(12) INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
23 April 2015 (23.04.2015)

WIPO | PCT

(10) International Publication Number
WO 2015/057188 Al

(51) International Patent Classification:
G06F 3/048 (2013.01)

(21) International Application Number:
PCT/US20 13/064893

(22) International Filing Date:
14 October 2013 (14.10.201 3)

(25) Filing Language: English

(26) Publication Language: English

(71) Applicant: **HEWLETT-PACKARD DEVELOPMENT COMPANY, L.P.** [US/US]; 11445 Compaq Center Drive West, Houston, Texas 77070 (US).

(72) Inventors: **KAMATH, Harish B**; Sy.No. 192, Whitefield Road, Mahadevapura Post, Bangalore, Karnataka 560048 (IN). **SAMUEL, Sunil;** Sy.No. 192, Whitefield Road, Mahadevapura Post, Bangalore, Karnataka 560048 (IN). **BADIGER, Aravind;** Sy.No. 192, Whitefield Road, Mahadevapura Post, Bangalore, Karnataka 560048 (IN). **NAYAK, Prashant B**; Sy.No. 192, Whitefield Road, Mahadevapura Post, Bangalore, Karnataka 560048 (IN).

(74) Agents: **BELL, Scott** et al; Hewlett-Packard Company, Intellectual Property Administration, 3404 E. Harmony Road, Mail Stop 35, Fort Collins, Colorado 80528 (US).

(81) **Designated States** *(unless otherwise indicated, for every kind of national protection available):* AE, AG, AL, AM, AO, AT, AU, AZ, BA, BB, BG, BH, BN, BR, BW, BY, BZ, CA, CH, CL, CN, CO, CR, CU, CZ, DE, DK, DM, DO, DZ, EC, EE, EG, ES, FI, GB, GD, GE, GH, GM, GT, HN, HR, HU, ID, IL, IN, IR, IS, JP, KE, KG, KN, KP, KR, KZ, LA, LC, LK, LR, LS, LT, LU, LY, MA, MD, ME, MG, MK, MN, MW, MX, MY, MZ, NA, NG, NI, NO, NZ, OM, PA, PE, PG, PH, PL, PT, QA, RO, RS, RU, RW, SA, SC, SD, SE, SG, SK, SL, SM, ST, SV, SY, TH, TJ, TM, TN, TR, TT, TZ, UA, UG, US, UZ, VC, VN, ZA, ZM, ZW.

(84) **Designated States** *(unless otherwise indicated, for every kind of regional protection available):* ARIPO (BW, GH, GM, KE, LR, LS, MW, MZ, NA, RW, SD, SL, SZ, TZ, UG, ZM, ZW), Eurasian (AM, AZ, BY, KG, KZ, RU, TJ, TM), European (AL, AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, HR, HU, IE, IS, IT, LT, LU, LV, MC, MK, MT, NL, NO, PL, PT, RO, RS, SE, SI, SK, SM, TR), OAPI (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, KM, ML, MR, NE, SN, TD, TG).

Declarations under Rule 4.17:

— *as to the identity of the inventor (Rule 4.1 7(i))*

— *as to applicant's entitlement to apply for and be granted a patent (Rule 4.1 7(H))*

Published:

— *with international search report (Art. 21(3))*

(54) **Title:** PACKAGE DEPENDENCY MAPS FOR DISTRIBUTED COMPUTING

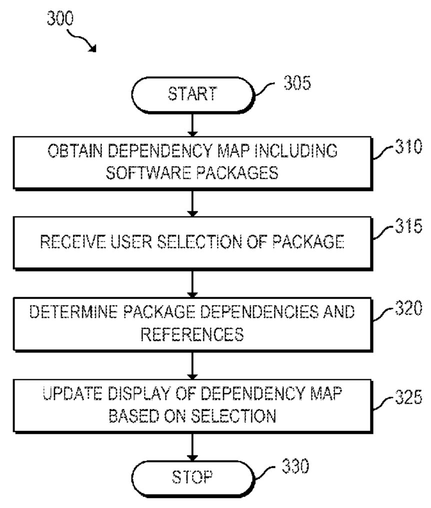

300
START
305
OBTAIN DEPENDENCY MAP INCLUDING SOFTWARE PACKAGES
310
RECEIVE USER SELECTION OF PACKAGE
315
DETERMINE PACKAGE DEPENDENCIES AND REFERENCES
320
UPDATE DISPLAY OF DEPENDENCY MAP BASED ON SELECTION
325
STOP
330

FIG. 3

(57) **Abstract:** Example embodiments relate to providing package dependency maps for distributed computing. In example embodiments, a dependency map is obtained that includes software packages, where each software package is associated with a site and a distributed node of a distributed system. In response to a user selection of a target package in the dependency map, a referenced subset on which the target package depends and a dependent subset that depend on the target package are determined. At this stage, a display of the dependency map is updated to include the target package, the referenced subset, and the dependent subset in a graphical hierarchy.

WO 2015/057188 A1

83602717

PACKAGE DEPENDENCY MAPS FOR DISTRIBUTED COMPUTING

BACKGROUND

[0001] Packages develop enclosures to software applications with a set of high availability rules that define where and when to run the software applications to satisfy the high availability rules. A package can have several dependencies that can be described in a hierarchical order. Complexity further compounds if a package is dependent with other packages configured in a clustered environment. For example, a cyclic configuration of dependent packages can cause undesirable effect in a distributed system if not addressed during the configuration phase. Typically, textual representations of package hierarchies are used to configure the dependencies and rules in infrastructure monitoring tools.

BRIEF DESCRIPTION OF THE DRAWINGS

[0002] The following detailed description references the drawings, wherein:

[0003] FIG. 1 is a block diagram of an example system for providing package dependency maps for distributed computing;

[0004] FIG. 2 is a block diagram of an example computing device including modules for providing package dependency maps for distributed computing;

[0005] FIG. 3 is a flowchart of an example method for execution by a computing device for providing package dependency maps for distributed computing;

[0006] FIG. 4 is a flowchart of an example method for execution by a computing device for displaying and updating a software dependency map for distributed computing; and

[0007] FIG. 5 is a block diagram of an example user interface for displaying a software dependency map for distributed computing.

