(19) Europäisches Patentamt
European Patent Office
Office européen des brevets

(11) **EP 3 762 842 B1**

(12) **EUROPEAN PATENT SPECIFICATION**

(45) Date of publication and mention of the grant of the patent:
10.08.2022 Bulletin 2022/32

(21) Application number: **19710946.5**

(22) Date of filing: **01.03.2019**

(51) International Patent Classification (IPC):
G06F 21/10 *(2013.01)* ***G06F 9/455*** *(2018.01)*

(52) Cooperative Patent Classification (CPC):
G06F 21/105; G06F 9/455; H04L 2463/103

(86) International application number:
PCT/US2019/020174

(87) International publication number:
WO 2019/173114 (12.09.2019 Gazette 2019/37)

(54) **ACTIVATION OF AN APPLICATION BASED ON PRIOR ACTIVATION OF AN ISOLATED COUNTERPART APPLICATION**

AKTIVIERUNG EINER ANWENDUNG AUF BASIS DER VORHERIGEN AKTIVIERUNG EINER ISOLIERTEN GEGENANWENDUNG

ACTIVATION D'UNE APPLICATION BASÉE SUR L'ACTIVATION PRÉALABLE D'UNE APPLICATION HOMOLOGUE ISOLÉE

(84) Designated Contracting States:
AL AT BE BG CH CY CZ DE DK EE ES FI FR GB GR HR HU IE IS IT LI LT LU LV MC MK MT NL NO PL PT RO RS SE SI SK SM TR

(30) Priority: **08.03.2018 US 201815916146**

(43) Date of publication of application:
13.01.2021 Bulletin 2021/02

(73) Proprietor: **Microsoft Technology Licensing, LLC**
Redmond, WA 98052-6399 (US)

(72) Inventors:
- **RAMANUJAM, Sinduja**
  Redmond, Washington 98052-6399 (US)
- **GUTIERREZ OLIVO, Axel Alejandro**
  Redmond, Washington 98052-6399 (US)
- **PEZA RAMIREZ, Carlos E.**
  Redmond, Washington 98052-6399 (US)
- **JAIN, Amit**
  Redmond, Washington 98052-6399 (US)
- **SCRIPNICIUC, Andrei**
  Redmond, Washington 98052-6399 (US)
- **RISCUTIA, Vlad**
  Redmond, Washington 98052-6399 (US)

(74) Representative: **Grünecker Patent- und Rechtsanwälte**
PartG mbB
Leopoldstraße 4
80802 München (DE)

(56) References cited:
US-A1- 2005 132 347 **US-B1- 8 763 159**

EP 3 762 842 B1

Description

BACKGROUND

[0001] Some computing systems are designed to run an application in a manner so as to isolate the application from one or more counterpart applications. For example, the application may be designed to run within a container that is initiated by an operating system (OS) when a user instructs the OS to launch the application. Although a container may be a useful tool to ensure that the application operates reliably on any given computing platform, the container can present challenges in relation to permitting the application to securely communicate with other applications that are running outside of the container. In particular, the OS will typically impose numerous limitations on how an application that is running within a particular container is permitted to communicate with other applications that are designed to run external to the particular container.

[0002] Unfortunately, communication barriers that are imposed with respect to containers may prevent even some desirable communications from being securely transmitted between an application that is running within the container and a counterpart application(s) that is designed to run external to the container. For example, under many circumstances an application that is running within a container may be unable to directly communicate with, or even securely write information to, a storage location that is accessible by the counterpart application. Thus, the counterpart application(s) may remain unapprised as to when certain activities are performed with respect to an application while it is running within the container, even when those activities are directly germane to the governance of certain behaviors and/or functionalities of the counterpart application(s).

[0003] It is with respect to these and other considerations that the disclosure made herein is presented.

[0004] US 8 763 159 B1 relates to a system and a method for managing licensing of virtual environment applications. US 8 763 159 B1 describes that a licensing module of a first installed container application detects installation of affiliated applications and gives them a group licensing key for passing it to the licensing server. The licensing server then derives licensing parameters of the affiliated applications from the group key and gives the licenses to the affiliated applications in case of successful validation. Like this, the licensing system provides protection from unauthorized copying of the applications.

[0005] It is the object of the present invention to provide systems and methods for jointly activating isolated counterpart applications in a secure and efficient manner and thus avoiding computationally burdensome redundant activation prompts.

[0006] This object is solved by the subject matter of the independent claims.

[0007] Preferred embodiments are defined in the dependent claims.

SUMMARY

[0008] Technologies described herein enable a system to securely activate an application on a computing device based on a prior activation of an isolated (e.g., containerized) counterpart application on the computing device. Embodiments disclosed herein enable an application that is restricted from directly communicating with a counterpart application (e.g., due to containerization) to share license data locally to communicate instead with an application license manager (that is external to the computing device) to manage license data corresponding to the counterpart application. For example, in a scenario where a container application and a native application are installed on a particular computing device, an operating system (OS) of the computing device may restrict communications to and from the container application so that a license that is obtained by the container application cannot be directly shared with the native application. According to the techniques described herein, when a license is obtained at the computing device for the container application, the application license manager is caused to update license data in association with the native application so that a counterpart license can be obtained automatically by the native application, and vice versa. In this way, once a user has positively responded to an activation prompt that is presented by either one of the container application or the native application, whichever application the user has not manually activated by providing user input will be automatically activated at the computing device based on communications with the application license manager.

[0009] The disclosed technologies represent a substantial advantage over existing application activation systems which can jointly activate counterpart applications only under circumstances where the counterpart applications can directly communicate with one another. For example, some modern application suites are configured to locally store a license to share licensing statuses between multiple applications. When a particular application is activated on a computing device, a license obtained during that activation is locally stored in a shared location (e.g., Registry Keys) to activate all of the counterpart applications of the application suite. However, under circumstances where a counterpart application is isolated (e.g., containerized) from the particular application through which the license was obtained, existing application activation systems lack functionality for securely sharing the license with the "isolated" counterpart application. The unfortunate result of this limitation is that redundant activation prompts may be exposed via counterpart application(s) even after a user has responded to a previous activation prompt. It can be appreciated that redundant computing processes (e.g., exposing redundant activation prompts) result in exorbitant consumption of computing resources. For example, by causing a user

to progress through redundant activation sequences for counterpart products, the processing units of a computing device are caused to undergo unnecessarily high numbers of processing cycles. Additionally, due to the user having to re-enter much of the same information (e.g., credentials, user names, passwords, etc.) in a redundant activation prompt as was previously entered in an initial activation prompt, it can further be appreciated that redundant activation prompts also unnecessarily increase networking traffic.

[0010] In an exemplary implementation, a computing device has installed thereon a native application and a container application that an operating system (OS) of the computing device at least partially restricts communications between. In particular, when the container application is launched, the OS initiates the container application within a container that restricts communications between it and other applications that are designed to run natively on the computing device. Exemplary containers include, but are not limited to, LINUX containers, MICROSOFT HYPER-V containers, WINDOWS SERVER containers, and/or any other operating-system-level virtualization method. The native application may be configured to be run by an operating system (OS) without containerization layers that are associated with the container application. By virtue of the container, the container application may be restricted from passing information to the native application, and vice versa. For example, if either one of the native application or the container application have been activated on the computing device by storing a license locally thereon, the container may prevent that license from being shared locally for use by both the container application and the native application.

[0011] In this exemplary implementation, the native application is a counterpart to the container application in the sense that an activation of either one of these applications on the computing device is also applicable to the other application. For example, the container application and the native application may be commonly associated with an application suite for which a single license is usable by multiple applications. As a specific but non-limiting example, the container application may be an application management portal (e.g., MY OFFICE by MICROSOFT) for managing subscriptions for and/or viewing recent activity of a plurality of different native applications (e.g., MICROSOFT VISIO, MICROSOFT WORD, MICROSOFT POWERPOINT, etc.) that are installed and/or are available for installation on the computing device. However, due to the container restricting communications between the native application and the container application, it may not be feasible to locally pass a license that is received by the container application to the native application(s) for activation on the computing device.

[0012] For example, upon a user launching the container application, the container application determines that it is not currently activated on the computing device and responds by exposing an activation prompt. The activation prompt may, for example, be designed to inform the user that the container application is not currently activated and to prompt the user to enter user credentials and/or a product key to activate the container application. Based on user input received via the activation prompt, the container application transmits an activation request to an application license manager with which both the container application and the native application can communicate. In this example, the activation request includes a machine identifier (ID) that uniquely identifies the computing device. The activation request may further include an activation scope that indicates whether the activation request applies to the native application.

[0013] As a specific but non-limiting example, the computing device may have installed thereon the container application as well as a first native application through an N^{th} native application. In this example, the activation request may define a subset of these applications as, for example, the first native application, a second native application, and the container application, to enable activation of this subset of applications on the computing device. Based on the machine ID and the activation scope that are provided in the activation request, the application license manager may update license data for the subset of applications in association with the machine ID. Thus, in this specific example, the application license manager will update the license data to designate all of the first application, the second native application, and the container application for activation on the computing device.

[0014] In some embodiments, the application license manager may provide, to the container application, product activation data that includes a container license for activation of the container application on the computing device. Then, the container application may store the container license in a "container available" portion of the local storage that is available to the container application but is not available to the native application(s). The container available portion of the local storage may be a virtual registry that is accessible only by the container application and/or other applications that are designed to run within the container. Accordingly, upon a user closing down and then re-launching the container application, the container license can be obtained by the container application (e.g., to unlock various functionalities of the container application). In contrast, when the native application is launched it will not be able to obtain the container license despite the container license being stored locally on the computing device.

[0015] Rather than exposing a redundant activation prompt to the user due to the local unavailability of a license, the native application may transmit an activation inquiry to the application license manager to determine whether it has been designated for activation on the computing device. For example, after having already deployed the container application to transmit the activation request, the user may instruct the OS to launch the native application. Then, the native application may determine that a native license is not locally available and, based

thereon, transmit the activation inquiry to the application license manager. Responsive to the activation inquiry, the application license manager examines the license data and determines that the native application has been designated for activation on the computing device. Accordingly, the application license manager returns counterpart product activation data for activation of the native application on the computing device. The counterpart product activation data may include a native license for the native application to store in a "native available" portion of the local storage that is available to the native application(s) but is not available to the container application.

[0016] Thus, once the counterpart product activation data is received by the native application, both of the container application and the native application(s) may be activated on the computing device based on the single activation request that is sent via the container application despite the communication barriers presented by the container. In this way, a user can seamlessly cause activation of both container application(s) and native application(s) by positively responding to a single activation prompt. It can be appreciated that the techniques disclosed herein provide benefits over existing application activation techniques which would require the user to respond to separate activation prompts in association with each of the container application and at least one native application since the container prevents these applications from securely sharing license information locally on the computing device.

[0017] It should be appreciated that any reference to "first," "second," etc. items and/or abstract concepts within the description is not intended to and should not be construed to necessarily correspond to any reference of "first," "second," etc. elements of the claims. In particular, within this Summary and/or the following Detailed Description, items and/or abstract concepts such as, for example, individual applications and/or activation inquiries and/or licenses may be distinguished by numerical designations without such designations corresponding to the claims or even other paragraphs of the Summary and/or Detailed Description. For example, any designation of a "first activation inquiry" and "second activation inquiry" within a paragraph of this disclosure is used solely to distinguish two different activation inquiries within that specific paragraph - not any other paragraph and particularly not the claims.

[0018] It should be appreciated that the above-described subject matter may also be implemented as a computer-controlled apparatus, a computer process, a computing system, or as an article of manufacture such as a computer-readable medium. These and various other features will be apparent from a reading of the following Detailed Description and a review of the associated drawings.

[0019] This Summary is provided to introduce a selection of concepts in a simplified form that are further described below in the Detailed Description. This Summary is not intended to identify key features or essential features of the claimed subject matter, nor is it intended that this Summary be used to limit the scope of the claimed subject matter. Furthermore, the claimed subject matter is not limited to implementations that solve any or all disadvantages noted in any part of this disclosure.

DRAWINGS

[0020] The Detailed Description is described with reference to the accompanying figures. In the figures, the left-most digit(s) of a reference number identifies the figure in which the reference number first appears. The same reference numbers in different figures indicate similar or identical items. References made to individual items of a plurality of items can use a reference number with another number included within a parenthetical to refer to each individual item. Generic references to the items may use the specific reference number without the sequence of letters.

