



(51) International Patent Classification:  
*G06F 9/44* (2006.01)

(21) International Application Number:  
PCT/US2012/057136

(22) International Filing Date:  
25 September 2012 (25.09.2012)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:  
61/541,059 29 September 2011 (29.09.2011) US  
13/423,035 16 March 2012 (16.03.2012) US

(71) Applicant: ORACLE INTERNATIONAL CORPORA-  
TION [US/US]; 500 Oracle Parkway, M/S 50p7, Redwood  
Shores, California 94065 (US).

(72) Inventors: LI, Zhenyu; No. 20 Chengfu Road 39-252,  
Haidian District, Beijing (CN). LIU, Lidan; No. 40 Huay-  
anbeili, Chaoyang District, Beijing (CN).

(74) Agents: MEYER, Sheldon, R. et al.; Fliesler Meyer LLP,  
650 California Street, Fourteenth Floor, San Francisco,  
California 94108 (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, IS, JP, KE, KG, KM, 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, 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,  
ML, MR, NE, SN, TD, TG).

Published:

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

(54) Title: SYSTEM AND METHOD FOR SUPPORTING AUTOMATICALLY DEPLOYING/UNDEPLOYING APPLICATION  
COMPONENTS IN A TRANSACTIONAL MIDDLEWARE MACHINE ENVIRONMENT

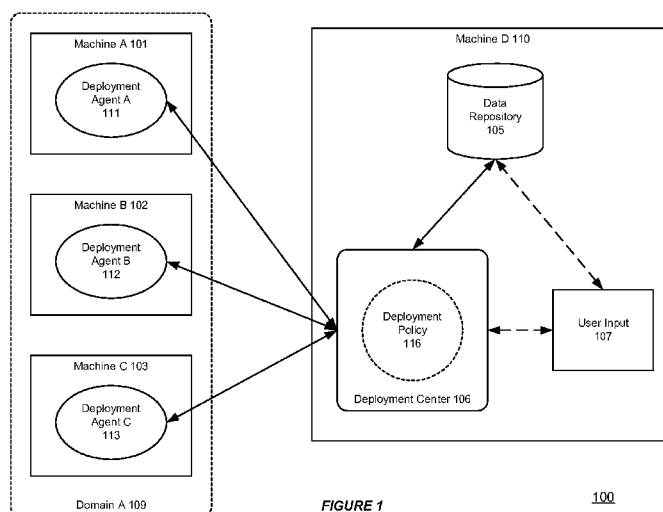


FIGURE 1

100

(57) Abstract: A system and method can support automatically deploying application components in a transactional middleware machine environment. A deployment center can receive one or more application packages, each of which contains binary files for one or more transactional servers and configuration information that describes relationship and parameters of the one or more transactional servers in the application package. The deployment center can further generate one or more distribution packages for each transactional middleware machine in the transactional middleware machine environment based on the one or more application packages. Then, the deployment center can deploy the one or more distribution packages to the plurality of transactional middleware machines in the transactional middleware machine environment.

**SYSTEM AND METHOD FOR SUPPORTING AUTOMATICALLY  
DEPLOYING/UNDEPLOYING APPLICATION COMPONENTS IN A TRANSACTIONAL  
MIDDLEWARE MACHINE ENVIRONMENT**

5

**Copyright Notice:**

[0001] A portion of the disclosure of this patent document contains material which is subject to copyright protection. The copyright owner has no objection to the facsimile reproduction by anyone of the patent document or the patent disclosure, as it appears in the Patent and  
10 Trademark Office patent file or records, but otherwise reserves all copyright rights whatsoever.

**Field of Invention:**

[0002] The present invention is generally related to computer systems and software such as middleware, and is particularly related to supporting a transactional middleware machine  
15 environment.

**Background:**

[0003] A transactional middleware system, or transaction oriented middleware, includes enterprise application servers that can process various transactions within an organization. With  
20 the developments in new technologies such as high performance network and multiprocessor computers, there is a need to further improve the performance of transactional middleware. These are the generally areas that embodiments of the invention are intended to address.

**Summary:**

[0004] Described herein is a system and method for supporting automatically deploying application components in a transactional middleware machine environment. A deployment center can receive one or more application packages, each of which contains binary files for one or more transactional servers and configuration information that describes relationship and parameters of the one or more transactional servers in the application package. The deployment  
30 center can further generate one or more distribution packages for each transactional middleware machine in the transactional middleware machine environment based on the one or more application packages. Then, the deployment center can deploy the one or more distribution packages to the plurality of transactional middleware machines in the transactional middleware machine environment.

35

**Brief Description of the Figures:**

[0005] **Figure 1** shows an illustration of a transactional middleware machine environment that supports a dynamic resource broker, in accordance with an embodiment of the invention.

[0006] **Figure 2** illustrates an exemplary flow chart for supporting automatically deploying/undeploying application components in a transactional middleware machine environment, in accordance with an embodiment of the invention.

[0007] **Figure 3** shows an illustration of the internal structure of an application package in a  
5 middleware machine environment, in accordance with an embodiment of the invention.

[0008] **Figure 4** shows an illustration of the internal structure of an application directory in a middleware machine environment, in accordance with an embodiment of the invention.

[0009] **Figure 5** shows an illustration of preparing an application package, in accordance with an embodiment of the invention.

10 [0010] **Figure 6** shows an illustration of creating a machine list in a GUI Console, in accordance with an embodiment of the invention.

[0011] **Figure 7** shows an illustration of uploading application packages, in accordance with an embodiment of the invention.

[0012] **Figure 8** shows an illustration of creating a domain in a transactional middleware  
15 environment, in accordance with an embodiment of the invention.

[0013] **Figure 9** shows an illustration of binding a physical machine to a virtual machine in a domain, in accordance with an embodiment of the invention.

[0014] **Figure 10** shows an illustration of adding an application package to a virtual machine in a domain, in accordance with an embodiment of the invention.

20 [0015] **Figure 11** shows an illustration of setting up multiple virtual machines in a domain, in accordance with an embodiment of the invention.

[0016] **Figure 12** shows an illustration of the internal structure of the data repository, in accordance with an embodiment of the invention.

[0017] **Figure 13** shows an illustration of the structure of an application directory, in  
25 accordance with an embodiment of the invention.

[0018] **Figure 14** shows more details of the transactional middleware machine environment that supports a dynamic resource broker in **Figure 1**, in accordance with an embodiment of the invention.

30 **Detailed Description:**

[0019] Described herein is a system and method for supporting a transactional middleware system that can take advantage of fast machines with multiple processors, and a high performance network connection. A dynamic resource broker can dynamically scale up/down a transactional system in the transactional middleware machine environment by adding/removing  
35 groups and machines according to the resource usage changes. The transactional middleware machine environment can comprise a deployment center in the transactional middleware

machine environment, wherein the deployment center maintains one or more deployment policies for the transactional middleware machine environment, and one or more deployment agents, each of which is associate with a transactional middleware machine of a plurality of transactional middleware machines in a transactional domain in the transactional middleware machine environment. The deployment center operates to receive machine usage information from the one or more deployment agents, and dynamically scale up or down resource used in the transactional domain based on the resource usage information collected by the one or more deployment agents.