DETAILED DESCRIPTION

[0008] As discussed above, package dependencies in a clustered environment can be difficult to express, especially if a cyclic configuration of dependent packages causes undesirable effects during the configuration phase. Managing and administering distributed systems can become particularly unwieldy as the number of software packages increases. Software packages may be configured to provide data and control dependencies across the packages. Properly monitoring such dependencies during configuration, deployment and production of a distributed system allows administrators to ensure that the packages remain operational and to address scheduling configuration requirements.

[0009] Example embodiments disclosed herein provide package dependency maps for distributed computing. For example, in some embodiments, a dependency map is obtained that includes software packages, where each software package is associated with a site and a distributed node of a distributed system. In response to a user selection of a target package in the dependency map, a referenced subset on which the target package depends and a dependent subset that depends on the target package are determined. At this stage, a display of the dependency map is updated to include the target package, the referenced subset, and the dependent subset in a graphical hierarchy.

[0010] In this manner, example embodiments disclosed herein graphically visualize software package dependencies in a hierarchical format. Specifically, by providing dependency maps that hierarchically and graphically show the dependencies between software packages, configuration of the packages in a distributed environment is facilitated by allowing the administrator to quickly identify dependency conflicts.

[001 1] Referring now to the drawings, FIG. 1 is a block diagram of an example computing device 100 for providing package dependency maps for distributed computing. The computing device 100 can include a processor 110 such as a processing unit in a notebook computer, a desktop computer, an all-in-one system, a tablet computing device; multiple computing devices in a distributed system; or any other electronic device suitable for providing package dependency maps for distributed computing. In the embodiment of FIG. 1, computing device 100 may also include an interface 115 and a machine-readable storage medium 120.

[0012] Processor 110 may be one or more central processing units (CPUs), microprocessors, and/or other hardware devices suitable for retrieval and execution of instructions stored in machine-readable storage medium 120. In some cases, processor 110 may include multiple CPUs that are distributed across multiple computing devices. Processor 110 may fetch, decode, and execute instructions 122, 124, 126, 128 to enable providing package dependency maps for distributed computing. As an alternative or in addition to retrieving and executing instructions, processor 110 may include one or more electronic circuits including a number of electronic components for performing the functionality of one or more of instructions 122, 124, 126, 128.

[0013] Interface 115 may include a number of electronic components for communicating with other computing devices in the distributed system. For example, interface 115 may be an Ethernet interface, a Universal Serial Bus (USB) interface, an IEEE 1394 (FireWire) interface, an external Serial Advanced Technology Attachment (eSATA) interface, or any other physical connection interface suitable for communication with a client device. Alternatively, interface 115 may be a wireless interface, such as a wireless local area network (WLAN) interface or a near-field communication (NFC) interface. In operation, as detailed below, interface 115 may be used to send and receive data, such as dependency data and status data, to and from corresponding interfaces of other computing devices in the distributed system.

[0014] Machine-readable storage medium 120 may be any electronic, magnetic, optical, multiple storage devices in a distributed system, or other physical storage device that stores executable instructions. Thus, machine-readable storage medium 120 may be, for example, Random Access Memory (RAM), an Electrically-Erasable Programmable Read-Only Memory (EEPROM), a storage drive, an optical disc, and the like. As described in detail below, machine-readable storage medium 120 may be encoded with executable instructions for providing package dependency maps for distributed computing. In some cases, the storage device that stores executable instructions. Thus, machine 120 may be accessible as a cloud resource, where an installation package including the executable instructions or output of the executable instructions may be obtained and used by processor 110.

[0015] Dependency map obtaining instructions 122 may obtain a dependency map for software packages in a distributed system. For example, a dependency map may be obtained from an application or module configured to automatically generate dependency maps for software packages. In another example, the dependency map may be generated manually by an administrator for use by processor 110. A software package defines an enclosure to a software application with a set of high availability rules. Rules may describe where and when to run the software application to satisfy the high availability requirements. A dependency map may be a graphical presentation of software packages within the hierarchy of a distributed system. Specifically, the dependency map may show dependencies between and locations of software packages within the distributed system. A dependency of a software package may describe a reliance on modules or services of a related software package. For example, a software package may rely on an application programming interface (API) of a related software package to perform requested operations. In this example, the software package may be inactivated if the related software package is unavailable.

[0016] Package selecting instructions 124 may process user interface selections performed by an administrator. For example, the administrator may select a software package from a list of related software packages to request a dependency map. In this example, the request may be processed to determine parameters for generating the dependency map of the software package. The parameters may include, but are not limited to, a package identifier, a node identifier that identifies the node on which the software package is installed, a map format that describes the layout and information to be included in the dependency map, etc.

[0017] Package dependency determining instructions 126 may identify dependent packages and referenced packages of the software package for the requested dependency map. A dependent package is a package that relies on functionality in the software package. A referenced package is a package that provides functionality that is used by the software package. For example, the dependent and referenced packages may be identified by using the package identifier to query a lookup table that includes data records describing dependencies between software packages. The dependent and referenced packages may be collectively referred to as the dependency requirements of the software package.

[0018] Dependency map display instructions 128 may render the dependency map for displaying on a display device (not shown). For example, the dependency map may be rendered as described with respect to FIG. 5 below. The rendering of the dependency map may include a graphical hierarchy that shows the dependencies between software packages in the distributed system. In some cases, the rendering may also include other information related to the software packages such as status information (e.g., availability), node information, etc. The administrator may use the dependency map to reconfigure the distributed system by, for example, resolving conflicts between software packages.

[0019] FIG. 2 is a block diagram of an example computing device 200 including modules for providing package dependency maps for distributed computing. As with processor 110 of FIG. 1, computing device 200 may be any electronic device suitable for providing dependency maps.