FIG. 1 illustrates an exemplary system for activating an application on a computing device based on a prior activation of an at least partially isolated counterpart application on the computing device.

FIG. 2 illustrates an exemplary system in which a user instructs the OS to launch a first application which then responds by checking for a license both at the local storage and with the application license manager prior to exposing an activation prompt.

FIG. 3 illustrates an exemplary system in which a user activates the first application which causes the application license manager to provide a license to the first application and also to update the license data to designate a second application for automatic activation at the computing device.

FIG. 4 illustrates an exemplary system in which a user instructs the OS to launch the second application after the license data has been updated as descried in FIG. 3 which results in the second application being automatically activated at the computing device.

FIG. 5 is a flow diagram of an example method for an application to perform upon being launched to check both locally for an existing license and remotely for an available license prior to exposing an activation prompt to a user.

FIG. 6 is a flow diagram of an example method for an application license manager to perform to activate multiple applications on a computing device based on an activation request received from an individual one of the multiple applications.

FIG. 7 shows additional details of an example computer architecture for a computer capable of executing the systems and methods described herein.

DETAILED DESCRIPTION

[0021] The following Detailed Description describes technologies that enable a system to securely activate one or more applications on a computing device based on a prior activation of an isolated (e.g., containerized) counterpart application on the computing device. An exemplary embodiment of the system described herein enables a user to provide user input in association with an activation prompt of a container application that is installed on a computing device. Aspects of the user input are used by the container application to generate an activation request that is sent from the computing device to an application license manager. The activation request may include a machine identifier (ID) that uniquely identifies the computing device. The activation request may also include an activation scope which indicates which application(s) on the computing device the activation request applies to. For example, the activation scope might indicate that the activation request is applicable to both the container application as well as a native application that is a counterpart to the container application. Responsive to the activation request, the application license manager may return, to the container application, product activation data that includes a license that is usable for activating the container application on the computing device. Upon receipt of the license, the container application may store the license locally so that each time a user closes and re-launches the container application the license can be read to determine the activation status of the container application.

[0022] Subsequent to having activated the container application, the user may launch the native application that the activation request which was previously sent by the container application is also applicable to. For example, the container application and the native application may be "counterparts" in the sense that an activation of either one is also applicable to the other. However, by virtue of the container application being designed to run within a container, an operating system (OS) of the computing device may impose restrictions on communications between the container application and the native application. Thus, even though the container application may have already received the license at the computing device, it may be impractical for this license to be locally shared between the container application and the native application.

[0023] Notwithstanding the native application being unable to locally communicate with the container application so as to obtain the license, the native application may refrain from exposing an activation prompt to the user. Rather, the native application may transmit to the application license manager an activation inquiry that includes the machine ID to cause the application license manager to determine whether the native application has been designated for activation on the computing device. For example, upon receipt of the activation request from the container application, the application licensing manager may update license data to indicate which application(s) on the computing device are to be automatically activated upon request. Because in the current example the activation scope indicated that the activation request was also applicable to the native application, the application license manager responds to the activation inquiry received from the native application with a license that is usable to activate the native application on the computing device. In this way, once a user has positively responded to an activation prompt that is presented by the container application, the native application (which the user has not manually activated by providing user input) will be automatically activated at the computing device based on communications with the application license manager.

[0024] The present disclosure is believed to be applicable to a variety of systems and approaches involving jointly licensing and/or activating two or more applications which are at least partially isolated from one another on a computing device. Aspects of the present disclosure are predominantly described in the context of an exemplary implementation in which an activation request is sent by a container application and wherein the activation request defines an activation scope that covers a native application that is installed on the same computing device as the container application. While the present disclosure is not necessarily limited to this exemplary implementation, an appreciation of various aspects of the disclosed techniques is best gained through a discussion of such an exemplary implementation. It can be appreciated that the disclosed techniques are also applicable to other implementations in which a native application sends an activation request defining an activation scope as including a container application. For example, a user manually activating a native application may result in an automatic activation of a container application, and vice versa.

[0025] Turning now to FIG. 1, illustrated is an exemplary system 100 for activating an application on a computing device 106 based on a prior activation of an at least partially isolated counterpart application on the computing device 106. More specifically, in the illustrated example, one or more of native applications 108 are automatically activated on the computing device 106 based on an activation request 126 that is generated by a container application 112 with which the one or more native applications 108 are at least partially restricted from communicating.

[0026] In the illustrated example, the computing device 106 is shown to include a first native application 108(1) through an N^{th} native application 108(N). As used herein, the term "native application" may refer generally to any computing application that is configured to run by accessing one or more native features of an operating system (OS) 116 of the computing device 106. Exemplary native applications include, but are not limited to, WIN32 applications that are capable of accessing a full set of WINDOWS application programing interfaces (APIs). It can be appreciated, therefore, that under various exemplary

circumstances desktop versions of MICROSOFT VISIO, MICROSOFT WORD, MICROSOFT POWERPOINT, ADOBE ACROBAT, and/or ABOBE PHOTOSHOP may aptly be described as native applications.

[0027] As further illustrated, the computing device 106 includes a container application 112 that is designed to be run by the OS 116 in a container 110. As used herein, the term "container application" may refer generally to any computing application that is configured to be run within an isolated (e.g., contained) computing environment such as, for example, a LINUX container, a MICROSOFT HYPER-V container, a WINDOWS SERVER container, and/or any other isolated computing environment that is facilitated via operating-system-level virtualization methods. The computing device 106 may further include a communication channel 114 that enables at least some communications between the OS 116 and the container application 112. In some embodiments, the communication channel 114 may be a virtual socket channel such as, for example, a HYPERVISOR socket channel.

[0028] In the illustrated example, the computing device 106 further includes a local storage 118. Exemplary local storage 118 may include, but is not limited to, SATA-type solid-state hard drives, SATA-type hard disks, PATA-type solid-state hard drives, PATA-type hard disks, USB "flash" drives, SD non-volatile memory cards, and/or any other suitable component for providing non-volatile computer-readable media. As illustrated, the local storage 118 may include a "native available" portion that is accessible by the native applications 108 but is inaccessible to the container application 112. As further illustrated, the local storage 118 may include a "container available" portion that is accessible by the container application 112 but is inaccessible to the native application 108. In some instances, the local storage 118 may further include a "commonly available" portion that is accessible by both of the native applications 108 and the container application 112.

[0029] The local storage 118 may include a machine ID 120 that uniquely identifies the computing device 106. In some implementations, the machine ID 120 may be stored in the "commonly available" portion of the local storage 118. Additionally, or alternatively, the machine ID may be stored in one or both of the "natively available" portion and/or the "container available" portion of the local storage 118.

[0030] With respect to the data flow scenario of FIG. 1, upon a user instructing the OS 116 to launch the container application 112, a first activation inquiry 122(1) may be generated and transmitted from the container 110 to an application license manager 102. As illustrated, the container 110 and the container application 112 therein reside in a local data layer in the sense that the container 110 is locally executed at the computing device 106. As further illustrated, the application license manager 102 resides in a cloud data layer in the sense that the application license manager 102 is implemented as a web service and is executed externally from the computing device 106.

[0031] In some implementations, prior to sending the first activation inquiry 122, the container application 112 is configured to check the local storage 118 to determine whether a container license 132 has been stored locally in order to activate the container application 112 on the computing device 106. Then, if the container license 132 is available on the local storage 118, the container application 112 may recognize that it has been activated on the computing device 106 and operate in a standard "activated" mode (e.g., by enabling a user to utilize various functionalities of the container application). However, if the container license 132 is not available on the local storage 118, then the container application 112 may recognize that it is not currently activated on the computing device 106.

[0032] It can be appreciated that according to conventional application activation techniques, upon a user instructing an OS to launch an application that is not currently activated on a computing device 106 due to a license not being available to the application, that application would typically launch an activation prompt user interface (UI) to inform the user that the application has not been activated and/or to prompt the user to activate the application. According to implementations of the techniques disclosed herein, such an application will instead transmit an activation inquiry 122 to the application license manager 102 to inquire as to whether the application has been designated for automatic activation (e.g., activation without exposure of an activation prompt).

[0033] In the illustrated implementation, upon determining that there is no container license 132 available on the local storage 118, the container application 112 responds by transmitting the first activation inquiry 122(1) to the application license manager 102. The first activation inquiry 122(1) includes the machine ID 120 to enable the activation license manager 102 to determine whether the container application 112 has been designated to be provided with a license at the computing device 106. For example, the application license manager 102 may examine license data 104 and determine that the container application 104 has been designated to receive a license. It is worth noting that the specific details of the license data 104 shown in FIG. 1 do not reflect the container application 112 not being designated to receive a license; rather, the specific details shown reflect the license data 104 as updated based on the activation request 126 discussed below. Therefore, in response to the first activation inquiry 122(1), the application license manager 102 transmits license status data 124 to the container application 112 which indicates that a license is not currently available for the container application 112.

[0034] The container application 112 may then transmit to the application license manager 102 an activation request 126 that includes the machine ID 120 and/or an activation scope 128 that indicates various details of what type of activation is being requested. Based on the activation request 126, the application license manager 102

may be caused to update the license data 104. For example, in the illustrated scenario, the activation request 126 causes the application license manager 102 to update the license data 104 in association with the machine ID 120 to specifically indicate that the license status for the machine ID is "Activated" in accordance with a license type of "Enterprise Class 1" and a license term that "Expires 1/1/2020." As further illustrated, the license data 104 has been updated to indicate that the activation scope 128 includes the first native application 108(1), the second native application 108(2), and the container application 112 but excludes the N^{th} native application 108(N).

[0035] In response to the activation request 126, the application license manager 102 provides product activation data 130 to the container application 112. As indicated by the lock symbol of the container application being toggled from "locked" to "unlocked," the product activation data 130 is usable to activate the container application 112 on the computing device 106. For example, the product activation data 130 may include a container license 132 that is stored by the container application 112 within the container available portion of the local storage 118. The container available portion of the local storage 118 may be a virtual registry and/or virtual storage that is accessible only by the container application 112 but not the native applications 108. Accordingly, upon a user closing down and then re-launching the container application 112, the container license 132 can be re-obtained by the container application 112 to determine whether it is appropriate to unlock various functionalities of the container application 112.

[0036] As described above, the OS 116 may impose strict limitations on an ability of the native application 108 to communicate with the container application 112, and vice versa. Accordingly, the native application(s) 108 may be unable to obtain the container license 132 locally or otherwise locally communicate with the container application 112 to determine that an activation has occurred which is applicable to the native application(s) 108. Thus, in the illustrated scenario, a second activation inquiry 122(2) may be transmitted to the application license manager 102 for determining whether the second native application 108(2) (and/or other native applications 108 for that matter) have been designated within the license data 104 for automatic activation. For example, responsive to a user instructing the OS 116 to launch the second native application 108(2), the second native application 108(2) may examine the local storage 118 and determine that a native license 136 is not locally available. Based on this determination, the second native application 108(2) may transmit the second activation inquiry 122(2) to the application license manager 102. Similar to the first activation inquiry 122(1), the second activation inquiry 122(2) also includes the machine ID 120 to enable the application license manager 102 to identify which particular computing device the second activation inquiry 122(2) corresponds to.

[0037] Upon receipt of the second activation inquiry 122(2), the application license manager 102 examines the license data 104 to determine whether the second native application 108(2) has been designated for automatic activation at the computing device 106. As illustrated, following receipt of the activation request 126, the application license manager 102 has updated the license data 104 to indicate that a license has been activated with respect to the computing device 106 as identified by the machine ID 120. The "updated" license data 104 further indicates that the license is applicable to second native application 108(2). Therefore, the application license manager 102 responds to the second activation inquiry 122(2) with counterpart product activation data 134.

[0038] As indicated by the lock symbol of the first native application 108(1) and the second native application 108(2) being toggled from "locked" to "unlocked," the counterpart product activation data 134 is usable to activate these two native applications 108 on the computing device 106. For example, the counterpart product activation data 134 may include a native license 136 which may be stored by the second native application 108(2) within the native available portion of the local storage 118. In some implementations, the native license 136 is a duplicate instance of the container license 132. Stated plainly, the native license 136 may be identical or substantially identical to the container license 132. The native available portion of the local storage 118 may correspond to Registry Keys and/or other configuration databases that are suitable for storing encrypted and/or unencrypted product licenses.