**[0020]** In accordance with an embodiment of the invention, the system comprises a combination of high performance hardware, e.g. 64-bit processor technology, high performance large memory, and redundant InfiniBand and Ethernet networking, together with an application server or middleware environment, such as WebLogic Suite, to provide a complete Java EE application server complex which includes a massively parallel in-memory grid, that can be provisioned quickly, and can scale on demand. In accordance with an embodiment, the system can be deployed as a full, half, or quarter rack, or other configuration, that provides an application server grid, storage area network, and InfiniBand (IB) network. The middleware machine software can provide application server, middleware and other functionality such as, for example, WebLogic Server, JRockit or Hotspot JVM, Oracle Linux or Solaris, and Oracle VM. In accordance with an embodiment, the system can include a plurality of compute nodes, IB switch gateways, and storage nodes or units, communicating with one another via an IB network. When implemented as a rack configuration, unused portions of the rack can be left empty or occupied by fillers.

**[0021]** In accordance with an embodiment of the invention, referred to herein as “Sun Oracle Exalogic” or “Exalogic”, the system is an easy-to-deploy solution for hosting middleware or application server software, such as the Oracle Middleware SW suite, or WebLogic. As described herein, in accordance with an embodiment the system is a “grid in a box” that comprises one or more servers, storage units, an IB fabric for storage networking, and all the other components required to host a middleware application. Significant performance can be delivered for all types of middleware applications by leveraging a massively parallel grid architecture using, e.g. Real Application Clusters and Exalogic Open storage. The system delivers improved performance with linear I/O scalability, is simple to use and manage, and delivers mission-critical availability and reliability.

**[0022]** In accordance with an embodiment of the invention, a transactional system, e.g. Tuxedo, can be a set of software modules that enables the construction, execution, and administration of high performance, distributed business applications and has been used as transactional middleware by a number of multi-tier application development tools. The

transactional system is a platform that can be used to manage distributed transaction processing in distributed computing environments. It is a platform for unlocking enterprise legacy applications and extending them to a services oriented architecture, while delivering unlimited scalability and standards-based interoperability.

5   **[0023]**     A dynamic resource broker can automatically deploy application components of a transaction system, e.g. a Tuxedo, in a transactional middleware machine environment. Thus, the transactional system can take advantage of fast machines with multiple processors, e.g. Exallogic middleware machines, and a high performance network connection, e.g. an Infiniband (IB) network.

10

#### **Dynamic Resource broker**

**[0024]**     In accordance with an embodiment of the invention, a dynamic resource broker allows the users to automatically deploy/undeploy transactional middleware applications to different remote machines. This can be beneficial to the users of the transactional middleware applications. For example, as a command line style middleware product, Tuxedo applications  
15   can also be deployed to a single machine or multiple network-connected machines manually. However, using the manual mode, users need to login every machine to deploy their applications even when there are a large number of network-connected machines.

**[0025]**     **Figure 1** shows an illustration of a transactional middleware machine environment  
20   that supports automatically deploying/undeploying application components, in accordance with an embodiment of the invention. As shown in **Figure 1**, the transactional middleware machine environment 100 includes a plurality of transactional middleware machines, such as Machines A-D 101-103 and 110.

**[0026]**     A dynamic resource broker can include components such as: a data repository 105  
25   on Machine D 110, a deployment center 106 on Machine D 110, and one or more deployment agent, Agents 111-113, each of which resides on a transactional middleware machine, Machine A-C 101-103 in the transactional middleware machine environment 100.

**[0027]**     The data repository 105 resides in a memory. The data repository 105 can be used to store the related information, such as: application packages, distribution packages and  
30   configuration files. The deployment center 106 run by the one or more microprocessors. The deployment center 106 can receive all the user inputs 107 and is responsible for distributing the instructions/packages to the destination machines, Machines A-C 101-103, based on one or more deployment policies 116. Furthermore, the deployment center 106 is responsible for receiving the execution result from the destination machines, Machines A-C 101-103. Each  
35   deployment agent, Agent 111-113, is responsible for receiving the distribution packages, executing the deployment/un-deployment/management tasks, and providing the execution result

back to the deployment center 106.

[0028] In an example as shown in **Figure 1**, a user can first upload the application packages to the data repository 105. Then, the user can create a domain, Domain A 109, which includes two machines: Machine A 101 and Machine B 102. Then, the user can notify the deployment center 106 to perform the deployment tasks. The deployment center 106 can deploy the distribution packages to Machine A 101 and Machine B 102 and boot Domain A 109.

[0029] Additionally, Domain A 109 can include a third machine, Machine C 103, as the candidate machine that may be dynamically activated at runtime. For example, when the CPU usage of Machine B 102 increases to 100%, the deployment center can activate Machine C 103 to share the load with Machine B 102. The deployment center 106 can distribute the packages to Machine C, and instruct the deployment agent, Agent C 113, to perform management tasks, such as Tuxedo Management Information Base (MIB) operations. Once the management tasks have been successfully performed, Machine C 103 can be dynamically added to Domain A, and all the servers and services on Machine C 103 can be activated.

[0030] **Figure 2** illustrates an exemplary flow chart for supporting automatically deploying/undeploying application components in a transactional middleware machine environment, in accordance with an embodiment of the invention. As shown in **Figure 2**, at step 201, a data repository can receive one or more application packages, wherein each application package contains binary files for one or more transactional servers and configuration information that describes relationship and parameters of the one or more transactional servers in the application package. Then, at step 202, a deployment center can generate one or more distribution packages on the one or more microprocessors for each transactional middleware machine of a plurality of transactional middleware machines in the transactional middleware machine environment based on the one or more application packages. Finally, at 203, the deployment center can deploy the one or more distribution packages to the plurality of transactional middleware machines in the transactional middleware machine environment.

[0031] In accordance with an embodiment of the invention, and as disclosed below, the deployment of a transactional application, such as a Tuxedo application, includes several steps: application packages distribution, Tuxedo system environment setup, Tuxedo configuration files generation, and Tuxedo system booting, each of which will be discussed in the following sections.

### **Application Package**

[0032] In accordance with an embodiment of the invention, an application package is an archive package that can be prepared by the users or customers and deployed on one or more middleware machines. An application package contains the compiled binary files and other

environment files and can be organized based on the machine which is to be deployed with the transactional applications.

[0033] For example, a transactional application, such as a Tuxedo application, can be based on one or more application packages, each of which can be a zip file. The Tuxedo application (domain) can be defined in a TUXCONFIG, or UBBCONFIG configuration file, that specifies a set of machines, servers, and other resources. The Tuxedo application can exist on a single machine or across multiple network-connected machines. Users can first upload their application packages, in order to deploy the Tuxedo application.