[0020] As illustrated in FIG. 2 and described in detail below, computing device 200 may also include a number of modules 210-228. Each of the modules may include a series of instructions encoded on a machine-readable storage medium and executable by a processor (not shown) of computing device 200. In addition or as an alternative, each module 210-228 may include one or more hardware devices including electronic circuitry for implementing the functionality described below. In some cases, the modules 210-228 may be implemented as a distributed system across multiple computing devices.

[0021] Monitoring module 210 may be configured to monitor elements of a distributed system. Examples of elements of a distributed system include software packages, nodes (e.g., servers, routers, client devices, etc.), etc. Monitoring module 210 may include modules as discussed below for providing functionality to dependency map module 220.

[0022] Node interface 212 may communicate with nodes in the distributed system. Specifically, node interface 212 may obtain information such as dependencies between software packages and status information (e.g., availability) from each of the nodes. For example, node interface 212 may access an API exposed by a node to identify dependencies between shared libraries and services installed on the node. In another example, node interface 212 may receive notifications that describe the software packages from the nodes.

[0023] Node status module 214 may process status information from the nodes. Status information collected by node interface 212 may be used to track the status (e.g., active, inactive, etc.) of each node in the distributed system. For example, the status of each node may be periodically determined and stored for use by the dependency map module 220. In another example, node status module 214 may monitor the status of each node and then notify dependency map module 220 when a change in status occurs.

[0024] Node conditional module 216 may manage conditional rules for the distributed system. For example, if a node becomes inactive, a conditional rule may specify that a substitute software package on another node should be used until the node becomes active again. In another example, a spillover node may be activated when the load of a primary node reaches a preconfigured threshold. Node conditional module 216 may notify dependency map module 220 of changes in the distributed system when conditional rules are triggered.

[0025] Dependency map module 220 may process and generate a display of dependency maps of software packages. Although the components of dependency map module 220 are described in detail below, additional details regarding an example implementation of dependency map module 220 are provided above with respect to instructions 122-128 of FIG. 1.

[0026] Dependency map managing module 222 may manage dependency maps of software packages. Specifically, dependency map managing module 222 may obtain dependency maps and provide them for use by the other modules of dependency map module 220. For example, dependency maps may be generated and then stored on computing device 200 as a file, database entries, etc. In this example, dependency map module 222 may load a dependency map stored on computing device 200 so that it may be accessed by a user of computing device 200. Dependency map managing module 222 may process the dependency map to determine dependencies of a corresponding software package, where each of the dependencies may be represented in a hierarchy of software packages within a distrusted system. Further, dependency map managing module 222 may also determine status information associated with each of the software packages, where each of the status information may be graphically shown within nodes that represent software packages.

[0027] Package dependency module 224 may monitor dependencies and status information of software packages to update dependency maps. For example, notifications of dependency changes from monitoring module 210 may be processed to update dependency maps in storage or in real-time. Package dependency module 224 may also analyze dependencies to identify conflicts for including in renderings of the dependency maps. .Examples of conflicts include resource requirement conflicts, a software package preventing another package from initializing properly, a missing referenced package, cyclic dependencies, etc.

[0028] Dependency configuration module 226 may manage user configurations of software packages in the context of a dependency map. Specifically, requests from an administrator reviewing a rendered dependency map may be processed to update dependencies in the map. For example, if an administrator sees that a referenced software package is inactive, he may request that an alternative package on a different node be referenced instead. The graphical hierarchy of the dependency map allows the administrator to quickly identify and resolve conflicts between software packages.

[0029] Map display module 228 may generate a display of and allow a user to interact with dependency maps. Specifically, the dependency map may be displayed as a graphical hierarchy that is representative of the dependencies between software packages. Further, status information may be represented in package nodes within the hierarchy. Map display module 228 may allow the user to select and modify software packages within the display of the dependency map. Map display module 228 may also be updated to reflect changes in dependencies of the software packages. For example, a software package being inactivated may be detected in real-time and reflected in the dependency map for the administrator's review.

[0030] May display module 228 may also allow the user to select and review information for related software packages. For example, if a user selects a referenced or dependent package, map display module 228 may display a dependency map for the selected package.

[0031] Generating a display may include directing a graphics device to display the dependency map or generating data that is processed and displayed by a client device. For example, map display module 228 may generate a display of a dependency map by providing instructions to a graphics card for displaying the dependency map on a display device such as a monitor. In another example, map display module 228 may generate display data that is transmitted to and then used by a client device to display the dependency map. In yet another example, computing device 200 may include a graphics processing unit (now shown) that is used by map display module 228 to generate a display of the dependency map.

[0032] FIG. 3 is a flowchart of an example method 300 for execution by a computing device 100 for providing package dependency maps for distributed computing. Although execution of method 300 is described below with reference to computing device 100 of FIG. 1, other suitable devices for execution of method 300 may be used, such as computing device 200 of FIG. 2. Method 300 may be implemented in the form of executable instructions stored on a machine-readable storage medium, such as storage resource 120, and/or in the form of electronic circuitry.

[0033] Method 300 may start in block 305 and continue to block 310, where computing device 100 obtains a dependency map that includes software packages. For example, computing device 100 may consult a lookup table to determine dependencies between software packages. In this example, the dependencies of the software packages in a distributed system may be initially determined and displayed in a textual format for review by an administrator.

[0034] In block 315, a user selection of a software package in the distributed system is received. The user selection may also specify a format for rendering a graphical hierarchy for the selected package. Next, in block 320, package dependencies and references are identified for the selected package. In this example, the dependencies and references are the same as shown in the textual display of the dependency map.

[0035] In block 325, the display of the dependency map may be updated based on the selection. Specifically, the graphical hierarchy may be rendered on the user interface so that the user can effectively review the dependencies related to the selected software package. Method 300 may subsequently proceed to block 330, where method 300 may stop.