[0039] Once the counterpart product activation data 134 is obtained by the second native application 108(2), any native application(s) 108 for which the counterpart product activation data 134 is applicable may be activated on the computing device 106 without any subsequent activation prompt being shown to the user in association with these native application(s) 108. In this way, one or more processing units of the computing device 106 may be spared from having to perform redundant processing cycles which would otherwise be required to re-collect user credentials, and/or any other data collected in association with the activation prompt presented by the container application 112.

[0040] To illustrate aspects of the techniques disclosed herein, FIGS. 2 - 4 illustrate exemplary data flow scenarios in which various operations and/or functionalities occur in a specific order. For example, FIG. 2 describes various operations and/or functionalities occurring at five sequential times that are individually labeled T_1 through T_5. However, the order in which these operations and/or functionalities are described and/or illustrated herein is not intended to be construed as a limitation. Rather, any number of the operations and/or functionalities described with respect to FIGS. 2 - 4 may be combined in any order and/or in parallel in accordance with the present disclosure. Other processes and/or operations and/or function-

alities described throughout this disclosure shall be interpreted accordingly.

[0041] Turning now to FIG. 2, illustrated is an exemplary system 200 in which a user 210 instructs the OS 116 to launch a first application 204(1) which then responds by checking for a license both at the local storage 118 and with the application license manager 102 prior to exposing an activation prompt 212.

[0042] At time T_1, the user 210 provides first user input 202(1) to the computing device 106 via an input device 208. As illustrated, the first user input 202(1) is a "Launch App" instruction that corresponds to the first application 204(1). Upon receipt of the first user input 202(1), the OS 116 may begin to execute the first application 204(1). In some implementations, the OS 116 further initializes the container 110 and/or the communication channel 114 to isolate the first application 204(1) from a second application 204(2).

[0043] At time T_2, the first application 204(1) analyzes the local storage 118 to check for a corresponding license. As shown, the first application 204(1) is shown to be checking for the container license 132 which is currently not present at the local storage 118. It can be appreciated that the presence or absence of a corresponding license may inform the first application 204(1) as to various operations to subsequently perform. For example, if a corresponding license is present in the local storage 118, this may inform the first application 204(1) to make certain functionalities available to the user 210. Moreover, according to conventional application activation techniques, if a corresponding license is not present, this may inform the first application 204(1) to perform alternative operations such as, for example, exposing the activation prompt 212.

[0044] In contrast to the aforementioned conventional application activation techniques, at time T_3, the first application 204(1) may respond to a corresponding license being absent from the local drive 118 by transmitting the first activation inquiry 122(1) to the application license manager 102. As shown in the license data 104 of FIG. 2, the License Status associated with the computing device 106 at the time when the first activation inquiry 122(1) is transmitted is "Not Activated."

[0045] At time T_4, therefore, the application license manager 102 responds to the first activation inquiry 122(1) with license status data 124 to inform the first application 204(1) that no license is currently available for the first application 204(1) at the computing device 106 (as identified by the machine ID 120).

[0046] At time T_5, the first application 204(1) responds to receiving the license status data 124 by causing an output device 206 to expose an activation prompt 212. As indicated by the lock symbol of the container application remaining "locked," the first application 204(1) remains locked throughout the exemplary data flow scenario of FIG. 2.

[0047] Turning now to FIG. 3, illustrated is an exemplary system 300 in which a user 210 activates the first application 204(1) which causes the application license manager 102 to provide a license to the first application 204(1) and also to update the license data 104 to designate a second application 204(2) for automatic activation at the computing device 106. For illustrative purposes, FIG. 3 is discussed with respect to an exemplary data flow scenario that begins at time T_6 which is subsequent to T_5 as discussed in relation to FIG. 2.

[0048] At time T_6, the user 210 provides second user input 202(2) to the computing device 106 via the input device 208. As illustrated, the second user input 202(2) is an "Activate App" instruction that corresponds to the first application 204(1). The "Activate App" instruction may include any information that is relevant to activation of the first application 204(1) and/or the second application 204(2). For purposes of the present discussion, assume that the "Activate App" instruction includes the machine ID 120 and indicates a type of license being requested and a term of the license being requested. In various implementations, the license type may indicate which applications on the computing device 106 the "Activate App" instruction is requesting activation for.

[0049] At time T_7, the first application 204(1) transmits an activation request 126 to the application license manager 102 to request a corresponding license to unlock certain functionalities of the first application 204(1) on the computing device 106. Responsive to the activation request 126, the application license manager 102 updates license data 104 to designate the second application 204(2) for automatic activation at the computing device 106. For example, the "updated" license data 104 indicates that a license that is issued to the first application 204(1) based on the activation request 126 is equally applicable to the second application 204(2).

[0050] At time T_8, the application license manager 102 responds to the activation request 126 with product activation data 130. As indicated by the lock symbol of the first application 204(1) being toggled from "locked" to "unlocked," the product activation data 130 is usable to activate the first application 204(1) on the computing device 106. For example, the product activation data 130 may include the container license 132.

[0051] At time T_9, the first application 204(1) saves at least some of the product activation data 130 into the local storage 118 for subsequent use in unlocking one or more application functionalities 302.

[0052] At time T_{10}, the first application 204(1) may expose the one or more application functionalities 302 to the user 210 via the output device 206.

[0053] Turning now to FIG. 4, illustrated is an exemplary system 400 in which a user 210 instructs the OS 116 to launch the second application 204(2) after the license data 104 has been updated as described in FIG. 3 to designate the second application 204(2) for automatic activation at the computing device 106. For illustrative purposes, FIG. 4 is discussed with respect to an exemplary data flow scenario that begins at time T_{11} which is subsequent to T_{10} as discussed in relation to

FIG. 3. Furthermore, as indicated by the strike-through text and the "X" symbol over the container 110 and the communication channel 114, the operations described in FIG. 4 may occur after the first application 204(1) and/or a container 110 corresponding thereto have been terminated.

[0054] At time T_{11}, the user 210 provides third user input 202(3) to the computing device 106 via an input device 208. As illustrated, the third user input 202(3) is a "Launch App" instruction that corresponds to the second application 204(2). Upon receipt of the third user input 202(3), the OS 116 may begin to execute the second application 204(2).

[0055] At time T_{12}, the second application 204(2) analyzes the local storage 118 to check for a corresponding license. It can be appreciated that the presence or absence of a corresponding license may inform the second application 204(2) as to various operations to subsequently perform. For example, if a corresponding license is present in the local storage 118, the second application 204(2) may unlock certain functionalities for use by the user 210. As illustrated, however, a corresponding license is not present at time T_{12}. For example, although a license does exist on the local storage 118, that license is not obtainable by the second application 204(2) for one or more reasons. For example, by virtue of one or both of the first application 204(1) or the second application 204(2) being designated to be run within a container 110, the container license 132 may be stored in association with a particular namespace and/or partition that is inaccessible to the second application 204(2).

[0056] At time T_{13}, therefore, the second application 204(2) responds to a corresponding license being absent from the local drive 118 by transmitting a second activation inquiry 122(2) to the application license manager 102. As shown in the license data 104 of FIG.4, the License Status associated with the computing device 106 at the time when the second activation inquiry 122(2) is transmitted is "Activated." Furthermore, the license scope of this activation includes the second application 204(2).

[0057] At time T_{14}, therefore, the application license manager 102 responds to the second activation inquiry 122(2) with counterpart product activation data 134 to automatically activate the second application 204(2) on the computing device 106 without the user 210 being exposed to any subsequent activation prompt as was previously exposed at time T_5 in FIG. 2.

[0058] At time T_{15}, the second application 204(2) saves at least some of the counterpart product activation data 134 to the local storage 118. For example, as illustrated, the second application 204(2) is shown to store the native license 136 as obtained within and/or constructed from the counterpart product activation data 134.

[0059] At time T_{16}, the second application 204(2) exposes the one or more application functionalities 402 to the user 210 via the output device 206. In this way, despite the second application 204(2) being launched by the user 210 prior to being activated on the computing device 106, and the computing device 106 being configured such that the first application 204(1) cannot share its license with the second application 204(2), the second application 204(2) refrains from re-prompting the user 210 to activate the second application 204(2) as would occur according to existing application activation systems.

[0060] Accordingly, the disclosed technologies represent a substantial advantage over existing application activation systems which can jointly activate counterpart applications only under circumstances where the counterpart applications can directly communicate with one another. For example, despite the second application 204(2) being provided with the container license 132, it may be limited in terms of where it is permitted to store this container license 132 (e.g., since it is designed to be containerized). Thus, it may be unable store the container license 132 in a location that is accessible to the first application 204(1) such as, for example, in Registry Keys of the OS 116. Accordingly, the techniques described herein provide for a secure and efficient manner of jointly activating isolated counterpart applications and avoiding obnoxious and computationally burdensome redundant activation prompts.

[0061] It can be appreciated that redundant computing processes (e.g., exposing redundant activation prompts) result in exorbitant consumption of computing resources. For example, by causing a user to progress through redundant activation sequences for counterpart products, the processing units of a computing device are caused to undergo unnecessarily high numbers of processing cycles. Additionally, due to the user having to re-enter much of the same information (e.g., credentials, user names, passwords, etc.) in a redundant activation prompt as was previously entered in an initial activation prompt, it can further be appreciated that redundant activation prompts also unnecessarily increase networking traffic.

[0062] FIG. 5 is a flow diagram of an example method 500 for an application to perform upon being launched to check both locally for an existing license and remotely for an available license prior to exposing an activation prompt to a user. It should be understood by those of ordinary skill in the art that the operations of the methods disclosed herein are not necessarily presented in any particular order and that performance of some or all of the operations in an alternative order(s) is possible and is contemplated. The operations have been presented in the demonstrated order for ease of description and illustration. Operations may be added, omitted, performed together, and/or performed simultaneously, without departing from the scope of the appended claims.

[0063] At block 501, the OS 116 of the computing device 106 may receive an instruction to launch an application such as, for example, the container application 112 or a native application 108. For example, first user input 202(1) may be received via an input device 208 such as a user double-clicking an icon for the application.

[0064] At block 503, the application checks a local storage 118 for an existing license to determine whether the application has been activated on the computing device 106. For example, the application may examine Registry Key values, a virtual registry, and/or any other suitable location for securely storing a license in association with the application.

[0065] If an existing license exists on the local storage 118, the process 500 may advance from block 503 to block 505 at which the application is launched with full functionality being provisioned to the user. Thus, similar to existing application licensing techniques, according to the techniques described herein, when a license exists locally for an application a user may launch that application to access functionality without being prompted to activate the application.

[0066] If an existing license is not available to the application from the local storage 118, the process 500 may advance from block 503 to block 507 at which an activation inquiry is transmitted to a remote resource to check whether a license is available to the application due to having been remotely designated for automatic activation at the computing device. For example, the activation inquiry 122 may be transmitted to the application license manager 102 which maintains license data 104 in the cloud (e.g., one or more remote and/or distributed servers) for multiple applications of the computing device 106.

[0067] At block 509, the application determines whether a license is available from the remote resource. For example, the application may receive license status data 124 indicating that the application has not been designated for automatic activation and, therefore, that an available license cannot be obtained from the remote resource. In contrast, the application may receive counterpart product activation data which includes a license for activating the application on the computing device.

[0068] If a license is available from the remote resource, the process 500 may advance from block 509 to block 511 at which the available license is obtained by, for example, extracting the license from the counterpart product activation data. Then, the application is launched with full functionality being provisioned to the user without the user being prompted to activate the application at the computing device. Thus, in contrast to existing application licensing techniques, according to the techniques described herein, when a license does not exist locally for an application but is available from a remote resource, a user may launch that application to access functionality without being prompted to activate the application. More specifically, existing application licensing techniques, would typically prompt the user to activate the product when no license exists locally without regard to whether a license may be available from a remote resource.

[0069] If a license is not available from the remote resource, the process 500 may advance from block 509 to block 513 at which the user is prompted to activate the application on the computing device. For example, the application may expose the activation prompt 212 to inform the user that the product is not activated and to provide one or more activation options to the user.

[0070] At block 515, the application may transmit an activation request to the remote resource based on user input that is received via the activation prompt exposed at block 513. Then, upon receiving product activation data in response to the activation request, the process 500 may advance to block 511 as described above.