[0034] **Figure 3** shows an illustration of the internal structure of an application package in a middleware machine environment, in accordance with an embodiment of the invention. As shown in **Figure 3**, an application package 300 can include multiple tiers, e.g. Tier 1 301, Tier 2 302, and Tier 3 303.

[0035] Tier 1 301 includes a name for the application package 300, which can be unique within all the uploaded application package names.

[0036] Tier 2 302 can be a child directory under Tier 1 301. Tier 2 302 contains a name that stands for the information about the platform the application package 300 can be deployed to. As show in **Figure 3**, Tier 2 can be used to represent a machine type, OEL55\_64\_ExaX22X8664.

[0037] Tier 3 303 includes child directories Tier 2 302, which contains the compiled binary files used by the transactional system, such as application servers, TMS servers, clients and so on. As show in **Figure 3**, under OEL55\_64\_ExaX22X8664 (Tier 2 302), the sub-directory "servers" can have two compiled server binary files: server1 and server2.

[0038] Furthermore, Tier 3 303 can contain different environmental files, such as Tuxedo machine level ENVFILE, Tuxedo group level ENVFILE, Tuxedo server level ENVFILE, and Tuxedo server level RCMD file. Additionally, under Tier 3 303, there can be sub-directories, where users can organize their applications.

[0039] Additionally, Tier 3 303 can contain a universal bulletin board configuration file, such as a Tuxedo UBB\_part file, which is a group level UBBCONFIG file. The Tuxedo UBB\_part file contains the GROUPS, RMS, SERVERS, SERVICES, ROUTINGS sections of a complete UBBCONFIG file and is mainly used to describe the relationship and parameters of all the servers within this package. The Tuxedo UBB\_part file can be used to generate the UBBCONFIG file when decide to deploy this package to a machine and its content can be modified at that time.

### Application Directory

[0040] In accordance with an embodiment of the invention, an application package can be used repeatedly in a transactional middleware machine environment. For example, an

application package can be applied to different domains, different machines in one domain or a single machine in one domain for multiple times.

**[0041]** Additionally, the dynamic resource broker can generate distribution packages that are deployed to the destination machines based on the application packages. After a distribution package is deployed to a destination machine, the transactional system can create an application directory on the destination machine.

**[0042]** **Figure 4** shows an illustration of the internal structure of an application directory in a middleware machine environment, in accordance with an embodiment of the invention. As shown in **Figure 4**, an application directory, such as a Tuxedo application directory APPDIR 401, can include multiple subdirectories, APP1 402 and APP2 403, for containing distribution packages for different applications. The structure of a deployed Tuxedo application 400 under the application directory, APPDIR 401, can resemble the structure of the application packages, with the exception of the Tuxedo UBB\_part file.

## Upload Application Package

**[0043]** In accordance with an embodiment of the invention, users can upload one or more prepared application packages to a data repository. Once the upload of the application packages is successful, the system allows the users to fill an application package information list, the content of which is shown in the following Table 1.

Item	Description
Application Package Name	The application package name, for example APP1.
Tuxedo Version	The Tuxedo Version this package is built on, for example, TUX11g. This item is just used for the users to indicate when they use this package later.
Supported Operation System	The Operation System this package can be deployed to. This information will be compared with the corresponding item in Machine list when deploy. E.g. "Oracle Enterprise Linux 5.5" and "Oracle Solaris 11 Express".
Tuxedo Word Size	To describe the word size this package is build on. For example, 32 bit or 64 bit. This information can be compared with the corresponding item in Machine list when deploy.
Machine Architecture	To describe which machine architecture this package can apply to. The candidate is: "Exalogic X2-2, X86-64" and "Exalogic X2-2,



	SPARC". This can be compared with the corresponding item in Machine list when deploy.
Services	The all services names supplied by this package. For example, TOUPPER1, TOUPPER2.
Dependent Services	The dependent service of this package. That means when users deploy this package, they must assure the packages which contain the dependent service are also deployed. This information is just for users to reference, the system will not check it automatically.
Data Base Type	If the application within this package relates to the Data Base, then please indicate the Data Base Type. This information is just for users to reference later, the system will not check it automatically.
Data Base Version	If the application within this package relates to the Data Base, then please indicate the Data Base Version. This information is just for users to reference later, the system will not check it automatically.

Table 1

### Create Machine List

- 5 **[0044]** In accordance with an embodiment of the invention, before a domain is created, users can create a list of machines that the application packages are deployed to. For each machine, users can fill a form that describes this particular machine. The content of the form is shown in the following Table 2.

10

Item	Description
Logical Name	It's the logical name of this machine and is named by the customer. The length of it must be 30 characters or less. This name must be unique among all the machines and can be the same with the machine's physical name.
Physical Name	It's the physical name of a processor, for example, the value produced by the UNIX system <code>uname -n</code> command. The length of it must be 30 characters or less.
Operation System	The operation system installed on this machine.
Machine Architecture	To describe this machine's architecture. The candidate is: "Exalogic X2-2, X86-64" and "Exalogic X2-2, SPARC". This will be compared with the "Machine Architecture" item in application package list when deploy.
C/C++ and COBOL compilers	This specifies the cc version and so on. And this information is just for users to reference.
Tuxedo installation	Specifies the Tuxedo's installation OS version. This information

	will be compared with the corresponding item "Supported Operation System" in application package list when deploy an application package to this machine. Now this feature only supports: "Oracle Enterprise Linux 5.5" and "Oracle Solaris 11 Express".
Tuxedo Word Size	To describe the word size of the Tuxedo installed on this machine. For example, 32 bit or 64 bit. This information will be compared with the corresponding item "Tuxedo Word Size" in application package list when deploy an application package to this machine.
Tuxedo Version	The Tuxedo Version, for example, TUX11g. This item is just used for the users to reference when they use this machine.
TUXDIR	The TUXDIR for the Tuxedo system installed on this machine. We only acknowledge one TUXDIR directory in one machine.
Data Base Type	All the Data Base types installed on this machine. This item is just used for the users to reference when they use this machine.
Data Base Version	All the Data Base Versions installed on this machine. This item is just used for the users to reference when they use this machine.

Table 2

**[0045]** In the above machine list, all the machines can be named by its logical names. The logical name of a machine can be unique in the machine list, and the logical name of a machine can be the same as a physical name of the machine. Thus, one physical machine can have more than one logical name and have more than one instance in the machine list. Therefore, the users can specify multiple logical names for one physical machine, and deploy the different application packages to the same machine, e.g. in the case of the faked MP model in a testing environment.

#### **UBBCONFIG Automatic Generation Rules**

**[0046]** In accordance with an embodiment of the invention, users can choose to automatically generate configuration files for the transactional servers, e.g. the UBBCONFIG file for each Tuxedo server. The UBBCONFIG file can include different sections, such as a RESOURCE section, a MACHINES section, a GROUPS section, a RMS section, a NETGROUPS section, a NETWORK section, a SERVERS section, a SERVICES section, a ROUTING Section.