[0036] FIG. 4 is a flowchart of an example method 400 for execution by a computing device 200 for displaying and updating a software dependency map for distributed computing. Although execution of method 400 is described below with reference to computing device 200 of FIG. 2, other suitable devices for execution of method 400 may be used, such as computing device 100 of FIG. 1. Method 400 may be implemented in the form of executable instructions stored on a machine-readable storage medium and/or in the form of electronic circuitry.

[0037] Method 400 may start in block 405 and proceed to block 410, where computing device 200 obtains a dependency map that includes software packages for a distributed system. In block 415, dependencies between each of the software packages may be determined and, for example, prepared for a textual format. In block 420, a display of the textual format of the dependency map is generated and displayed in a table that includes a row for each software package.

[0038] In block 425, computing device 200 determines if a package has been selected by the administrator. If a package is not selected, method 400 may proceed to block 450 and stop. If a package is selected, computing device 200 updates the display to include a graphical hierarchy for the selected package in block 430. The administrator may change the graphical hierarchy presented by selecting different packages from the list of packages shown in the table.

[0039] In block 435, computing device 200 determines if a package has been modified. If a package is not modified, method 400 may proceed to block 450 and stop. If a package is modified, the dependencies of the selected package are updated based on the modification in block 440. For example, the administrator may add or remove referenced packages of the selected package. In this example, the dependency requirements (i.e., referenced and dependent packages) of each of the modified referenced packages are verified to ensure that no conflicts have been introduced by the modification. In block 445, the display of the dependency map is updated to reflect the modified package. In this example, the referenced packages may be updated to include new referenced packages added by the administrator. Method 400 may subsequently proceed to block 450, where method 400 may stop.

[0040] FIG. 5 is a block diagram of an example user interface 500 for displaying a software dependency map for distributed computing. As depicted, the user interface 500 includes a title bar 505 identifying the dependency map and a close icon 510 that may be used by the user to exit the application. The user interface 500 also includes navigation components: a back button 515, a forward button 520, a packages button 525, a nodes button 530, and a sites button 535. As shown in FIG. 5, the packages button 525 may be used to navigate to a display of the current software packages loaded in the application. The nodes button 530 may be used to navigate to a list of nodes in the distributed system. A node may be selected from the list to load a display of the software packages installed on the selected node. The sites button 535 may be used to navigate to a list of sites of the distributed system. A site may be selected from the list to load a display of the software packages installed at the selected site. A site may be a physical location where one or more of the nodes of the distributed system are located.

[0041] The current package of interest 537 is shown with an add dependencies button 539 for adding additional dependent packages of the current package 537. Dependent packages of the current package 537 are shown in a table with a package name column 540, a node column 545, a status column 550, a dependent packages column 555, a referenced packages column 560, and an actions column 565. The table includes rows for each of the dependent packages of the current package 537, where a package row 575 and a selected package row 580 are shown. Each of the package rows 575, 580 includes a close button 577 and an add dependent package button 579. Selecting the close button 577 removes the corresponding package as a dependent package from the current package 537. Selecting the add dependent package button 579 adds a dependent package to the corresponding package. The user interface 500 also includes a scroll bar for 570 for browsing the list of package rows 575, 580. In this example, an alert 582 is shown for package 6 because it has a cyclic dependency. Specifically, package 6 both references and depends on package 7, which may tightly couple the packages such that they can no longer be used independently. The selected package row 580 shows information for package 3 585, which is displayed in the rendered graphical hierarchy.

[0042] The hierarchy for package 3 585 includes dependency edges 597 and package nodes 590. Each of the package nodes 590 corresponds to a package in the distributed system and includes a status indicator 595 that shows the current status of the package. The dependency edges 597 show dependencies with the dependent and referenced packages of package 3 585 as also shown in the selected package row 580.

[0043] The foregoing disclosure describes a number of example embodiments for dependency maps for distributed computing. In this manner, the embodiments disclosed herein enable administrators to rapidly assess package dependencies to identify and resolve conflicts by providing a graphical hierarchy of the dependencies.

CLAIMS

We claim:

1. A system for providing package dependency maps for distributed computing, the system comprising:

a processor to:

obtain a dependency map comprising a plurality of software packages, wherein each of the plurality of software packages is associated with one of a plurality of sites and one of a plurality of distributed nodes;

in response to a user selection of a target package of the plurality of software packages in the dependency map, determine a referenced subset of the plurality of software packages on which the target package depends and a dependent subset of the plurality of software packages that depend on the target package; and

update a display of the dependency map that comprises the target package, the referenced subset, and the dependent subset in a graphical hierarchy.

2. The system of claim 1, wherein the processor is further to:

detect a modification of a modified package of the plurality of software packages; and

modify the display of the dependency map based on the modification.

3. The system of claim 2, wherein the modification is an availability status of the modified package, and wherein the dependency map shows that a failover package of the plurality of software packages is activated at a backup node of the plurality of distributed nodes in response to the modification.

4. The system of claim 1, wherein the processor is further to:

receive a reconfiguration request for the target package of the plurality of software packages; and

in response to the reconfiguration request, determine that the reconfiguration request does not violate dependency requirements of the referenced subset and the dependent subset.

5. The system of claim 1, wherein the graphical hierarchy comprises a package node for each of the plurality of software packages and a plurality of dependency edges to connect the target package to the referenced subset and the dependent subset.

6. The system of claim 5, wherein the display of the dependency map further comprises a status indicator for each of the plurality of software packages that describes a status of a corresponding package.

7. A method for providing package dependency maps for distributed computing, the method comprising:

obtaining a dependency map comprising a plurality of software packages, wherein each of the plurality of software packages is associated with one of a plurality of sites and one of a plurality of distributed nodes;

in response to a user selection of a target package of the plurality of software packages in the dependency map, determining a referenced subset of the plurality of software packages on which the target package depends and a dependent subset of the plurality of software packages that depend on the target package; and

updating a display of the dependency map that comprises the target package, the referenced subset, and the dependent subset in a graphical hierarchy, wherein the graphical hierarchy comprises a package node for each of the plurality of software packages and a plurality of dependency edges to connect the target package to the referenced subset and the dependent subset.

