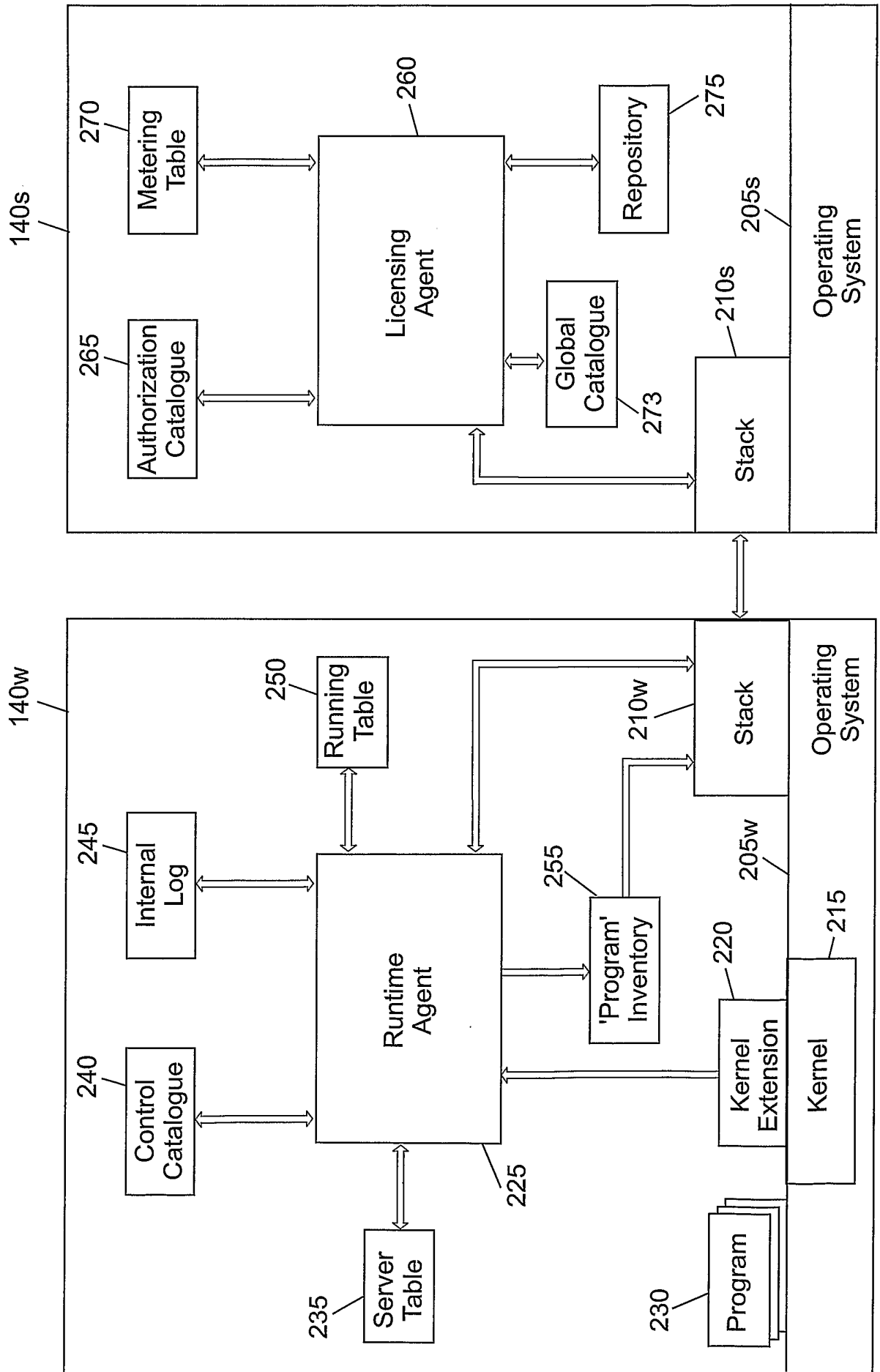


FIG. 2