**[0047]** Attached is Appendix A that provides further information regarding supporting a dynamic resource broker in a transactional middleware machine system, and various other aspects of the platform described throughout this disclosure. The information in Appendix A is provided for illustrational purposes and should not be construed to limit all of the embodiments of the invention.

### An Example of Automatic Deployment in Tuxedo

[0048] In the following sections, an example is used for illustrating the procedure of automatically deploying or undeploying Tuxedo applications in a transactional middleware environment. In this example, two Tuxedo applications, APP1 and APP2, can be automatically  
 5 deployed on two middleware machines, lclnx 24 and lclnx16. In other examples, different configurations can be used in a transactional middleware environment without limitation.

### Create a Tuxedo Domain

[0049] In accordance with an embodiment of the invention, users can prepare the application  
 10 packages before creating a domain in a transactional middleware for deploying transactional applications. In this example, every machine in the domain uses one application package. In other examples, a machine can have one or more application packages as needed.

[0050] **Figure 5** shows an illustration of preparing an application package 500, in accordance with an embodiment of the invention. As shown in **Figure 5**, an application package, APP1.zip 501, can include a UBB\_part section 502 and a servers section 503. The following  
 15 Listing 1 shows the UBB\_part section sections in APP1.zip.

```

20      *GROUPS:
        G1      LMID=simple1  GRPNO=30
      *SERVERS
        APP1/OEL55_64_ExaX22X8664/servers/simpserv1
                                SRVGRP=G1      SRVID=20
                                CLOPT="-A"
25      *SERVICES
                                TOUPPER1

```

Listing 1

[0051] The above **Figure 5** and Listing 1 illustrate the application package, APP1.zip, for  
 30 deploying a first application, APP1.

[0052] Additionally, in this example, another application package, APP2.zip, can be prepared  
 in a similar fashion for deploying a second application, APP2.

[0053] **Figure 6** shows an illustration of creating a machine list for a domain in a GUI  
 Console, in accordance with an embodiment of the invention. As shown in Figure 6, the GUI  
 Console use a machine form 600 to define different properties associated with a particular  
 machine with a logical name, lclnx24 601. Also as shown in Figure 6, the entries in the machine  
 form 600 can be based on Table 2 as disclosed above.

[0054] Additionally, in this example, another similar machine form (not shown) can be used for setting up a second machine in the system.

[0055] **Figure 7** shows an illustration of uploading an application package, in accordance with an embodiment of the invention. As shown in **Figure 7**, an application package form 700 can be displayed for requiring a user to file, when users successfully upload an application package 701. Also as shown in **Figure 7**, the various entries in the machine form 700 can be based on Table 2 as disclosed above.

[0056] Similarly, there can be another application package form (not shown) for uploading the application package, APP2.zip.

[0057] **Figure 8** shows an illustration of creating a domain in a transactional middleware environment, in accordance with an embodiment of the invention. As shown in **Figure 8**, the users can create a domain with a unique domain name, e.g. DOM1 801. This domain can be configured to include machines from a machine list 802 and to run application packages from a package list 803.

[0058] When a domain is created in a transactional middleware environment, different templates can be used to automatically generate the configuration files for deployment. For example, the contents of the automatically generated Tuxedo configuration files can be detailed in Appendix A.

[0059] Additionally, in Tuxedo, there can be two default UBBCONFIG templates: one UBBCONFIG template for RESOURCES part named UBB\_Resource, and another UBBCONFIG template for MACHINES part named UBB\_Machine. Also, users can define additional templates and save them to the data repository.

[0060] As shown in **Figure 8**, at the time when DOM1 801 is created, users can choose the default UBB\_Resource template and can indicate that the domain is in MP mode. Furthermore, users can modify and add the values to the RESOURCES section in the configuration file.

### Create Virtual Machines in a Tuxedo Domain

[0061] In accordance with an embodiment of the invention, users can create one or more virtual machines in a domain. A virtual machine is a symbol that does not stand for a real physical machine. A logical machine from the machine list can be bound to the virtual machine. The naming rule for a virtual machine can be the same as that of a logical machine, and a name for a virtual machine can be unique among all the virtual machines within one domain.

[0062] **Figure 9** shows an illustration of binding a logical machine to a virtual machine in a domain, in accordance with an embodiment of the invention. As shown in **Figure 9**, a virtual machine M1 902 is created in DOM1 901. Additionally, users can bind a logical machine, lclnx24 903, from the machine list to the virtual machine M1 902, and can indicate that this logical

machine, lclnx24 903, is the master for the domain DOM1 901. Then, the default UBB\_Resource section 904 can be added to the UBBCONFIG file and the content can be modified according to the information related to lclnx24 903 in the machine list. Furthermore, users can also modify or add the values to corresponding items of MACHINES section 905 and NETWORK section 906.

**[0063]** **Figure 10** shows an illustration of adding an application package to a virtual machine in a domain, in accordance with an embodiment of the invention. As shown in **Figure 10**, users can add an application package, APP1 1003, from the package list to a virtual machine M1 1002 in a domain, DOM1 1001. Then, the UBB\_part from the application package APP1, e.g. the default UBB\_Resource section 1004, can be added to the UBBCONFIG file and the content can be modified according to the information of the logical machine lclnx24, which is bound to the virtual machine M1 1002. Additionally, users can modify or add the values to items in MACHINES, GROUPS, SERVERS and SERVICES sections 1005-1009.

**[0064]** **Figure 11** shows an illustration of setting up multiple virtual machines in a domain, in accordance with an embodiment of the invention. As shown in **Figure 11**, users can create multiple virtual machines, such as a virtual machine, M1 1102 and a virtual machine, M2 1103, in a domain, DOM1 1101. Furthermore, the users can bind a logical machine to a virtual machine. For example, a logical machine, lclnx16, can be bound to the virtual machine M1 1002 and responsible for running application package AZPP1, and a logical machine, lclnx24, can be bound to the virtual machine M2, and responsible for running another application package, APP2.

**[0065]** The system can compare the information in the application package form with the machine form, every time when an archive package is added to a virtual machine bound to a logical machine. If there is an error, the system can report the error to the users.

### **Configure a Tuxedo Domain**

**[0066]** In accordance with an embodiment of the invention, the system can provide configuration templates for each machine in a domain.

**[0067]** In Tuxedo, users can select and modify the default DMCONFIG templates from the data repository. Additionally, the data repository can provide a setenv.ksh template for each machine. Users can use the setenv.ksh template to set various environment variables for Tuxedo applications, e.g. TUXDIR, PATH, and APPDIR.

**[0068]** Furthermore, the system provides a deployment descriptor and an undeployment descriptor for each machine according to the above information. Users can modify the two descriptors according to their own requirements.