[0071] FIG. 6 is a flow diagram of an example method 600 for an application license manager to perform to activate multiple applications on a computing device based on an activation request received from an individual one of the multiple applications.

[0072] At block 601, the application license manager 102 receives an activation request from a first application 204(1) that is installed on a computing device 106. For example, a user may launch the first application 204(1) which then determines that no license exists locally and furthermore that no license is available from the application license manager. Accordingly, the user may be prompted to activate the first application 204(1) at the computing device.

[0073] At block 603, the application license manager 102 updates license data 104 to designate both the first application 204(1) and a second application 204(2) on the client device 106 for activation. For example, the activation request may define an activation scope to include both applications.

[0074] At block 605, the application license manager 102 may respond to the activation request by transmitting to the client device product activation data that is addressed to the first application. The product activation data may include a license to activate the first application 204(1).

[0075] At block 607, the application license manager 102 receives an activation inquiry from the second application 204(2) that is installed on the computing device 106. For example, a user may launch the second application 204(2), which then determines that no license exists locally; then, rather than immediately exposing an activation prompt to the user, transmits the activation inquiry to inquire as to whether any counterpart products have performed an activation that is applicable to the second application 204(2).

[0076] At block 609, the application license manager 102 may respond to the activation inquiry by transmitting to the client device counterpart product activation data that is addressed to the second application. The counterpart product activation data may include a license to activate the second application 204(2). In this way, if a user has already activated a counterpart application which is also installed on the computing device, but which is restricted from locally sharing a license, the user will not be prompted to perform another activation which is redundant to that performed with respect to the counterpart product.

[0077] FIG. 7 shows additional details of an example computer architecture 700 for a computer capable of ex-

ecuting the systems and methods described herein. In particular, the example computing architecture 700 is capable of executing functions described in relation to the OS 116, the container application 112, the native application 108, the application license manager 102, and/or any other program components thereof as described herein. Thus, the computer architecture 700 illustrated in FIG. 7 illustrates an architecture for a server computer, network of server computers, or any other types of computing devices suitable for implementing the functionality described herein. The computer architecture 700 may be utilized to execute any aspects of the software components presented herein.

[0078] The computer architecture 700 illustrated in FIG. 7 includes a central processing unit 702 ("CPU"), a system memory 704, including a random-access memory 706 ("RAM") and a read-only memory ("ROM") 708, and a system bus 710 that couples the memory 704 to the CPU 702. A basic input/output system containing the basic routines that help to transfer information between elements within the computer architecture 700, such as during startup, is stored in the ROM 708. The computer architecture 700 further includes a mass storage device 712 for storing the host OS that supports the containers 110, other data, and one or more application programs. The mass storage device 712 may further include one or more of the OS 116, the local storage 118, the native application(s) 108, and/or the container application 112.

[0079] The mass storage device 712 is connected to the CPU 702 through a mass storage controller (not shown) connected to the bus 710. The mass storage device 712 and its associated computer-readable media provide non-volatile storage for the computer architecture 700. Although the description of computer-readable media contained herein refers to a mass storage device, such as a solid-state drive, a hard disk or CD-ROM drive, it should be appreciated by those skilled in the art that computer-readable media can be any available computer storage media or communication media that can be accessed by the computer architecture 700.

[0080] Communication media includes 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 delivery media. The term "modulated data signal" means a signal that has one or more of its characteristics changed or set in 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 the any of the above should also be included within the scope of computer-readable media.

[0081] By way of example, and not limitation, computer storage media may include volatile and non-volatile, 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. For example, computer media includes, but is not limited to, RAM, ROM, EPROM, EEPROM, flash memory or other solid-state memory technology, CD-ROM, digital versatile disks ("DVD"), HD-DVD, BLU-RAY, or other optical 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 be accessed by the computer architecture 700. For purposes of the claims, the phrase "computer storage medium," "computer-readable storage medium" and variations thereof, does not include waves, signals, and/or other transitory and/or intangible communication media, per se.

[0082] According to various techniques, the computer architecture 700 may operate in a networked environment using logical connections to remote computers through a network 750 and/or another network (not shown). The computer architecture 700 may connect to the network 750 through one or more network interface units 716 connected to the bus 710 and/or the container application 112. In the illustrated embodiment, container application 112 is connected to a first network interface unit 716(A) that provides container application 112 with access to the application license manager 102. The computer architecture 700 further includes a communications channel 114 that isolates container application 112 from various components of the computer architecture 700 such as, for example, the OS 116 and/or certain native available portions of the local storage 118.

[0083] The illustrated computer architecture 700 further includes a second network interface unit 716(B) that connects various native applications 108 to the one or more networks 750 via a firewall 720. As illustrated, the networks 750 and firewall 720 enable communications to be passed through to the application license manager 102. The firewall 720 is configured to provide the computer architecture 700 with an inability to perform at least some outward communications while blocking unauthorized access from the networks 750. It should be appreciated that the network interface unit(s) 716 also may be utilized to connect to other types of networks and remote computer systems. The computer architecture 700 also may include an input/output controller 718 for receiving and processing input from a number of other devices, including a keyboard, mouse, or electronic stylus (not shown in FIG. 7). Similarly, the input/output controller 718 may provide output to a display screen, a printer, or other type of output device (also not shown in FIG. 7). It should also be appreciated that via a connection to the network 750 through a network interface unit 716, the computing architecture may enable communication between the functional components described herein.

[0084] It should be appreciated that the software components described herein may, when loaded into the CPU 702 and executed, transform the CPU 702 and the overall computer architecture 700 from a general-purpose computing system into a special-purpose comput-

ing system customized to facilitate the functionality presented herein. The CPU 702 may be constructed from any number of transistors or other discrete circuit elements, which may individually or collectively assume any number of states. More specifically, the CPU 702 may operate as a finite-state machine, in response to executable instructions contained within the software modules disclosed herein. These computer-executable instructions may transform the CPU 702 by specifying how the CPU 702 transitions between states, thereby transforming the transistors or other discrete hardware elements constituting the CPU 702.

[0085] Encoding the software modules presented herein also may transform the physical structure of the computer-readable media presented herein. The specific transformation of physical structure may depend on various factors, in different implementations of this description. Examples of such factors may include, but are not limited to, the technology used to implement the computer-readable media, whether the computer-readable media is characterized as primary or secondary storage, and the like. For example, if the computer-readable media is implemented as semiconductor-based memory, the software disclosed herein may be encoded on the computer-readable media by transforming the physical state of the semiconductor memory. For example, the software may transform the state of transistors, capacitors, or other discrete circuit elements constituting the semiconductor memory. The software also may transform the physical state of such components in order to store data thereupon.

[0086] As another example, the computer-readable media disclosed herein may be implemented using magnetic or optical technology. In such implementations, the software presented herein may transform the physical state of magnetic or optical media, when the software is encoded therein. These transformations may include altering the magnetic characteristics of particular locations within given magnetic media. These transformations also may include altering the physical features or characteristics of particular locations within given optical media, to change the optical characteristics of those locations. Other transformations of physical media are possible without departing from the scope and spirit of the present description, with the foregoing examples provided only to facilitate this discussion.

[0087] In light of the above, it should be appreciated that many types of physical transformations take place in the computer architecture 700 in order to store and execute the software components presented herein. It also should be appreciated that the computer architecture 700 may include other types of computing devices, including hand-held computers, embedded computer systems, personal digital assistants, and other types of computing devices known to those skilled in the art. It is also contemplated that the computer architecture 700 may not include all of the components shown in FIG. 7, may include other components that are not explicitly shown in FIG. 7, or may utilize an architecture completely different than that shown in FIG. 7.

CONCLUSION

[0088] In closing, although the various techniques have been described in language specific to structural features and/or methodological acts, it is to be understood that the subject matter defined in the appended representations is not necessarily limited to the specific features or acts described. Rather, the specific features and acts are disclosed as example forms of implementing the claimed subject matter.

Claims

1. A computing device (106), comprising:

one or more processors (702);
a memory (704) in communication with the one or more processors (702), the memory (704) having computer-readable instructions stored thereupon which, when executed by the one or more processors (702), implement a native application (108) and a container application (112), that is restricted from communicating with the native application (108), the container application (112) configured to:

based on a first user input (202(1)), cause an output device (206) to expose an activation prompt (212);based on a second user input (202(2)) being provided in response to the activation prompt, transmit, from the container application (112), an activation request (126) that includes at least a machine identification, ID (120) to cause an application license manager (102) to update license data (104) for the native application (108) in association with the machine ID (120);
the native application (108) configured to:

based on a third user input (202(3)), subsequent to transmitting the activation request (126) from the container application (112), transmit an activation inquiry (122) from the native application (108) to the application license manager (102), wherein the activation inquiry (122) includes at least the machine ID (120) to query whether the native application (108) is licensed with respect to the machine ID (120);
receive, from the application license manager (102), in response to the activation inquiry (122), counterpart prod-

uct activation data (134) that includes a native license (136) corresponding to the native application (108); and
store the native license (136) in association with the native application (108) to activate the native application (108) on the computing device (106).

2. The computing device (106) of claim 1, wherein activating the native application (108) on the computing device (106) includes storing the native license (136), that is received from the application license manager (102), in a native available portion of a local storage (118) that is available to the native application (108) and is unavailable to the container application (110).

3. The computing device (106) of claim 2, wherein the computer-readable instructions further cause the container application (112) to:

receive, from the application license manager (102), product activation data (130) that includes a container license (132) corresponding to the container application (112); and
store the container license (132) in a container available portion of the local storage (118) that is available to the container application (112) and is unavailable to the native application (108).

4. The computing device (106) of claim 1, wherein the computer-readable instructions further cause the native application (108)to:
determine, in response to the third user input (202(3)), whether the native license (136) is stored on a local storage in association with the native application (108), wherein the activation inquiry (122) is transmitted to the application license manager (102) in response to a determination that the native license (136) is not stored on the local storage in association with the native application (108).

5. The computing device (106) of claim 1, wherein the computer-readable instructions further cause the container application (112) to:

prior to transmitting the activation request (126), transmit a second activation inquiry (122(2)) to the application license manager (102), wherein the second activation inquiry (122(2) includes at least the machine ID (120) to query whether the container application (112) is licensed with respect to the machine ID (120); and
cause an output device (206) to expose an activation prompt (212) in response to a determination that the container application (112) is not licensed with respect to the machine ID (120).

6. The computing device (106) of claim 1, wherein the activation request (126) that causes the application license manager (102) to update the license data (104) for the native application (108) is generated based on user input (202) that is received via the container application (112) that the OS (116) is running within the container (110) that at least partially restricts communications with the native application (108).

7. The computing device (106) of claim 1, wherein the native license (136) is a substantially duplicative instance of a container license that is usable for activation of the container application (112) on the computing device (106).

8. The computing device (106) of claim 1, wherein the activation request (126) that is transmitted by the container application (112) indicates a license term that is applicable to the native application (108).

9. The computing device (106) of claim 1, wherein the activation request (126) that is transmitted from the container application (112) indicates an activation scope that defines a subset of a plurality of native applications (108(1), 108(2), ... 108(N)) for activation on the computing device (106).

10. The computing device (106) of claim 1, wherein the container application (112) is further configured to:

determine, in response to the first user input (202(1)), whether a corresponding license is stored in a local storage (118);
transmit a second activation inquiry (122(2)) to the application license manager (102) based on a determination that a corresponding license is absent from the local storage (118); and
receive license status data (124) from the application license manager (102) in response to the second activation inquiry (122(2)), wherein exposing the activation prompt (212) is further based on the license status data (124).

11. A computer-implemented method, comprising:

based on a first user input (202(1)), causing an output device (206) to expose an activation prompt (212);
based on a second user input (202(2)), transmitting, from a container application (112) on a computing device (106), an activation request (126) that includes at least a machine identification, ID (120), wherein the computing device (106) is configured to restrict the container application (112) from communicating with a native application (108), and wherein the second user input is provided in response to the activation

prompt;
based at least in part on the activation request (126) being transmitted from the container application (112), updating license data (104) for the native application (108) in association with the machine ID (120);
based on a third user input (202(3)), subsequent to transmitting the activation request (126) from the container application (112), transmitting an activation inquiry (122) from the native application (108) to an application license manager (102), wherein the activation inquiry (122) includes at least the machine ID (120) to query whether the native application (108) is licensed with respect to the machine ID (120);
receiving, from the application license manager (102), in response to the activation inquiry (122), counterpart product activation data that includes a native license (136) corresponding to the native application (108); and
storing the native license (136) in association with the native application (108) to activate the native application (108) on the computing device (106).