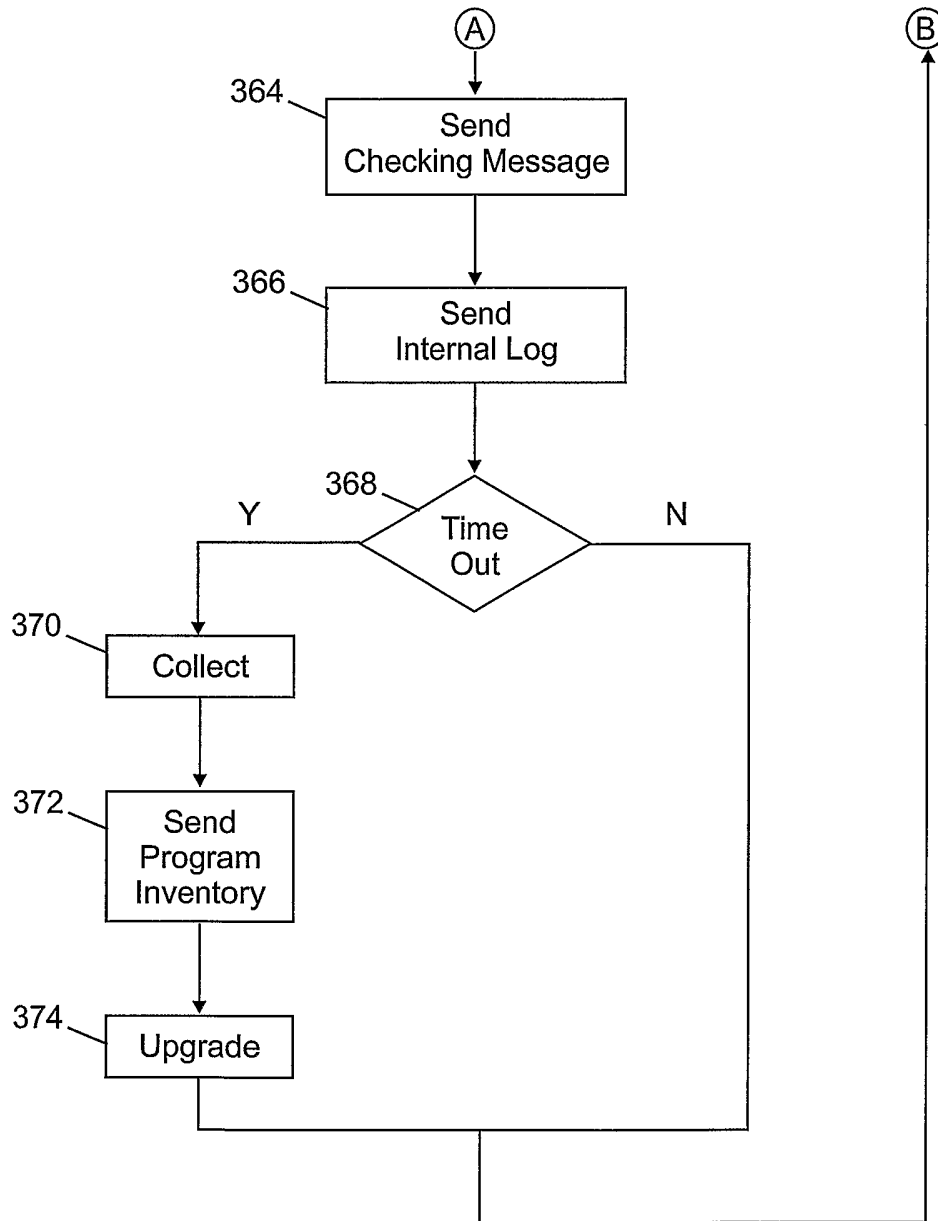


FIG. 3b