8. The method of claim 7, further comprising:

detecting a modification of a modified package of the plurality of software packages; and

modifying the display of the dependency map based on the modification.

9. The method of claim 8, wherein the modification is an availability status of the modified package, and wherein the dependency map shows that a failover package of the plurality of software packages is activated at a backup node of the plurality of distributed nodes in response to the modification.

10. The method of claim 7, further comprising:

receiving a reconfiguration request for the target package of the plurality of software packages; and

in response to the reconfiguration request, determining that the reconfiguration request does not violate dependency requirements of the referenced subset and the dependent subset.

11. The method of claim 7, wherein the display of the dependency map further comprises a status indicator for each of the plurality of software packages that describes a status of a corresponding package.

12. A non-transitory machine-readable storage medium encoded with instructions executable by a processor for providing package dependency maps for distributed computing, the machine-readable storage medium comprising instructions to:

obtain a dependency map comprising a plurality of software packages, wherein each of the plurality of software packages is associated with one of a plurality of sites and one of a plurality of distributed nodes;

in response to a user selection of a target package of the plurality of software packages in the dependency map, determine a referenced subset of the plurality of software packages on which the target package depends and a dependent subset of the plurality of software packages that depend on the target package;

update a display of the dependency map that comprises the target package, the referenced subset, and the dependent subset in a graphical hierarchy.

receive a reconfiguration request for the target package of the plurality of software packages; and

in response to the reconfiguration request, determine that the reconfiguration request does not violate dependency requirements of the referenced subset and the dependent subset.

13. The non-transitory machine-readable storage medium of claim 12, wherein the instructions are further to:

detect a modification of a modified package of the plurality of software packages; and

modify the display of the dependency map based on the modification.

14. The non-transitory machine-readable storage medium of claim 13, wherein the modification is an availability status of the modified package, and wherein the dependency map shows that a failover package of the plurality of software packages is activated at a backup node of the plurality of distributed nodes in response to the modification.

15. The non-transitory machine-readable storage medium of claim 12, wherein the graphical hierarchy comprises a package node for each of the plurality of software packages and a plurality of dependency edges to connect the target package to the referenced subset and the dependent subset.

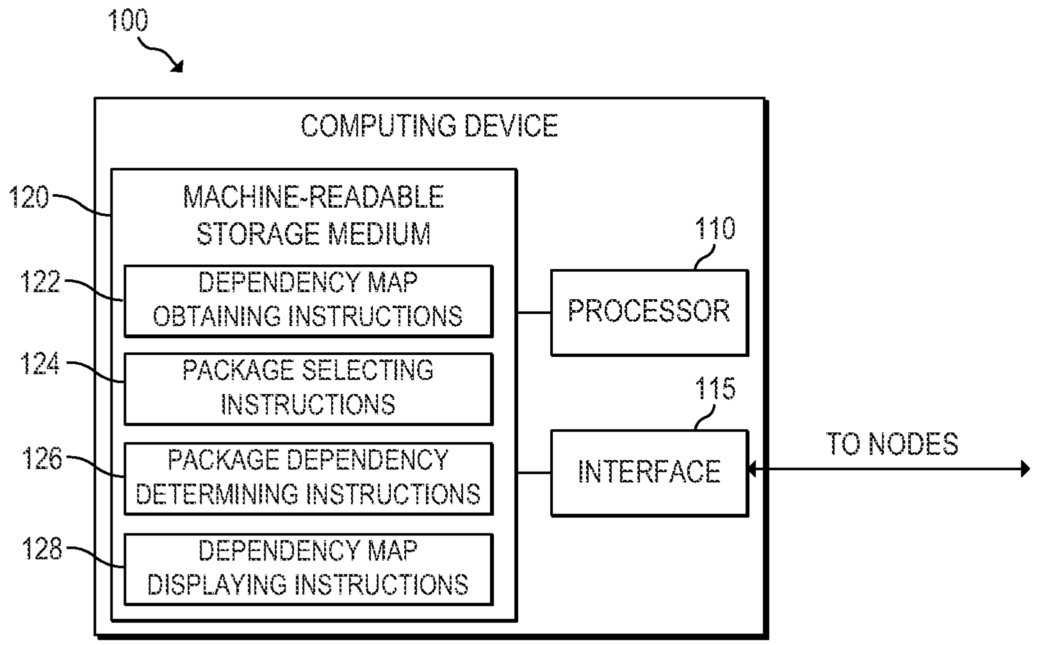

100
COMPUTING DEVICE
120
MACHINE-READABLE STORAGE MEDIUM
122
DEPENDENCY MAP OBTAINING INSTRUCTIONS
124
PACKAGE SELECTING INSTRUCTIONS
126
PACKAGE DEPENDENCY DETERMINING INSTRUCTIONS
128
DEPENDENCY MAP DISPLAYING INSTRUCTIONS
110
PROCESSOR
115
INTERFACE
TO NODES