**[0069]** The deployment descriptor is a script describing the steps which can be taken after

the distribution package is distributed to the destination machines. The following Listing 2 shows an exemplary deployment descriptor.

```

5      #!/usr/bin/sh
      . setenv.ksh
      tlistpwd $TUXDIR
      tmloadcf -y UBBCONFIG
      ...

```

Listing 2

**[0070]** The undeployment descriptor is a script describing the steps which can be taken when users want to undeploy the domain. The following Listing 3 shows an exemplary undeployment descriptor.

```

15      . setenv.ksh
      kill -9 tlisten
      rm -rf $APPPDIR
      ...

```

Listing 3

**[0071]** In accordance with an embodiment of the invention, the distribution packages can be generated to contain the modified setenv.ksh (setenv.cmd) template, UBBCONFIG, DMCONFIG, deployment/undeployment descriptor and the compiled binary executable files.

## 25 Data Repository

**[0072]** In accordance with an embodiment of the invention, both the application packages and the distribution packages can be stored in the data repository.

**[0073]** **Figure 12** shows an illustration of the internal structure of the data repository, in accordance with an embodiment of the invention. As shown in **Figure 12**, the data repository, Repository 1201, includes an application package directory, AppPackage 1202, and a deployment directory, DeployDir 1203. The original uploaded application packages are stored under the AppPackage directory 1202, while the deployment units, such as domains, are stored under the DeployDir directory 1203.

**[0074]** As shown in **Figure 12**, the domain name is 'DOM1', and the machine names are 'lclnx24' and 'lclnx26', each of which contains a DeployDescriptor, an UndeployDescriptor, and various distribution packages.

**[0075]** Additionally, one application package can be added to a single machine in a domain for multiple times, in which case the system can automatically create an index for it, e.g. APP1\_1, APP1\_2, etc.

**Deploy/Undeploy a Tuxedo Domain**

**[0076]** In accordance with an embodiment of the invention, the deployment center can deploy the distribution archive files to every destination machine according to a distribution file.

5 Subsequently, the deployment agent can configure and set the Tuxedo system. If successful, the system can record the status of the Tuxedo system into a status file. If the deployment of any machine in the domain fails, the system can roll back the deployment operations that have been done.

**[0077]** **Figure 13** shows an illustration of the structure of an application directory, in accordance with an embodiment of the invention. As shown in **Figure 13**, an application directory, APPDIR 1301, is create to include multiple subdirectories, APP1 1302 and APP2 1303, for containing deployment file for different applications after a successful deployment.

**[0078]** Additionally, when the domain is in a shutdown status, then users can undeploy the Tuxedo application. Furthermore, a deployment agent on a deployed machine can use the undeployment descriptor to perform the undeployment tasks.

**[0079]** **Figure 14** shows more details of the transactional middleware machine environment that supports automatically deploying/undeploying application components in **Figure 1**, in accordance with an embodiment of the invention. Same components in **Figure 14** as those in **Figure 1** are denoted with the same reference numbers and the detail description thereof is omitted.

**[0080]** As shown in **Figure 14**, the deployment center 106 comprises: a application package receiving unit 1061 receiving one or more application packages, wherein each application package contains binary files for one or more transactional servers and configuration information that describes relationship and parameters of the one or more transactional servers in the application package; a distribution package generating unit 1062 generating one or more distribution packages for each transactional middleware machine of a plurality of transactional middleware machines in the transactional middleware machine environment based on the one or more application packages; and a distribution package deploying unit 1063 deploying the one or more distribution packages to the plurality of transactional middleware machines in the transactional middleware machine environment.

**[0081]** Preferably, each application package is allowed to contain information about a platform that the application package can be deployed to.

**[0082]** Preferably, an application package is applied to different transactional domains, wherein each transactional domain includes one or more transactional middleware machines.

35 **[0083]** Preferably, the one or more application directories are created on each transactional middleware machine that is deployed with one or more transactional servers, based on the

application packages.

**[0084]** Preferably, the deployment center 106 further comprises a deploying unit 1064 deploying multiple application packages into a same application directory.

**[0085]** Preferably, the deployment center 106 further comprises a configuration metadata generating unit 1065 automatically generating configuration metadata for each transactional middleware machine that is deployed with the one or more transactional servers based on the configuration information in the application package.

**[0086]** Preferably, the data repository 105 stores the one or more application packages uploaded by a user.

**[0087]** Preferably, the deployment center 106 further comprises a transactional application deploying unit 1066 deploying a transactional application that includes multiple application packages to either a single transactional middleware machine or multiple cross-connected transactional middleware machines.

**[0088]** Preferably, each said application package is a compressed file that contains: a first tier that contains names of the application package, a second tier that contains the configuration information that describes relationship and parameters of the one or more transactional servers in the application package, and a third tier that contains compiled binary files for the transactional servers.

**[0089]** Preferably, the deployment center 106 further comprises a undeploying unit 1067 undeploying at least one distribution package from the plurality of transactional middleware machines in the transactional middleware machine environment.

**[0090]** These units and components can be implemented by software and/or hardware.

### **Boot/Shutdown the Tuxedo application**

**[0091]** In accordance with an embodiment of the invention, after the Tuxedo application is deployed to the destination machine, the deployment center can connect with the deploy agents, so that the users can choose to boot the whole system. When the system is successfully booted, the system can record the status into a status file and update the status of the domain. The users are allowed to perform the dynamic deployment, once the domain has been successfully booted. On the other hand, the booted Tuxedo system may be shutdown if any error is found.

**[0092]** The present invention may be conveniently implemented using one or more conventional general purpose or specialized digital computer, computing device, machine, or microprocessor, including one or more processors, memory and/or computer readable storage media programmed according to the teachings of the present disclosure. Appropriate software coding can readily be prepared by skilled programmers based on the teachings of the present disclosure, as will be apparent to those skilled in the software art.



**[0093]** In some embodiments, the present invention includes a computer program product which is a storage medium or computer readable medium (media) having instructions stored thereon/in which can be used to program a computer to perform any of the processes of the present invention. The storage medium can include, but is not limited to, any type of disk including floppy disks, optical discs, DVD, CD-ROMs, microdrive, and magneto-optical disks, ROMs, RAMs, EPROMs, EEPROMs, DRAMs, VRAMs, flash memory devices, magnetic or optical cards, nanosystems (including molecular memory ICs), or any type of media or device suitable for storing instructions and/or data.

**[0094]** The foregoing description of the present invention has been provided for the purposes of illustration and description. It is not intended to be exhaustive or to limit the invention to the precise forms disclosed. Many modifications and variations will be apparent to the practitioner skilled in the art. The embodiments were chosen and described in order to best explain the principles of the invention and its practical application, thereby enabling others skilled in the art to understand the invention for various embodiments and with various modifications that are suited to the particular use contemplated. It is intended that the scope of the invention be defined by the following claims and their equivalence.