12. The computer-implemented method of claim 11, further comprising transmitting, to the computing device (106), product activation data (130) that is addressed to the container application (112) and that includes first license data that is usable to activate the container application (112) on the computing device (106) responsive to the activation request (126) that is transmitted from the container application (112).

Patentansprüche

1. Rechenvorrichtung (106), umfassend:

einen oder mehrere Prozessoren (702);
einen Speicher (704), der mit dem einen oder den mehreren Prozessoren (702) kommuniziert, wobei in dem Speicher (704) computerlesbare Anweisungen gespeichert sind, die, wenn sie von dem einen oder den mehreren Prozessoren (702) ausgeführt werden, eine native Anwendung (108) und eine Containeranwendung (112) implementieren, die von der Kommunikation mit der nativen Anwendung (108) abgehalten wird, wobei die Containeranwendung (112) so konfiguriert ist, dass sie:

basierend auf einer ersten Benutzereingabe (202(1)) eine Ausgabevorrichtung (206) veranlasst, eine Aktivierungsaufforderung (212) freizulegen; basierend auf einer zweiten Benutzereingabe (202(2)), die als Reaktion auf die Aktivierungsaufforderung bereitgestellt wird, von der Containeranwendung (112) eine Aktivierungsanforderung (126) überträgt, die mindestens eine Maschinenkennung, ID(120) enthält, um einen Anwendungslizenzmanager (102) zu veranlassen, Lizenzdaten (104) für die native Anwendung (108) in Verbindung mit der Maschinenkennung (120) zu aktualisieren;
wobei die native Anwendung (108) so konfiguriert ist, dass sie:

basierend auf einer dritten Benutzereingabe (202(3)), im Anschluss an das Übertragen der Aktivierungsanforderung (126) von der Containeranwendung (112) eine Aktivierungsanfrage (122) von der nativen Anwendung (108) an den Anwendungslizenzmanager (102) übertragt, wobei die Aktivierungsanfrage (122) mindestens die Maschinenkennung (120) enthält, um abzufragen, ob die native Anwendung (108) in Bezug auf die Maschinenkennung (120) lizenziert ist;
von dem Anwendungslizenzmanager (102) als Reaktion auf die Aktivierungsanfrage (122) Gegenstück-Produktaktivierungsdaten (134) empfängt, die eine native Lizenz (136) enthalten, die der nativen Anwendung (108) entspricht; und
die native Lizenz (136) in Verbindung mit der nativen Anwendung (108) speichert, um die native Anwendung (108) auf der Rechenvorrichtung (106) zu aktivieren.

2. Rechenvorrichtung (106) nach Anspruch 1, wobei das Aktivieren der nativen Anwendung (108) auf der Rechenvorrichtung (106) das Speichern der nativen Lizenz (136), die von dem Anwendungslizenzmanager (102) empfangen wird,
in einem nativen verfügbaren Abschnitt eines lokalen Speichers (118) umfasst, der für die native Anwendung (108) verfügbar und für die Containeranwendung (110) nicht verfügbar ist.

3. Rechenvorrichtung (106) nach Anspruch 2, wobei die computerlesbaren Anweisungen weiter die Containeranwendung (112) veranlassen zum:

Empfangen, von dem Anwendungslizenzmanager (102), von Produktaktivierungsdaten (130), die eine der Containeranwendung (112) entsprechende Containerlizenz (132) enthalten; und
Speichern der die Containerlizenz (132) in ei-

nem für Container verfügbaren Abschnitt des lokalen Speichers (118) speichern, der für die Containeranwendung (112) verfügbar ist und für die native Anwendung (108) nicht verfügbar ist.

4. Rechenvorrichtung (106) nach Anspruch 1, wobei die computerlesbaren Anweisungen weiter die native Anwendung (108) veranlassen zum:
Bestimmen, als Reaktion auf die dritte Benutzereingabe (202(3)), ob die native Lizenz (136) auf einem lokalen Speicher in Verbindung mit der nativen Anwendung (108) gespeichert ist, wobei die Aktivierungsanfrage (122) an den Anwendungslizenzmanager (102) als Reaktion auf eine Bestimmung, dass die native Lizenz (136) nicht auf dem lokalen Speicher in Verbindung mit der nativen Anwendung (108) gespeichert ist, übertragen wird.

5. Rechenvorrichtung (106) nach Anspruch 1, wobei die computerlesbaren Anweisungen weiter die Containeranwendung (112) veranlassen zum:

Übertragen, vor dem Übertragen der Aktivierungsanforderung (126), einer zweiten Aktivierungsanfrage (122(2)) an den Anwendungslizenzmanager (102), wobei die zweite Aktivierungsanfrage (122(2)) zumindest die Maschinenkennung (120) enthält, um abzufragen, ob die Containeranwendung (112) in Bezug auf die Maschinenkennung (120) lizenziert ist; und
Veranlassen einer Ausgabevorrichtung (206), eine Aktivierungsaufforderung (212) als Reaktion auf die Bestimmung freizulegen, dass die Containeranwendung (112) in Bezug auf die Maschinenkennung (120) nicht lizenziert ist.

6. Rechenvorrichtung (106) nach Anspruch 1, wobei die Aktivierungsanforderung (126), die den Anwendungslizenzmanager (102) veranlasst, die Lizenzdaten (104) für die native Anwendung (108) zu aktualisieren, basierend auf einer Benutzereingabe (202) erzeugt wird, die über die Containeranwendung (112) empfangen wird, die das Betriebssystem (116) innerhalb des Containers (110) ausführt, der die Kommunikation mit der nativen Anwendung (108) zumindest teilweise einschränkt.

7. Rechenvorrichtung (106) nach Anspruch 1, wobei die native Lizenz (136) eine im Wesentlichen doppelte Instanz einer Containerlizenz ist, die für die Aktivierung der Containeranwendung (112) auf der Rechenvorrichtung (106) verwendbar ist.

8. Rechenvorrichtung (106) nach Anspruch 1, wobei die Aktivierungsanforderung (126), die durch die Containeranwendung (112) übertragen wird, eine Lizenzbedingung angibt, die für die native Anwendung (108) gilt.

9. Rechenvorrichtung (106) nach Anspruch 1, wobei die Aktivierungsanforderung (126), die von der Containeranwendung (112) übertragen wird, einen Aktivierungsbereich angibt, der eine Teilmenge einer Vielzahl von nativen Anwendungen (108(1), 108(2), ... 108(N)) zur Aktivierung auf der Rechenvorrichtung (106) definiert.

10. Rechenvorrichtung (106) nach Anspruch 1, wobei die Containeranwendung (112) weiter konfiguriert ist zum:

Bestimmen, als Reaktion auf die erste Benutzereingabe (202(1)), ob eine entsprechende Lizenz in einem lokalen Speicher (118) gespeichert ist;
Übertragen einer zweiten Aktivierungsanfrage (122(2)) an den Anwendungslizenzmanager (102) basierend auf einer Bestimmung, dass eine entsprechende Lizenz im lokalen Speicher (118) fehlt; und
Empfangen von Lizenzstatusdaten (124) von dem Anwendungslizenzmanager (102) als Reaktion auf die zweite Aktivierungsanfrage (122(2)), wobei das Freilegen der Aktivierungsaufforderung (212) weiterhin auf den Lizenzstatusdaten (124) basiert.

11. Computerimplementiertes Verfahren, umfassend:

basierend auf einer ersten Benutzereingabe (202(1), Veranlassen einer Ausgabevorrichtung (206), eine Aktivierungsaufforderung (212) freizulegen;
basierend auf einer zweiten Benutzereingabe (202(2)), Übertragen, von einer Containeranwendung (112) auf einer Rechenvorrichtung (106), einer Aktivierungsanforderung (126), die mindestens eine Maschinenkennung, ID (120) enthält, wobei die Rechenvorrichtung (106) so konfiguriert ist, dass sie die Containeranwendung (112) von der Kommunikation mit einer nativen Anwendung (108) abhält, und wobei die zweite Benutzereingabe als Reaktion auf die Aktivierungsaufforderung bereitgestellt wird;
basierend zumindest teilweise auf der Aktivierungsanforderung (126), die von der Containeranwendung (112) übertragen wird, Aktualisierung der Lizenzdaten (104) für die native Anwendung (108) in Verbindung mit der Maschinenkennung (120);
basierend auf einer dritten Benutzereingabe (202(3)), im Anschluss an das Übertragen der Aktivierungsanforderung (126) von der Containeranwendung (112), Übertragen einer Aktivierungsanfrage (122) von der nativen Anwendung (108) an einen Anwendungslizenzmanager (102), wobei die Aktivierungsanfrage (122) min-

destens die Maschinenkennung (120) enthält, um abzufragen, ob die native Anwendung (108) in Bezug auf die Maschinenkennung (120) lizenziert ist;
Empfangen, von dem Anwendungslizenzmanager (102) als Reaktion auf die Aktivierungsanfrage (122), von Gegenstück-Produktaktivierungsdaten, die eine native Lizenz (136) enthalten, die der nativen Anwendung (108) entspricht, und
Speichern der nativen Lizenz (136) in Verbindung mit der nativen Anwendung (108), um die native Anwendung (108) auf der Rechenvorrichtung (106) zu aktivieren.

12. Computerimplementierte Verfahren nach Anspruch 11, weiter umfassend das Übertragen, an die Rechenvorrichtung (106), von Produktaktivierungsdaten (130), die an die Containeranwendung (112) adressiert sind und erste Lizenzdaten enthalten, die zur Aktivierung der Containeranwendung (112) auf der Rechenvorrichtung (106) als Reaktion auf die von der Containeranwendung (112) übertragene Aktivierungsanforderung (126) verwendet werden können.

Revendications

1. Dispositif de calcul (106), comprenant :

un ou plusieurs processeurs (702) ;
une mémoire (704) en communication avec les un ou plusieurs processeurs (702), la mémoire (704) comportant des instructions lisibles par ordinateur, stockées dans celle-ci, qui, lorsqu'elles sont exécutées par les un ou plusieurs processeurs (702), mettent en oeuvre une application native (108) et une application de contenant (112), à laquelle il est interdit de communiquer avec l'application native (108), l'application de contenant (112) étant configurée pour :

sur la base d'une première entrée d'utilisateur (202(1)), amener un dispositif de sortie (206) à exposer une invite d'activation (212) ; sur la base d'une deuxième entrée d'utilisateur (202(2)) étant fournie en réponse à l'invite d'activation, transmettre, depuis l'application de contenant (112), une demande d'activation (126) qui inclut au moins une identification, ID, de machine pour amener un gestionnaire de licence d'application (102) à mettre à jour des données de licence (104) pour l'application native (108) en association avec l'ID de machine (120); l'application native (108) étant configurée pour :

sur la base d'une troisième entrée d'utilisateur (202(3)), à la suite de la transmission de la demande d'activation (126) depuis l'application de contenant (112), transmettre une interrogation d'activation (122) depuis l'application native (108) au gestionnaire de licence d'application (102), dans lequel l'interrogation d'activation (122) inclut au moins l'ID de machine pour interroger si l'application native (108) est ou non sous licence par rapport à l'ID de machine (120);
recevoir, depuis le gestionnaire de licence d'application (102), en réponse à l'interrogation d'activation (122), des données d'activation de produit homologue (134) qui incluent une licence native (136) correspondant à l'application native (108) ; et
stocker la licence native (136) en association avec l'application native (108) pour activer l'application native (108) sur le dispositif de calcul (106).

2. Dispositif de calcul (106) selon la revendication 1, dans lequel l'activation de l'application native (108) sur le dispositif de calcul (106) inclut le stockage de la licence native (136), qui est reçue depuis le gestionnaire de licence d'application (102), dans une portion disponible native d'un stockage local (118) qui est disponible pour l'application native (108) et qui est indisponible pour l'application de contenant (110).

3. Dispositif de calcul (106) selon la revendication 2, dans lequel les instructions lisibles par ordinateur amènent en outre l'application de contenant (112) à :

recevoir, depuis le gestionnaire de licence d'application (102), des données d'activation de produit (130) qui incluent une licence de contenant (132) correspondant à l'application de contenant (112) ; et
stocker la licence de contenant (132) dans une portion disponible de contenant du stockage local (118) qui est disponible pour l'application de contenant (112) et qui est indisponible pour l'application native (108).