FIG. 1

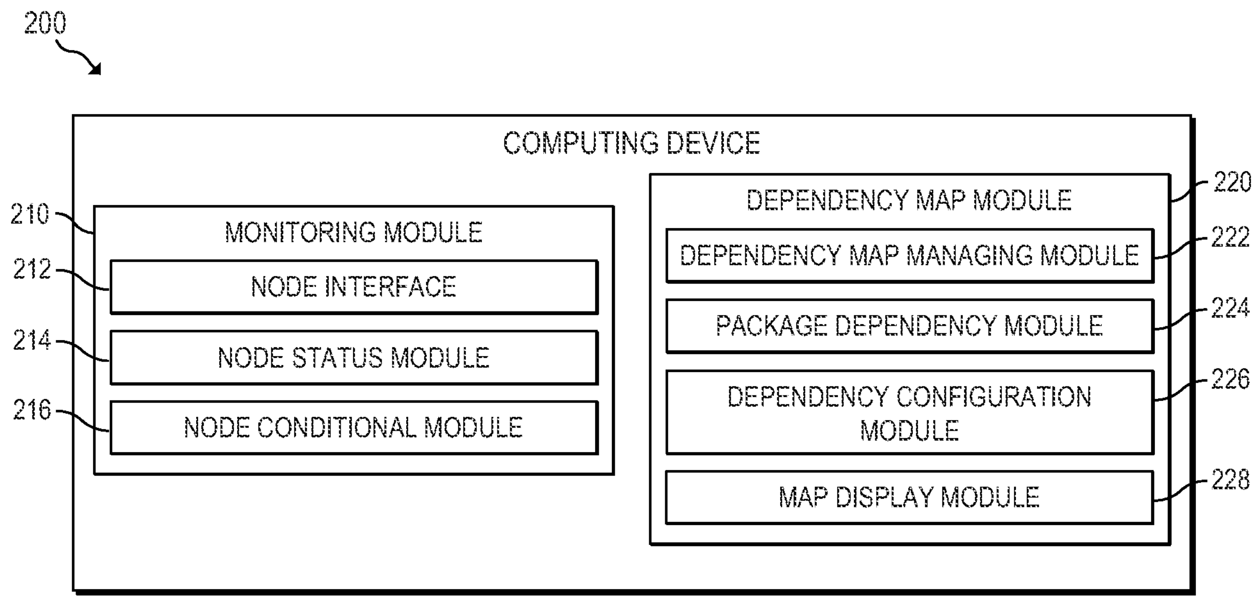

200
COMPUTING DEVICE
210
MONITORING MODULE
212
NODE INTERFACE
214
NODE STATUS MODULE
216
NODE CONDITIONAL MODULE
220
DEPENDENCY MAP MODULE
222
DEPENDENCY MAP MANAGING MODULE
224
PACKAGE DEPENDENCY MODULE
226
DEPENDENCY CONFIGURATION MODULE
228
MAP DISPLAY MODULE

FIG. 2

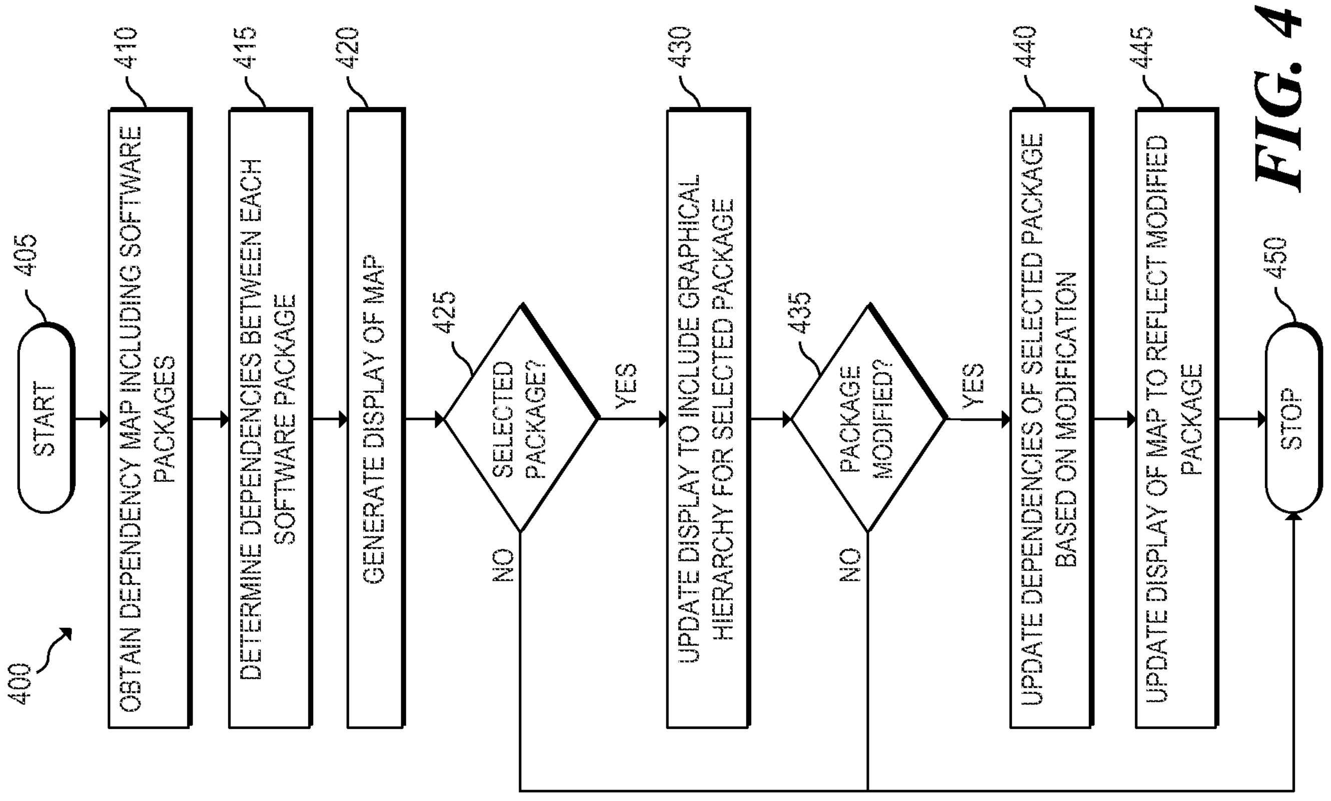

400
START
405
OBTAIN DEPENDENCY MAP INCLUDING SOFTWARE PACKAGES
410
DETERMINE DEPENDENCIES BETWEEN EACH SOFTWARE PACKAGE
415
GENERATE DISPLAY OF MAP
420
SELECTED PACKAGE?
425
YES
NO
UPDATE DISPLAY TO INCLUDE GRAPHICAL HIERARCHY FOR SELECTED PACKAGE
430
PACKAGE MODIFIED?
435
YES
NO
UPDATE DEPENDENCIES OF SELECTED PACKAGE BASED ON MODIFICATION
440
UPDATE DISPLAY OF MAP TO REFLECT MODIFIED PACKAGE
445
STOP
450