## Appendix A

**[0095]** In accordance with an embodiment of the invention, the system can provide an UBB\_Resource template to the Tuxedo users for defining the RESOURCE section. When creating a domain, a user can choose this template to generate the RESOURCES section for UBBCONFIG. An exemplary item list in the template is shown in the following Table 3.

Item	Value
IPCKEY	33333
MASTER	NULL When the users specify which machine is master, this item will be generated automatically.
MODE	User can specify SHM or MP.

Table 3

**[0096]** The items in the above Table 3 can be modified by the users except the MASTER item. The MASTER item can be filled when the users specify which machine is a master in a domain and which machine is the backup. Users can add other parameters besides these parameters. Users can also create their own UBB\_Resource templates and save them to the system. In all templates, the MASTER can be filled by the system automatically following above rules.

**[0097]** In accordance with an embodiment of the invention, the system can provide an UBB\_Machine template to the Tuxedo users. When creating a domain and specifying machine, a user can choose this template to generate the MACHINES section for the UBBCONFIG file. An exemplary item list in the template is shown in the following Table 4.

5

Item	Value
ADDRESS	This is generated by the system according to the machine's logical name. Can't be modified by the users.
LMID	This is generated automatically by the system. The naming rule is SITE1, SITE2..., can't be modified by the users.
APPPDIR	Specified by the users when they add machines to one domain.
TUXCONFIG	Be default, it's the \$APPPDIR/tuxconfig. Because the tuxconfig can be placed in raw disk, so users can fill it with other value, and this feature will deploy tuxconfig to that place according to it.
TUXDIR	This is generated automatically by the system using the TUXDIR specified by the machine list. Users can not modify it.

Table 4

10 **[0098]** Users can add other parameters of MACHINES section to the UBBCONFIG file. Users can also specify the TLOG to raw disk, and the system can delete it when undeploying this domain. User can also create additional UBB\_Machine templates and save them to the system. The parameters replacement rule can follow the above.

15 **[0099]** In accordance with an embodiment of the invention, when the users add an application package to one domain, the system can replace some parameters of the GROUPS section in the UBB\_part of that package. An exemplary item list in the template is shown in the following Table 5.

Item	Value
GROUPNAME	This is generated automatically by the system. The naming rule is GROUP1, GROUP2..., can't be modified by the users.
LMID	This is generated by the system according to the machine which the package is deployed to.
GRPNO	This is generated automatically by the system. It can't be modified by the customers.

20

Table 5

**[00100]** The system can keep the values of other parameters of GROUPS section in a UBB\_part file, and allows the users to modify them freely. Similarly, users can add other

parameters to the UBBCONFIG file.

**[00101]** In accordance with an embodiment of the invention, when adding an application package to one domain, the system can replace some parameters of the RMS section in the UBB\_part of that package. An exemplary item list in the template is shown in the following Table

5 6.

Item	Value
RMSNAME	This is generated automatically by the system. The naming rule is RMS1, RMS2..., can't be modified by the users.
SRVGRP	This will be replaced with the group's new name which is generated by the system before. This can't be modified by the customers.
RMID	This is generated automatically by the system. The naming rule is 1, 2 and so on. It can't be modified by the customers.

Table 6

10

**[00102]** The system can keep the values of other parameters i RMS section in a UBB\_part file, and allows the users to modify them freely. Similarly, the users can add other parameters to the UBBCONFIG.

15

**[00103]** In accordance with an embodiment of the invention, all the parameters in the NETGROUPS section can be filled by the users themselves if they need.

**[00104]** In accordance with an embodiment of the invention, the NETWORK section is also in the UBB\_Machine template. If the domain is in a MP mode, then system can automatically add this section to the UBBCONFIG. An exemplary item list in the template is shown in the following Table 7.

20

Item	Value
LMID	This is generated automatically by the system. The users can't modify it.
NADDR	The system will generate //hostname:port_number, and users can modify the port_number.
NLSADDR	The system will generate //hostname:port_number, and users can modify the port_number.

Table 7

25

**[00105]** Additionally, users can add other parameters to the NETWORK section.

**[00106]** In accordance with an embodiment of the invention, when adding an application package to one domain, the system can replace some parameters of the SERVERS section in the UBB\_part of that package. An exemplary item list in the template is shown in the following

Table 8.

Item	Value
SVRGRP	This will be replaced with the group's new name which is generated by the system before. This can't be modified by the customers.
SRVID	This is generated automatically by the system. The naming rule is 1, 2..., can't be modified by the users.

Table 8

**[00107]** The system can keep the values of other parameters of SERVERS section in UBB\_part file, and allows the users to modify them freely. Similarly, the users can add other parameters to the UBBCONFIG.

**[00108]** In accordance with an embodiment of the invention, when adding an application package to one domain, system can replace some parameters of the SERVICES section in the UBB\_part of that package. An exemplary item list in the template is shown in the following Table 9.

Item	Value
SRVGRP	If the UBB_part has this parameter, then system will replace it with the corresponding group generated before. Although this parameter is not a required parameter, we recommend users to add it. When there're same services in different application packages, they can have different operations.
ROUTING	If the UBB_part has this parameter, then system will replace it with the corresponding routing name generate by the system.

Table 9

**[00109]** The system can keep the values of other parameters of SERVICES section in UBB\_part file, and allows the users to modify them freely. Similarly, the users can add other parameters to the UBBCONFIG.

**[00110]** In accordance with an embodiment of the invention, when adding an application package to a domain, the system can replace some parameters of the ROUTING section in the UBB\_part of that package. An exemplary item list in the template is shown in the following Table 10.

Item	Value
ROUTING_CRITERIA_NAME	This is generated automatically by the system. The naming rule is ROUTING1, ROUTING2..., can't be modified by the users.
RANGES	This parameter's group will be replaced by the system.

Table 10

- 5 **[00111]** The system can keep the values of other parameters of ROUTING section in UBB\_part file, and allows the users to modify them freely. Similarly, the users can add other parameters to the UBBCONFIG.

**Claims:**

What is claimed is:

- 5 1. A method for supporting automatically deploying application components in a transactional middleware machine environment, comprising:  
receiving one or more application packages, wherein each application package contains binary files for one or more transactional servers and configuration information that describes relationship and parameters of the one or more transactional servers in the application package;  
10 generating one or more distribution packages on the one or more microprocessors for each transactional middleware machine of a plurality of transactional middleware machines in the transactional middleware machine environment based on the one or more application packages; and  
deploying the one or more distribution packages to the plurality of transactional  
15 middleware machines in the transactional middleware machine environment.
2. The method of claim 1, further comprising allowing each application package to contain information about a platform that the application package can be deployed to.
- 20 3. The method of claim 1, further comprising applying an application package to different transactional domains, wherein each transactional domain includes one or more transactional middleware machines.
4. The method of claim 1, further comprising creating one or more application directory on  
25 each transactional middleware machine that is deployed with one or more transactional servers, based on the application packages.
5. The method of claim 1, further comprising deploying multiple application packages into a same application directory.  
30
6. The method of claim 1, further comprising automatically generating configuration metadata for each transactional middleware machine that is deployed with the one or more transactional servers based on the configuration information in the application package.
- 35 7. The method of claim 1, further comprising providing a data repository that stores the one or more application packages uploaded by a user.