4. Dispositif de calcul (106) selon la revendication 1, dans lequel les instructions lisibles par ordinateur amènent en outre l'application native (108) à :
déterminer, en réponse à la troisième entrée d'utilisateur (202(3)), si la licence native (136) est ou non stockée sur un stockage local en association avec l'application native (108), dans lequel l'interrogation d'activation (122) est transmise au gestionnaire de

licence d'application (102) en réponse à une détermination que la licence native (136) n'est pas stockée sur le stockage local en association avec l'application native (108).

5. Dispositif de calcul (106) selon la revendication 1, dans lequel les instructions lisibles par ordinateur amènent en outre l'application de contenant (112) à :

avant la transmission de la demande d'activation (126), transmettre une deuxième interrogation d'activation (122(2)) au gestionnaire de licence d'application (102), dans lequel la deuxième interrogation d'activation (122(2)) inclut au moins l'ID de machine pour interroger si l'application de contenant (112) est sous licence par rapport à l'ID de machine (120); et
amener un dispositif de sortie (206) à exposer une invite d'activation (212) en réponse à une détermination que l'application de contenant (112) n'est pas sous licence par rapport à l'ID de machine (120).

6. Dispositif de calcul (106) selon la revendication 1, dans lequel la demande d'activation (126) qui amène le gestionnaire de licence d'application (102) à mettre à jour les données de licence (104) pour l'application native (108) est générée sur la base d'une entrée d'utilisateur (202) qui est reçue par l'intermédiaire de l'application de contenant (112) que l'OS (116) exécute à l'intérieur du contenant (110) qui interdit au moins partiellement des communications avec l'application native (108).

7. Dispositif de calcul (106) selon la revendication 1, dans lequel la licence native (136) est une instance faisant sensiblement double emploi d'une licence de contenant qui est utilisable pour une activation de l'application de contenant (112) sur le dispositif de calcul (106).

8. Dispositif de calcul (106) selon la revendication 1, dans lequel la demande d'activation (126) qui est transmise par l'application de contenant (112) indique une période de licence qui est applicable à l'application native (108).

9. Dispositif de calcul (106) selon la revendication 1, dans lequel la demande d'activation (126) qui est transmise depuis l'application de contenant (112) indique une portée d'activation qui définit un sous-ensemble d'une pluralité d'applications natives (108(1), 108(2), ... 108(N)) pour une activation sur le dispositif de calcul (106).

10. Dispositif de calcul (106) selon la revendication 1, dans lequel l'application de contenant (112) est en outre configurée pour :

déterminer, en réponse à la première entrée d'utilisateur (202(1)), si une licence correspondante est stockée dans un stockage local (118) ;
transmettre une deuxième interrogation d'activation (122(2)) au gestionnaire de licence d'application (102) sur la base d'une détermination qu'une licence correspondante est absente du stockage local (118) ; et
recevoir des données de statut de licence (124) depuis le gestionnaire de licence d'application (102) en réponse à la deuxième interrogation d'activation (122(2)), dans lequel l'exposition de l'invite d'activation (212) est en outre basée sur les données de statut de licence (124).

11. Procédé mis en oeuvre par ordinateur, comprenant :

sur la base d'une première entrée d'utilisateur (202(1)), l'acte d'amener un dispositif de sortie (206) à exposer une invite d'activation (212) ;
sur la base d'une deuxième entrée d'utilisateur (202(2)), la transmission, depuis une application de contenant (112) sur un dispositif de calcul (106), d'une demande d'activation (126) qui inclut au moins une identification, ID, de machine (120), dans lequel le dispositif de calcul (106) est configuré pour interdire à l'application de contenant (112) de communiquer avec une application native (108), et dans lequel la deuxième entrée d'utilisateur est fournie en réponse à l'invite d'activation ;
sur la base au moins en partie de la demande d'activation (126) qui est transmise depuis l'application de contenant (112), la mise à jour de données de licence (104) pour l'application native (108) en association avec l'ID de machine (120);
sur la base d'une troisième entrée d'utilisateur (202(3)), à la suite de la transmission de la demande d'activation (126) depuis l'application de contenant (112), la transmission d'une interrogation d'activation (122) depuis l'application native (108) à un gestionnaire de licence d'application (102), dans lequel l'interrogation d'activation (122) inclut au moins l'ID de machine pour interroger si l'application native (108) est ou non sous licence par rapport à l'ID de machine (120);
la réception, depuis le gestionnaire de licence d'application (102), en réponse à l'interrogation d'activation (122), de données d'activation de produit homologue qui incluent une licence native (136) correspondant à l'application native (108) ; et
le stockage de la licence native (136) en association avec l'application native (108) pour activer l'application native (108) sur le dispositif de calcul (106).

12. Procédé mis en oeuvre par ordinateur selon la revendication 11, comprenant en outre la transmission, au dispositif de calcul (106), de données d'activation de produit (130) qui sont adressées à l'application de contenant (112) et qui incluent des premières données de licence qui sont utilisables pour activer l'application de contenant (112) sur le dispositif de calcul (106) en réponse à la demande d'activation (126) qui est transmise depuis l'application de contenant (112).

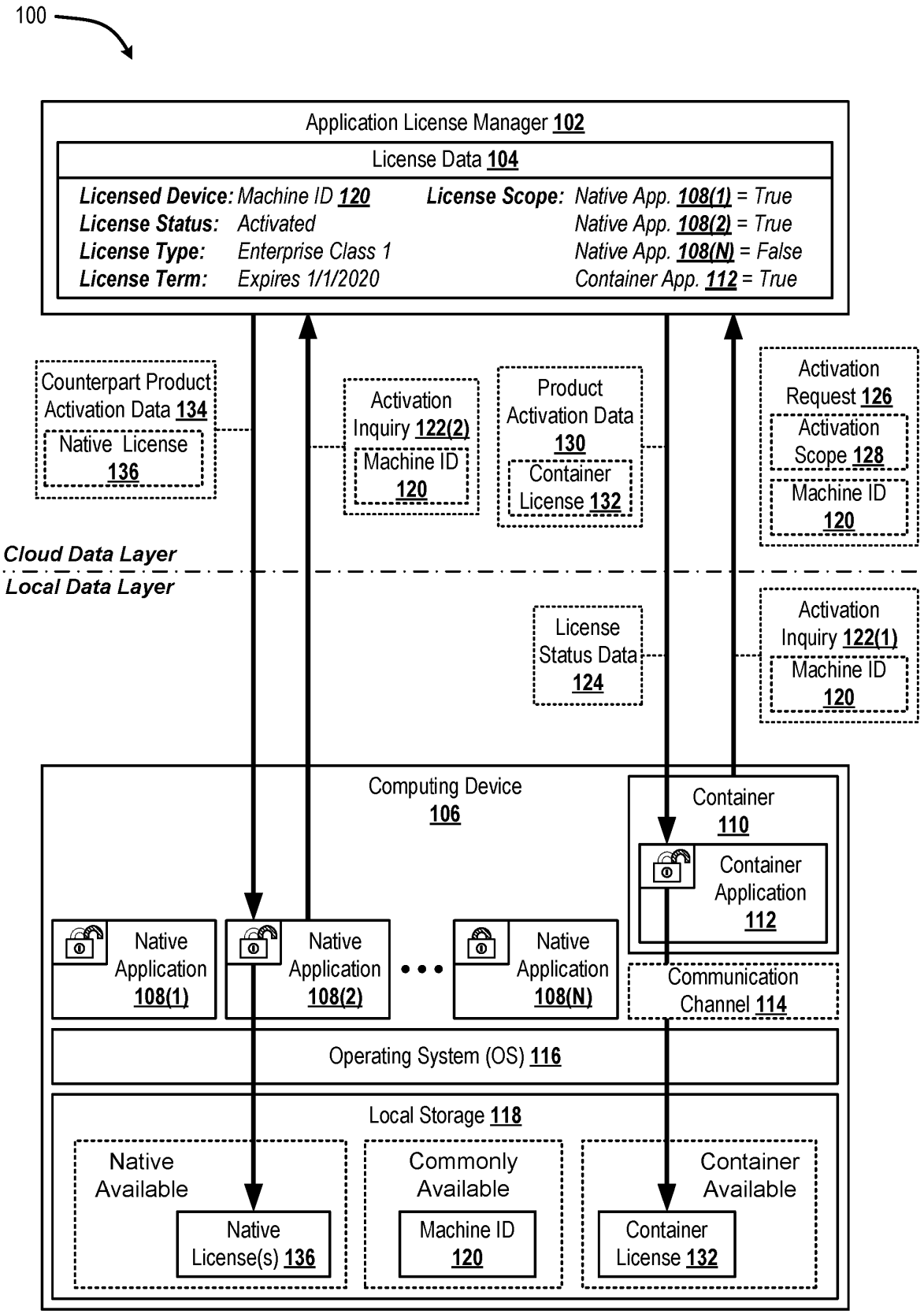
100
Application License Manager 102
License Data 104
Licensed Device: Machine ID 120
License Status: Activated
License Type: Enterprise Class 1
License Term: Expires 1/1/2020
License Scope: Native App. 108(1) = True
Native App. 108(2) = True
Native App. 108(N) = False
Container App. 112 = True
Counterpart Product Activation Data 134
Native License 136
Activation Inquiry 122(2)
Machine ID 120
Product Activation Data 130
Container License 132
Activation Request 126
Activation Scope 128
Machine ID 120
Cloud Data Layer
Local Data Layer
License Status Data 124
Activation Inquiry 122(1)
Machine ID 120
Computing Device 106
Container 110
Container Application 112
Native Application 108(1)
Native Application 108(2)
Native Application 108(N)
Communication Channel 114
Operating System (OS) 116
Local Storage 118
Native Available
Native License(s) 136
Commonly Available
Machine ID 120
Container Available
Container License 132

FIGURE 1

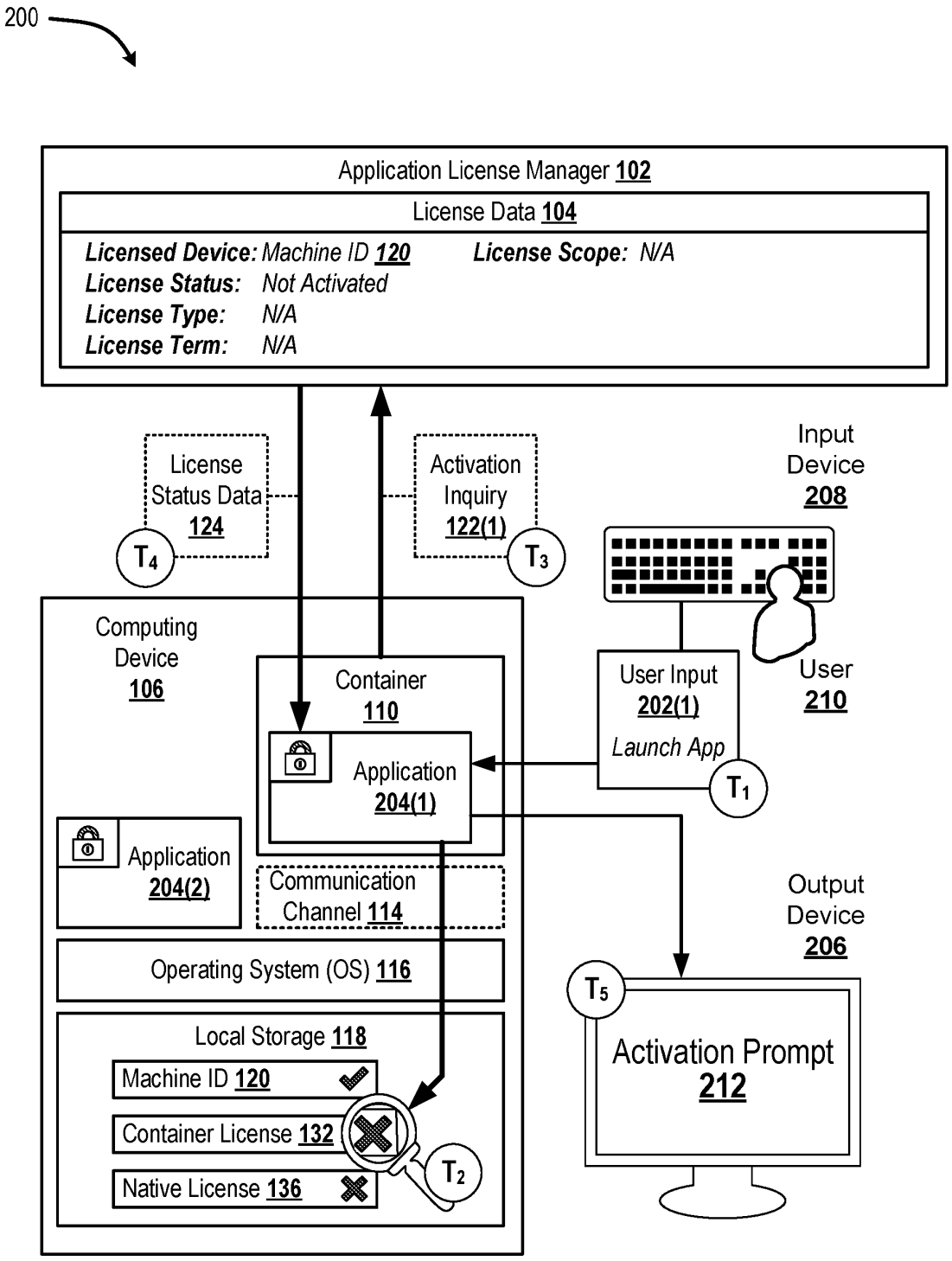
200
Application License Manager 102
License Data 104
Licensed Device: Machine ID 120
License Scope: N/A
License Status: Not Activated
License Type: N/A
License Term: N/A
License Status Data 124
Activation Inquiry 122(1)
Input Device 208
T4
T3
Computing Device 106
Container 110
Application 204(1)
User Input 202(1)
Launch App
User 210
T1
Application 204(2)
Communication Channel 114
Output Device 206
Operating System (OS) 116
T5
Activation Prompt 212
Local Storage 118
Machine ID 120
Container License 132
Native License 136
T2