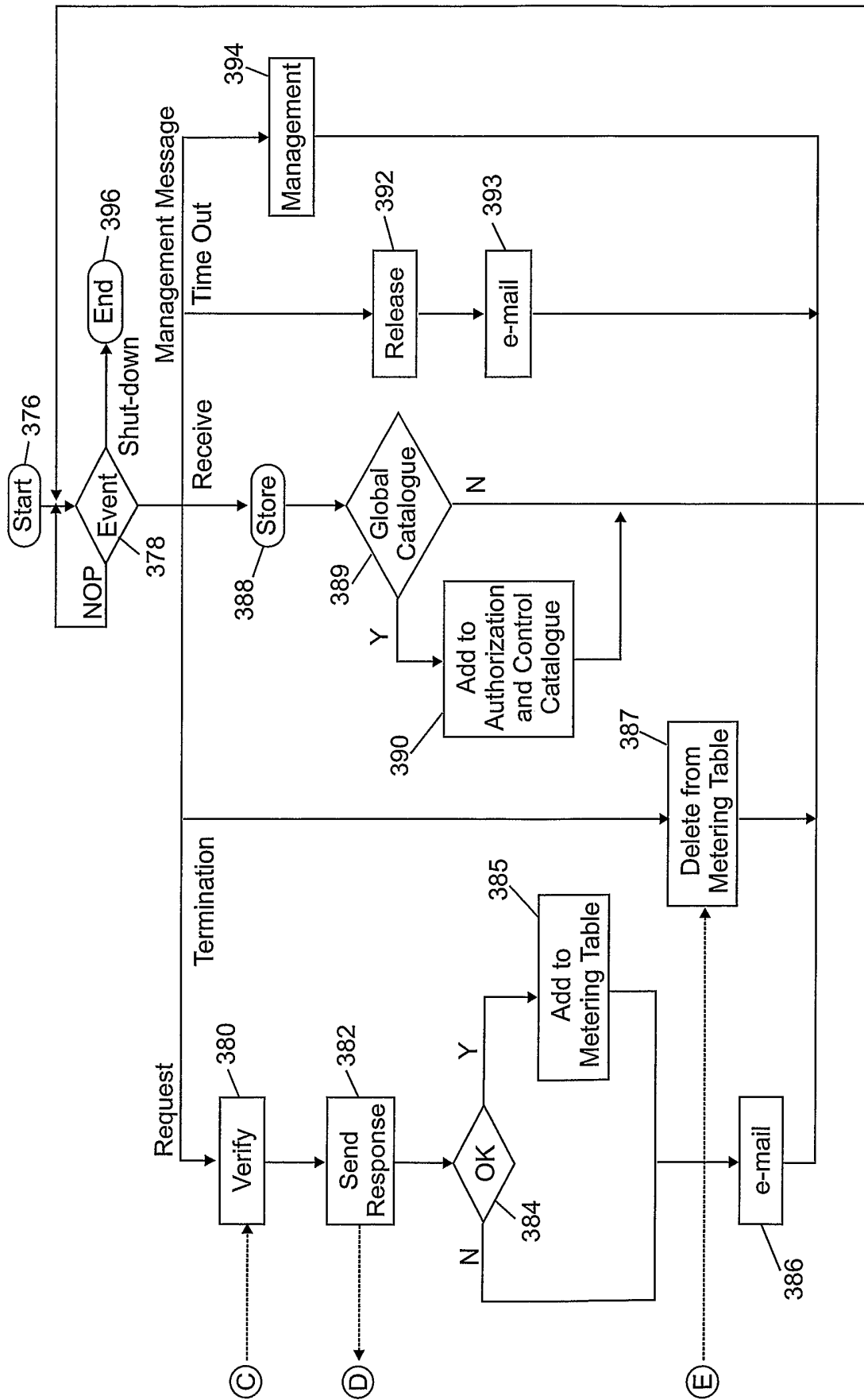


FIG. 3c