FIG. 4

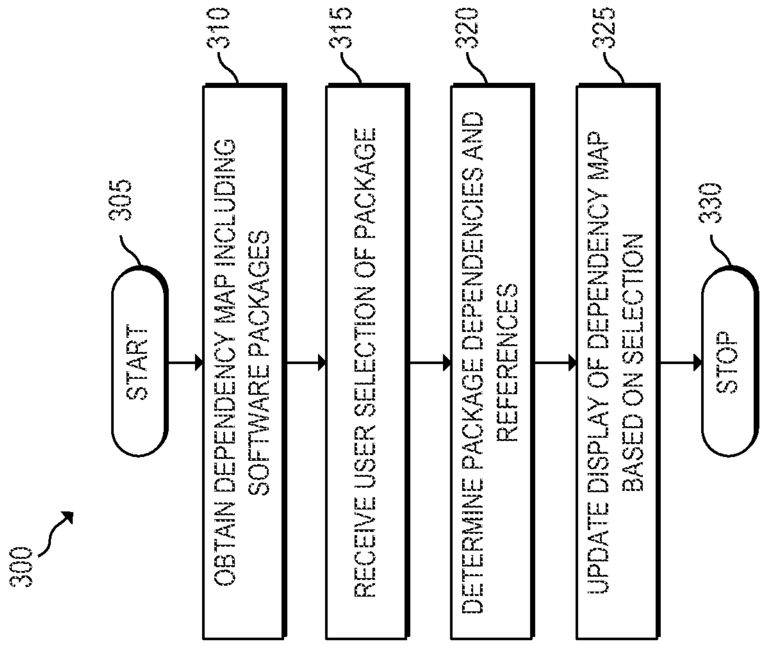
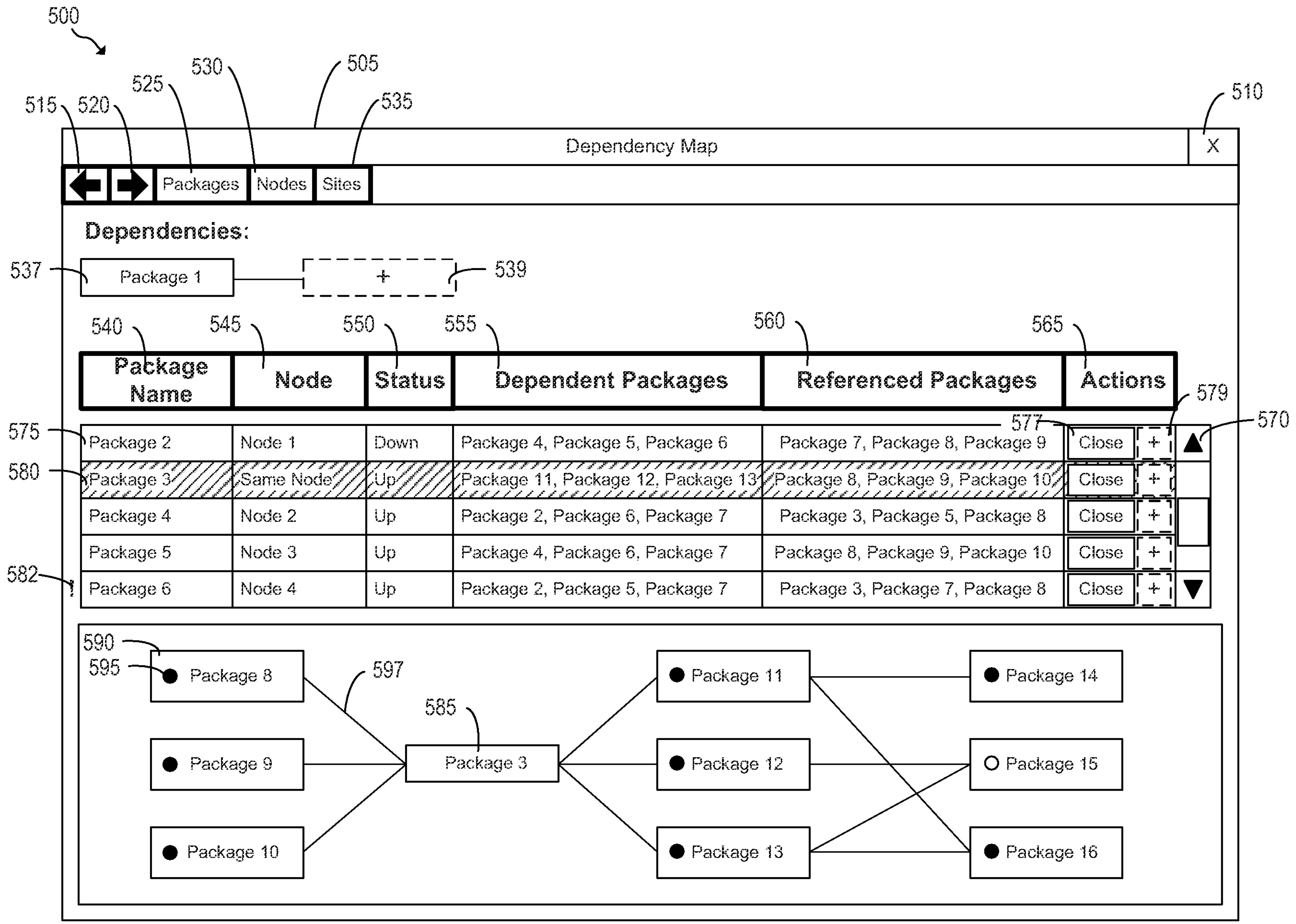