FIGURE 2

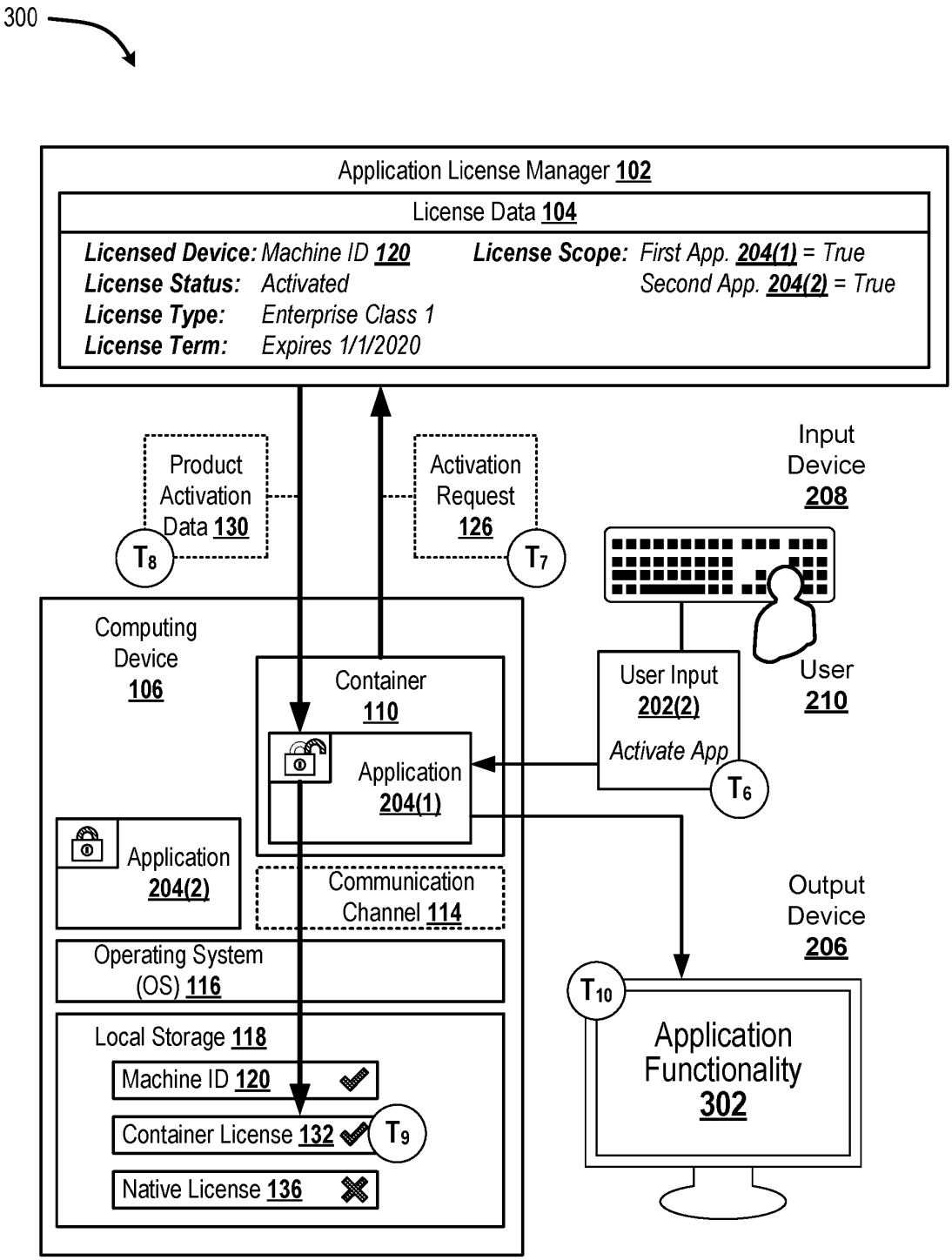
300
Application License Manager 102
License Data 104
Licensed Device: Machine ID 120
License Status: Activated
License Type: Enterprise Class 1
License Term: Expires 1/1/2020
License Scope: First App. 204(1) = True
Second App. 204(2) = True
Product Activation Data 130
T8
Activation Request 126
T7
Input Device 208
Computing Device 106
Container 110
Application 204(1)
User Input 202(2)
Activate App
T6
User 210
Application 204(2)
Communication Channel 114
Output Device 206
Operating System (OS) 116
T10
Application Functionality 302
Local Storage 118
Machine ID 120
Container License 132
T9
Native License 136

FIGURE 3

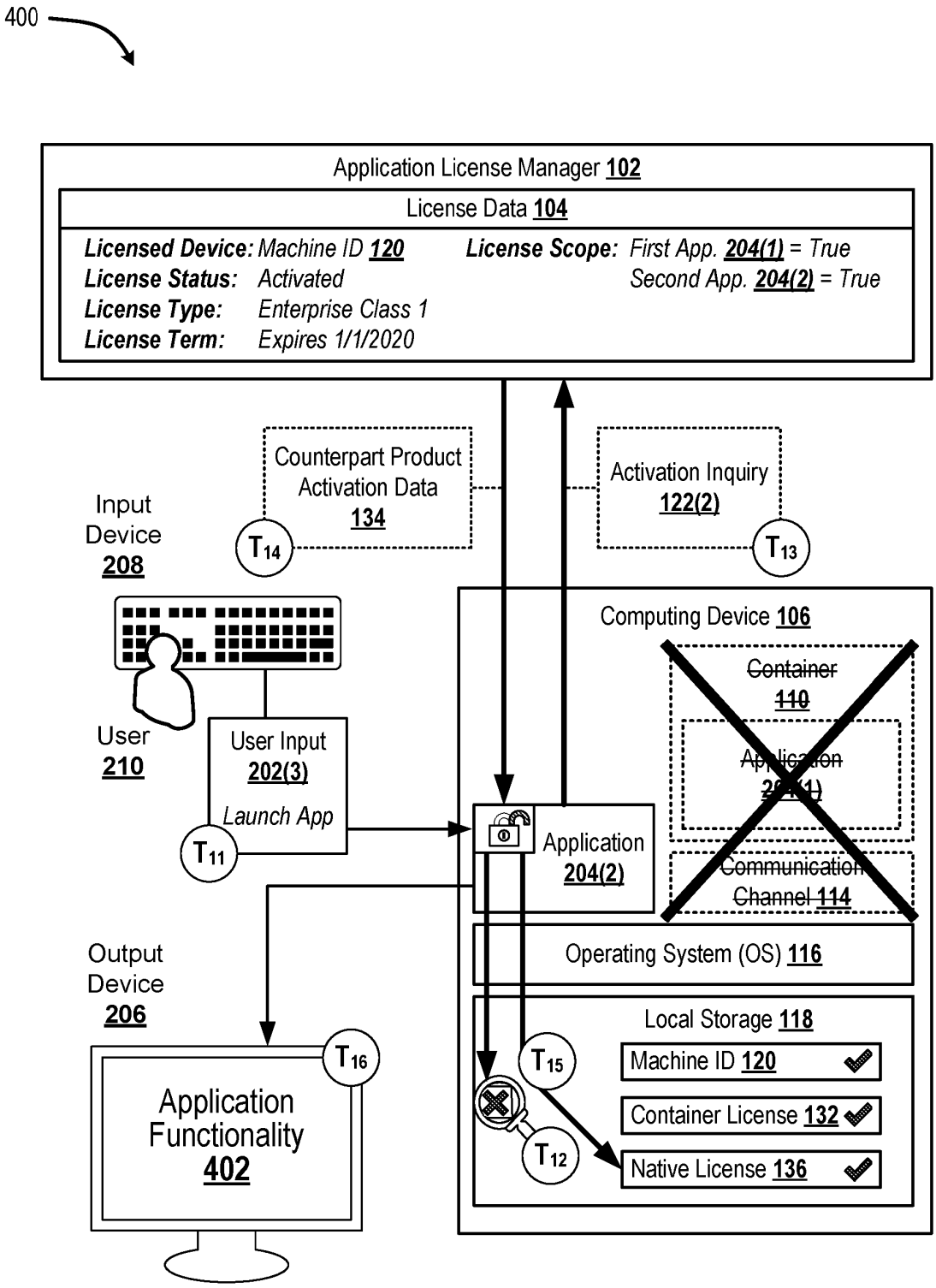
400
Application License Manager 102
License Data 104
Licensed Device: Machine ID 120
License Status: Activated
License Type: Enterprise Class 1
License Term: Expires 1/1/2020
License Scope: First App. 204(1) = True
Second App. 204(2) = True
Counterpart Product Activation Data 134
Activation Inquiry 122(2)
T14
T13
Input Device 208
User 210
User Input 202(3)
Launch App
T11
Computing Device 106
Container 110
Application 204(1)
Communication Channel 114
Application 204(2)
Operating System (OS) 116
Local Storage 118
Machine ID 120
Container License 132
Native License 136
T15
T12
Output Device 206
T16
Application Functionality 402