8. The method of claim 1, further comprising deploying a transactional application that includes multiple application packages to either a single transactional middleware machine or multiple cross-connected transactional middleware machines.

5

9. The method of claim 1, wherein each said application package is a compressed file that contains:

a first tier that contains names of the application package,

a second tier that contains the configuration information that describes relationship and

10 parameters of the one or more transactional servers in the application package, and

a third tier that contains compiled binary files for the transactional servers.

10. The method of claim 1, further comprising undeploying at least one distribution package from the plurality of transactional middleware machines in the transactional middleware machine environment.

15

11. A system for supporting automatically deploying application components in a transactional middleware machine environment, comprising:

one or more microprocessors;

20 a deployment center running on the one or more microprocessors in the transactional middleware machine environment, wherein the deployment center is capable of:

receiving one or more application packages, wherein each application package contains binary files for one or more transactional servers and configuration information that describes relationship and parameters of the one or more transactional servers in the application package;

25

generating one or more distribution packages for each transactional middleware machine of a plurality of transactional middleware machines in the transactional middleware machine environment based on the one or more application packages; and

deploying the one or more distribution packages to the plurality of transactional middleware machines in the transactional middleware machine environment.

30

12. The system of claim 11, wherein each application package is allowed to contain information about a platform that the application package can be deployed to.

35 13. The system of claim 11, wherein an application package is applied to different transactional domains, wherein each transactional domain includes one or more transactional

middleware machines.

14. The system of claim 11, wherein the one or more application directories are created on each transactional middleware machine that is deployed with one or more transactional servers,  
5 based on the application packages.

15. The system of claim 11, wherein the deployment center is capable of deploying multiple application packages into a same application directory.

10 16. The system of claim 11, wherein the deployment center is capable of automatically generating configuration metadata for each transactional middleware machine that is deployed with the one or more transactional servers based on the configuration information in the application package.

15 17. The system of claim 11, further comprising a data repository that stores the one or more application packages uploaded by a user.

18. The system of claim 11, wherein the deployment center is capable of deploying a transactional application that includes multiple application packages to either a single  
20 transactional middleware machine or multiple cross-connected transactional middleware machines.

19. The system of claim 11, wherein each said application package is a compressed file that contains:

25 a first tier that contains names of the application package,  
a second tier that contains the configuration information that describes relationship and parameters of the one or more transactional servers in the application package, and  
a third tier that contains compiled binary files for the transactional servers.

30 20. The system of claim 11, wherein the deployment center is capable of undeploying at least one distribution package from the plurality of transactional middleware machines in the transactional middleware machine environment.

21. A program that causes a computer to perform the method of any of claim 1 to 10.

35

22. A computer readable non-volatile media that stores the program of claim 21.



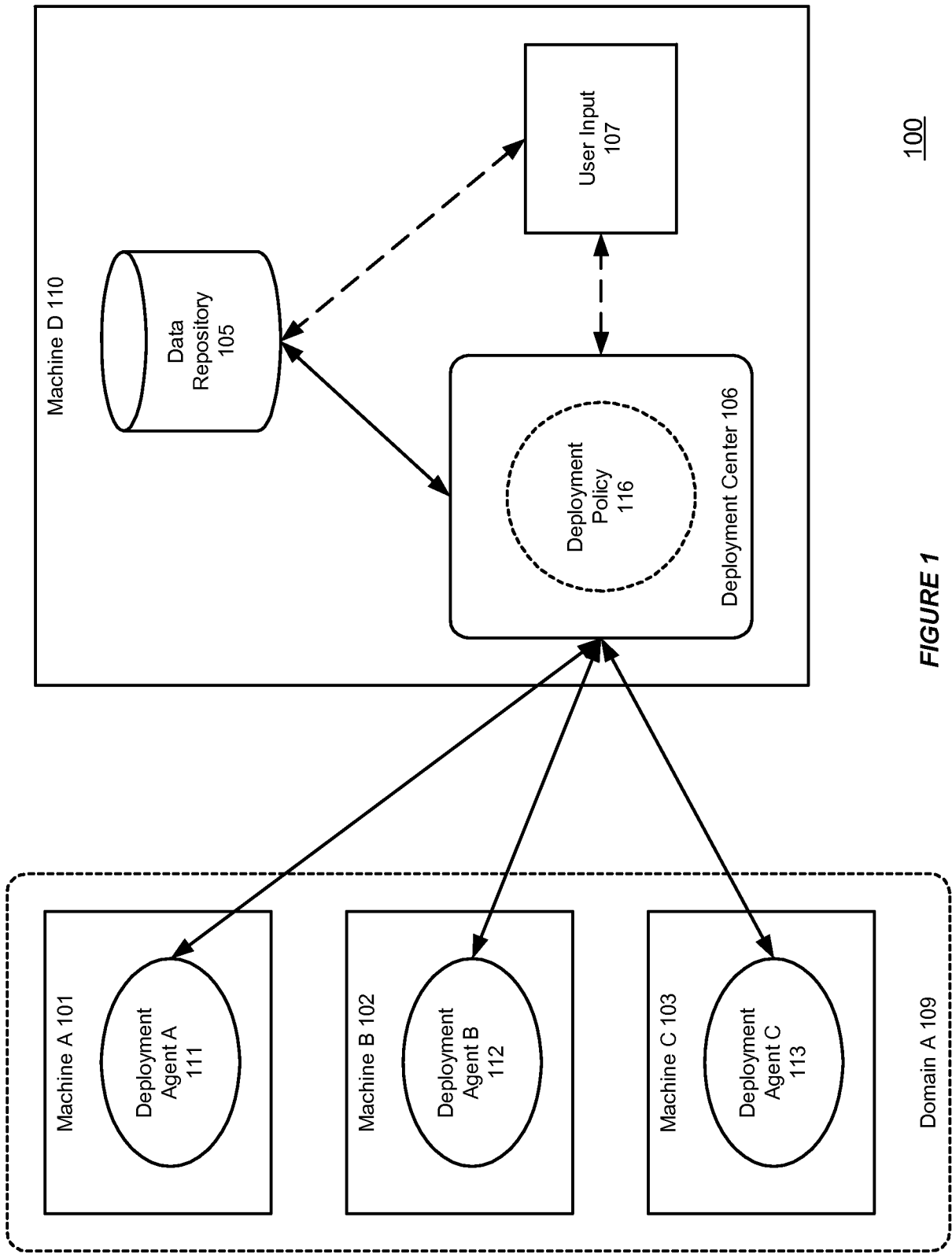
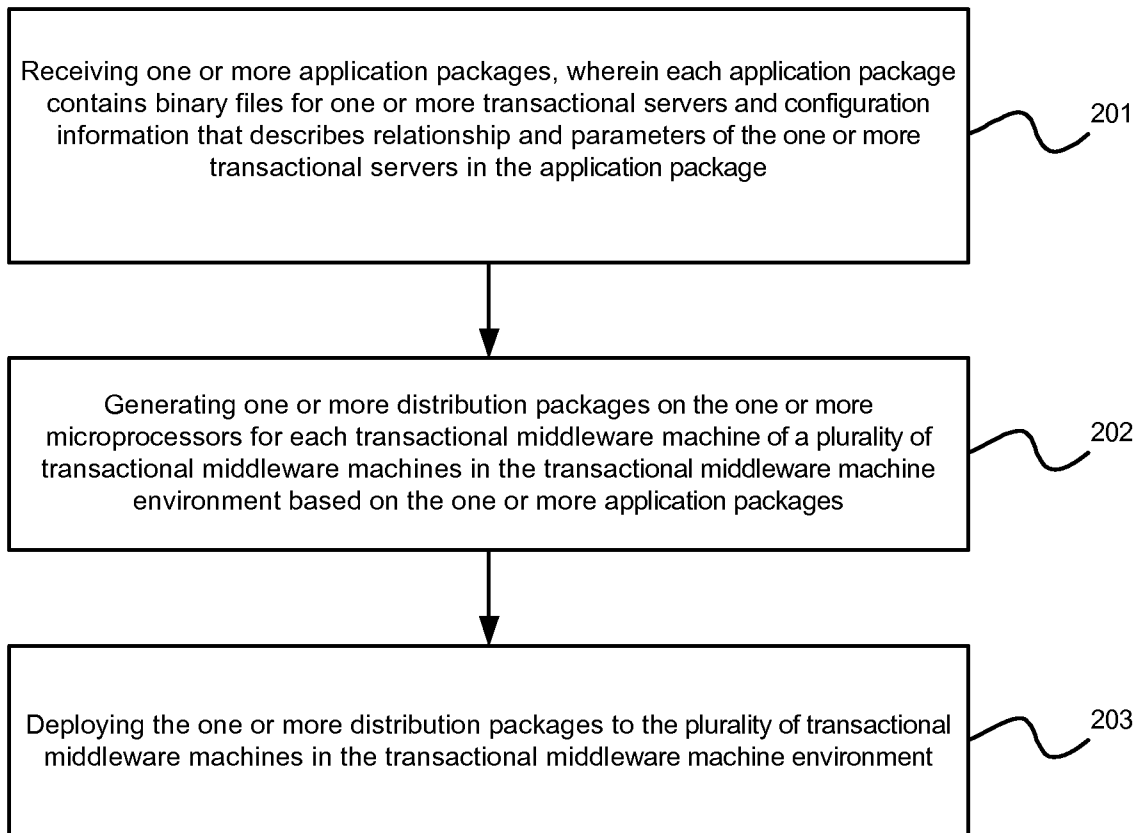