300
START
305
OBTAIN DEPENDENCY MAP INCLUDING SOFTWARE PACKAGES
310
RECEIVE USER SELECTION OF PACKAGE
315
DETERMINE PACKAGE DEPENDENCIES AND REFERENCES
320
UPDATE DISPLAY OF DEPENDENCY MAP BASED ON SELECTION
325
STOP
330

FIG. 3



Package Name	Node	Status	Dependent Packages	Referenced Packages	Actions
Package 2	Node 1	Down	Package 4, Package 5, Package 6	Package 7, Package 8, Package 9	Close +
Package 3	Same Node	Up	Package 11, Package 12, Package 13	Package 8, Package 9, Package 10	Close +
Package 4	Node 2	Up	Package 2, Package 6, Package 7	Package 3, Package 5, Package 8	Close +
Package 5	Node 3	Up	Package 4, Package 6, Package 7	Package 8, Package 9, Package 10	Close +
Package 6	Node 4	Up	Package 2, Package 5, Package 7	Package 3, Package 7, Package 8	Close +

FIG. 5

INTERNATIONAL SEARCH REPORT

International application No.
PCT/US2013/064893

A. CLASSIFICATION OF SUBJECT MATTER

G06F 3/048(2006.01)i

According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)
G06F 3/048; G06F 11/30; G06F 9/445; G06F 11/34; G06F 9/44; G06F 17/30; G06F 15/177

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched
Korean utility models and applications for utility models
Japanese utility models and applications for utility models

Electronic data base consulted during the international search (name of data base and, where practicable, search terms used)
eKOMPASS(KIPO internal) & Keywords dependency, map, interface, distributed computing

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category *	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
Y	US 2010-0131854 Al (MARK CAMERON LITTLE) 27 May 2010 See paragraphs [0001], [0012], [0030], [0034], [0036H0037], [0044]; and figures 2A-2C, 5.	1-2, 5-8, 11
A		3-4, 9-10, 12-15
Y	US 2008-0201705 Al (MICHAEL J. WCOKEY) 21 August 2008 See paragraphs [0002], [0060], [0063], [0086], [0109]; claim 101; and figure 6.	1-2, 5-8, 11
A		3-4, 9-10, 12-15
A	US 2009-0144305 Al (MARK CAMERON LITTLE) 04 June 2009 See paragraphs [0001], [0018], [0036], [0041], [0043]- [0045]; and figures 2-4.	1-15
A	KR 10-2012-0030320 A (COMPUTER ASSOCIATES THINK, INC.) 28 March 2012 See paragraphs [0001], [0005], [0017H0020]; and figures 2-3.	1-15
A	JP 2004-103015 A (INTERNATL BUSINESS MACH CORP) 02 April 2004 See paragraphs [0001], [0015], [0018H0019]; and figure 2.	1-15

☐ Further documents are listed in the continuation of Box C.   ☒ See patent family annex.

* Special categories of cited documents:	
"A" document defining the general state of the art which is not considered to be of particular relevance	"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention
"E" earlier application or patent but published on or after the international filing date	"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone
"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)	"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art
"O" document referring to an oral disclosure, use, exhibition or other means	"&" document member of the same patent family
"P" document published prior to the international filing date but later than the priority date claimed	

Date of the actual completion of the international search 03 July 2014 (03.07.2014)	Date of mailing of the international search report **04 July 2014 (04.07.2014)**
Name and mailing address of the ISA/KR International Application Division Korean Intellectual Property Office **189** Cheongsa-ro, Seo-gu, Daejeon Metropolitan City, **302-701,** Republic of Korea Facsimile No. +82-42-472-7140	Authorized officer LEE, Dong Yun Telephone No. +82-42-481-8734

INTERNATIONAL SEARCH REPORT
Information on patent family members

International application No.
PCT/US2013/064893

Patent document cited in search report	Publication date	Patent family member(s)	Publication date
US 2010-0131854 Al	27/05/2010	US 8645837 B2	04/02/2014
US 2008-0201705 Al	21/08/2008	US 2011-191766 Al	04/08/2011
		us 2011-209142 Al	25/08/2011
		us 2011-214119 Al	01/09/2011
		us 2011-225461 Al	15/09/2011
		us 2011-225577 Al	15/09/2011
		us 2011-231836 Al	22/09/2011
		us 2011-231838 Al	22/09/2011
		us 2011-239212 Al	29/09/2011
		us 2011-246982 Al	06/10/2011
		us 2011-258619 Al	20/10/2011
		us 2012-144386 Al	07/06/2012
		us 2012-151468 Al	14/06/2012
		us 2012-151469 Al	14/06/2012
		us 8527979 B2	03/09/2013
		us 8533704 B2	10/09/2013
		us 8566819 B2	22/10/2013
		us 8589914 B2	19/11/2013
		us 8589915 B2	19/11/2013
		us 8621453 B2	31/12/2013
		us 8621454 B2	31/12/2013
		us 8631400 B2	14/01/2014
		us 8640123 B2	28/01/2014
		us 8645946 B2	04/02/2014
		us 8645947 B2	04/02/2014
		us 8719814 B2	06/05/2014
US 2009-0144305 Al	04/06/2009	us 2013-275487 Al	17/10/2013
		us 8464270 B2	11/06/2013
KR 10-2012-0030320 A	28/03/2012	EP 2431879 Al	21/03/2012
		JP 2012-074028 A	12/04/2012
		JP 5431430 B2	05/03/2014
		us 2012-0072887 Al	22/03/2012
		us 8490055 B2	16/07/2013
JP 2004-103015 A	02/04/2004	CN 1489078 A	14/04/2004
		KR 10-0546973 Bl	31/01/2006
		KR 10-2004-0023529 A	18/03/2004
		us 2004-0049509 Al	11/03/2004
		us 6847970 B2	25/01/2005