FIGURE 4

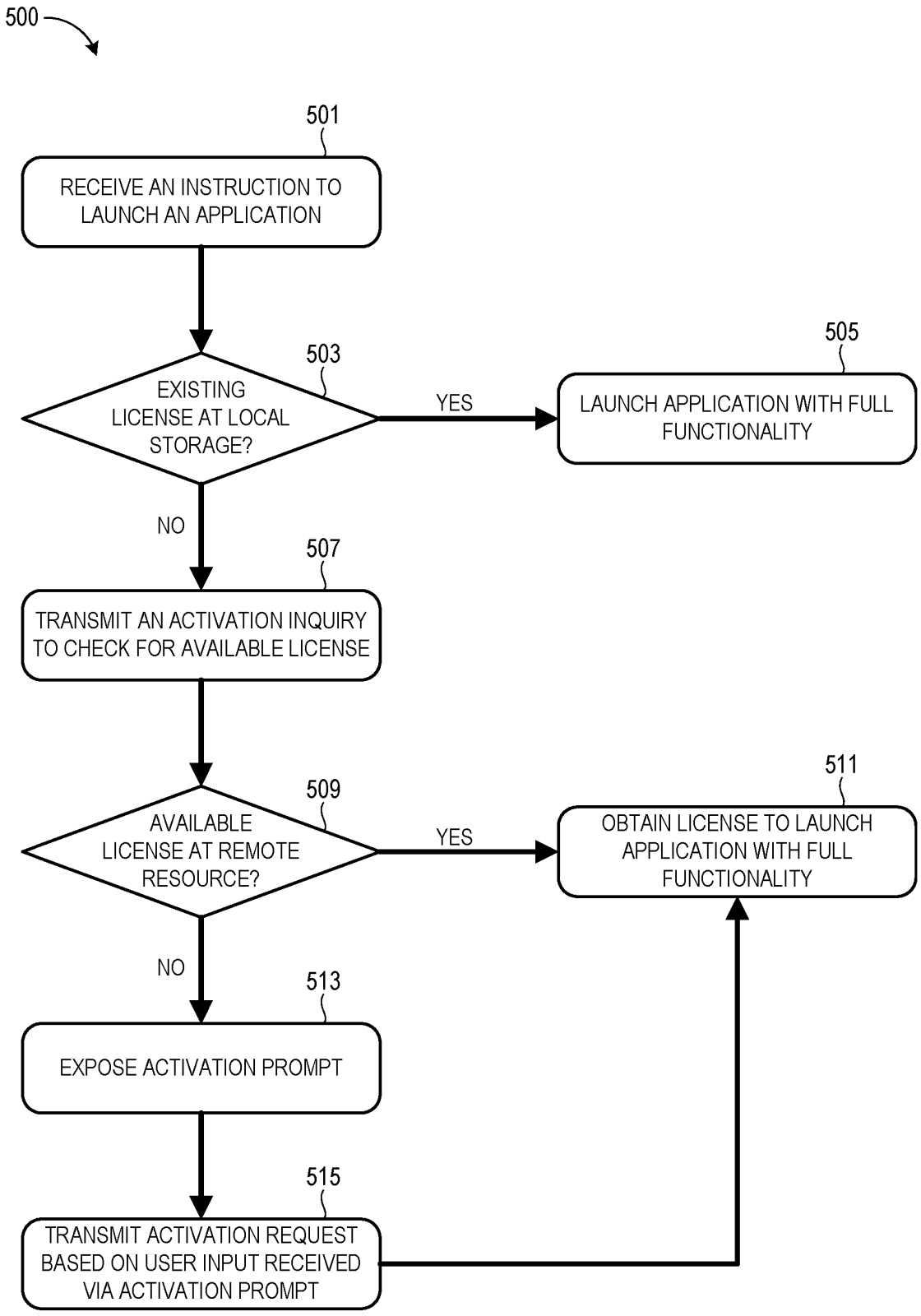
500
501
RECEIVE AN INSTRUCTION TO LAUNCH AN APPLICATION
503
EXISTING LICENSE AT LOCAL STORAGE?
YES
505
LAUNCH APPLICATION WITH FULL FUNCTIONALITY
NO
507
TRANSMIT AN ACTIVATION INQUIRY TO CHECK FOR AVAILABLE LICENSE
509
AVAILABLE LICENSE AT REMOTE RESOURCE?
YES
511
OBTAIN LICENSE TO LAUNCH APPLICATION WITH FULL FUNCTIONALITY
NO
513
EXPOSE ACTIVATION PROMPT
515
TRANSMIT ACTIVATION REQUEST BASED ON USER INPUT RECEIVED VIA ACTIVATION PROMPT

FIGURE 5

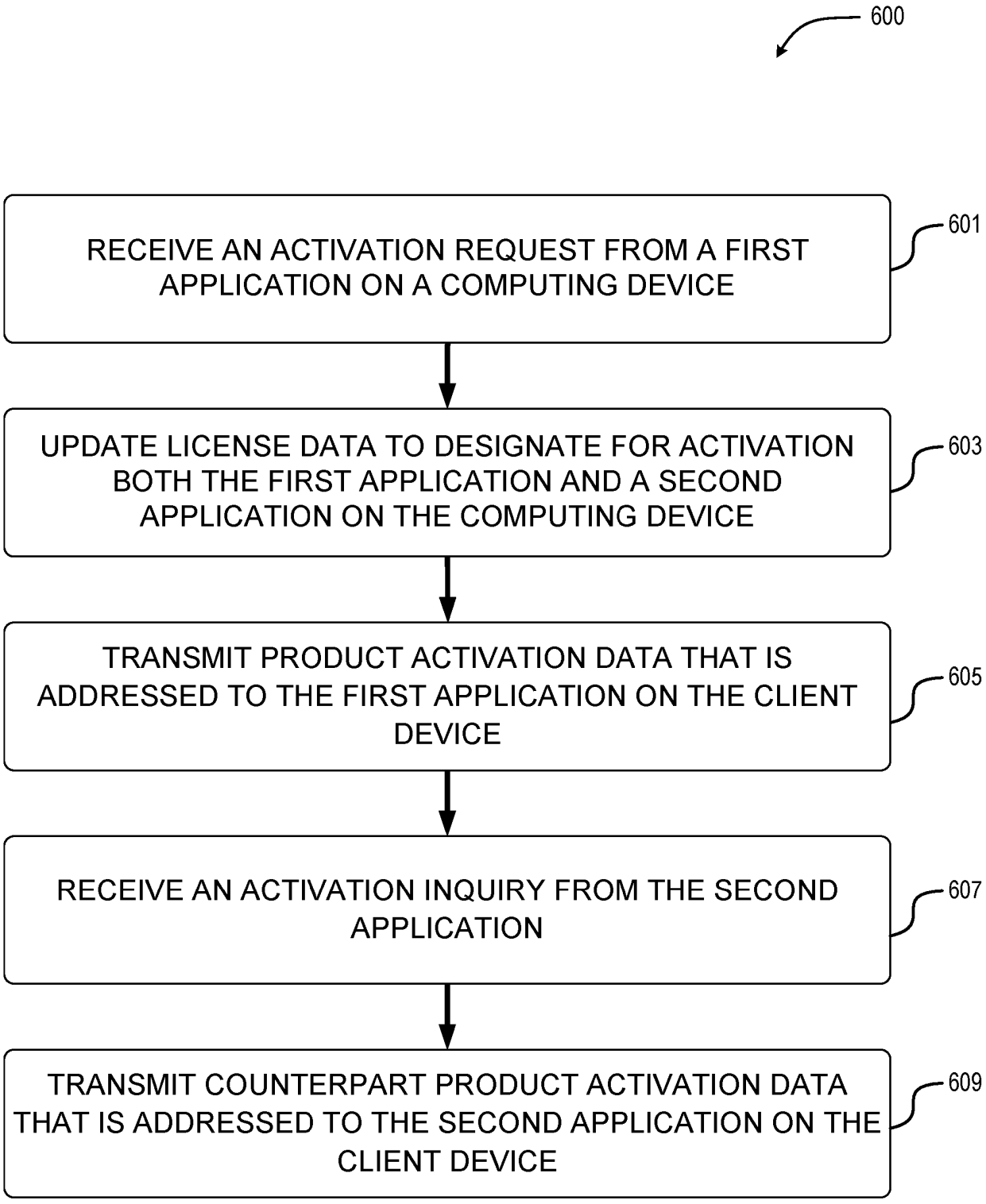
600
RECEIVE AN ACTIVATION REQUEST FROM A FIRST APPLICATION ON A COMPUTING DEVICE
601
UPDATE LICENSE DATA TO DESIGNATE FOR ACTIVATION BOTH THE FIRST APPLICATION AND A SECOND APPLICATION ON THE COMPUTING DEVICE
603
TRANSMIT PRODUCT ACTIVATION DATA THAT IS ADDRESSED TO THE FIRST APPLICATION ON THE CLIENT DEVICE
605
RECEIVE AN ACTIVATION INQUIRY FROM THE SECOND APPLICATION
607
TRANSMIT COUNTERPART PRODUCT ACTIVATION DATA THAT IS ADDRESSED TO THE SECOND APPLICATION ON THE CLIENT DEVICE
609

FIGURE 6

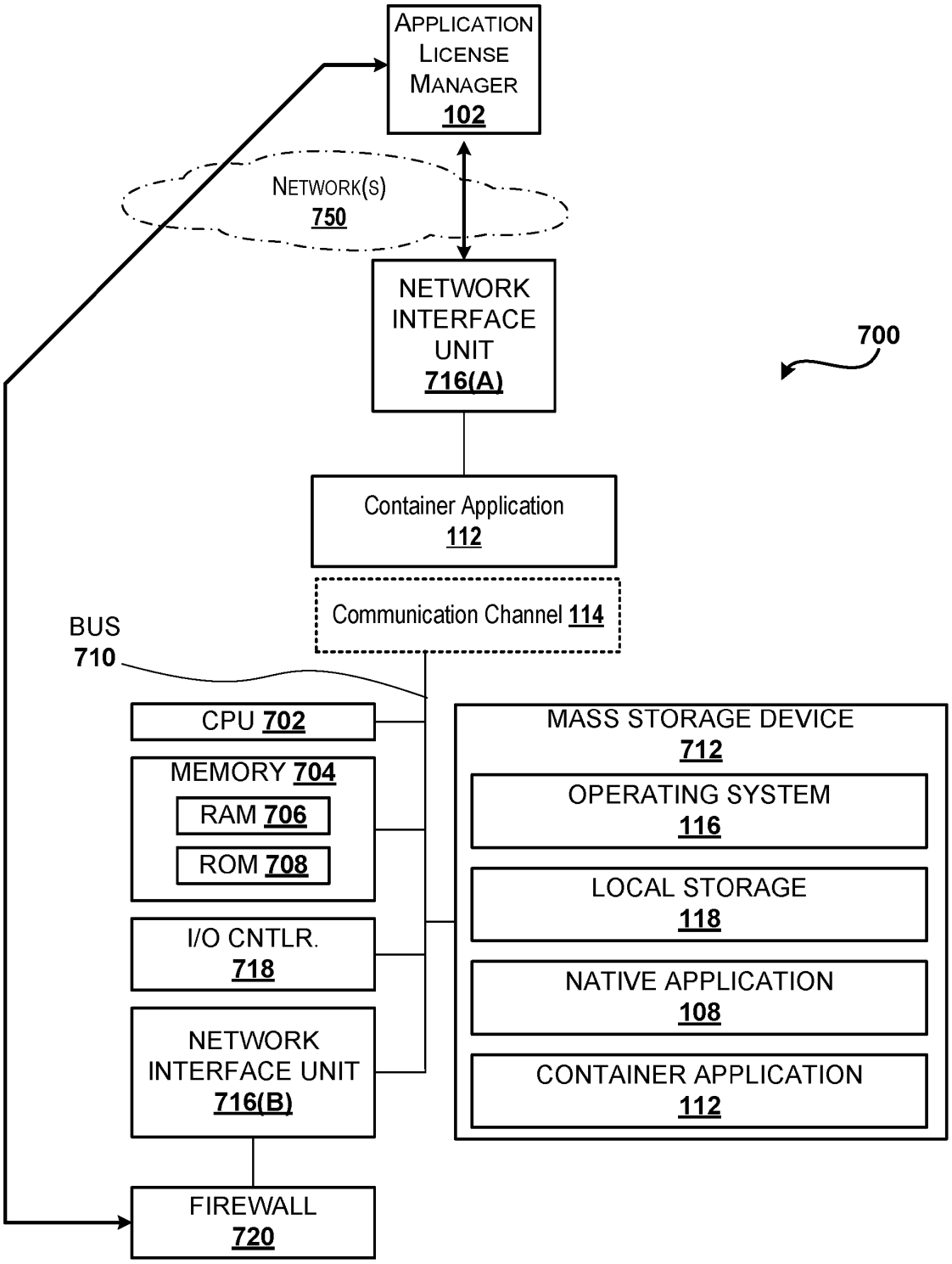
APPLICATION LICENSE MANAGER 102
NETWORK(S) 750
NETWORK INTERFACE UNIT 716(A)
700
Container Application 112
Communication Channel 114
BUS 710
CPU 702
MEMORY 704
RAM 706
ROM 708
I/O CNTLR. 718
NETWORK INTERFACE UNIT 716(B)
FIREWALL 720
MASS STORAGE DEVICE 712
OPERATING SYSTEM 116
LOCAL STORAGE 118
NATIVE APPLICATION 108
CONTAINER APPLICATION 112

FIGURE 7

REFERENCES CITED IN THE DESCRIPTION

This list of references cited by the applicant is for the reader's convenience only. It does not form part of the European patent document. Even though great care has been taken in compiling the references, errors or omissions cannot be excluded and the EPO disclaims all liability in this regard.

Patent documents cited in the description

- US 8763159 B1 **[0004]**