FIGURE 1

**FIGURE 2**

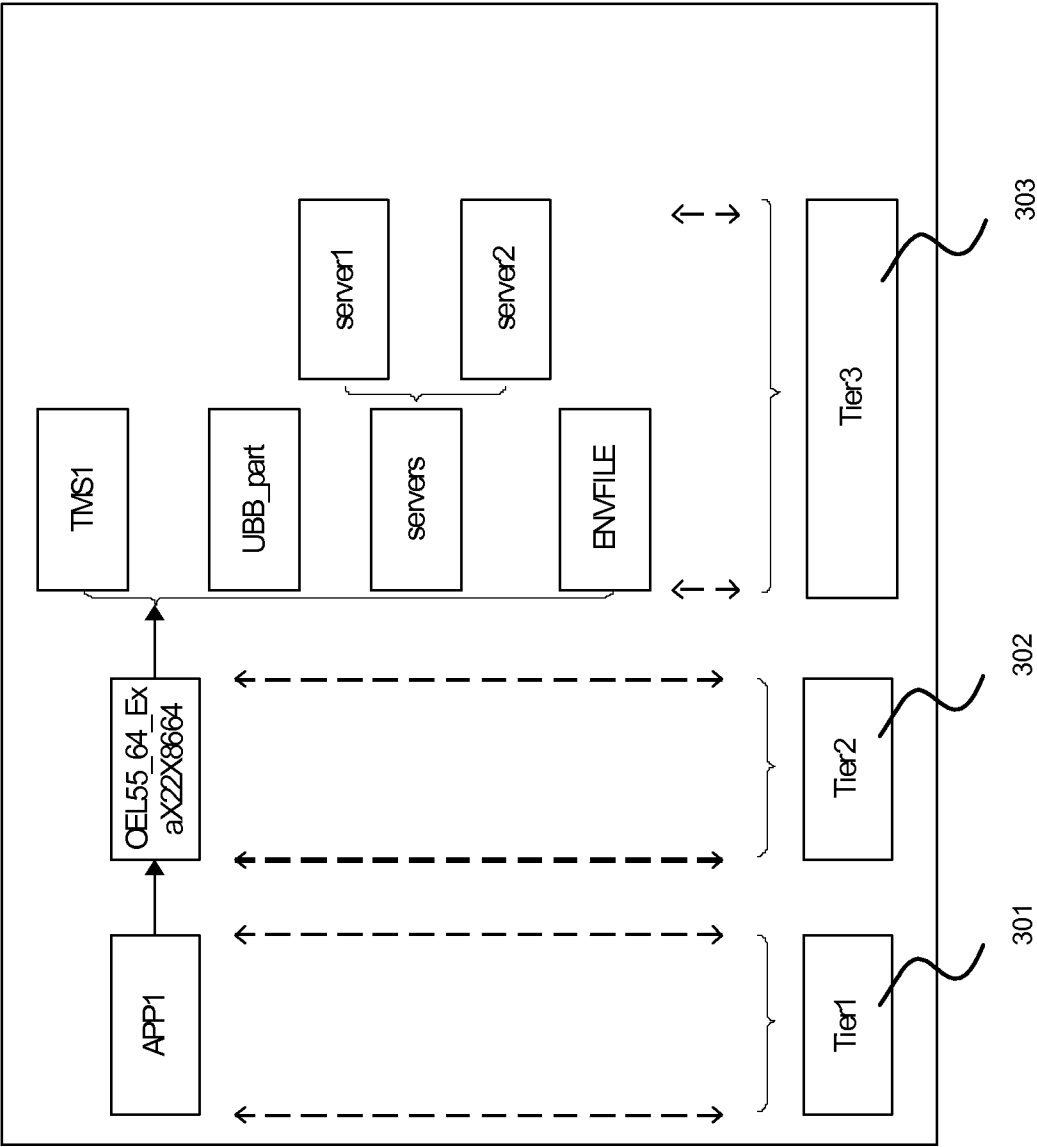


FIGURE 3

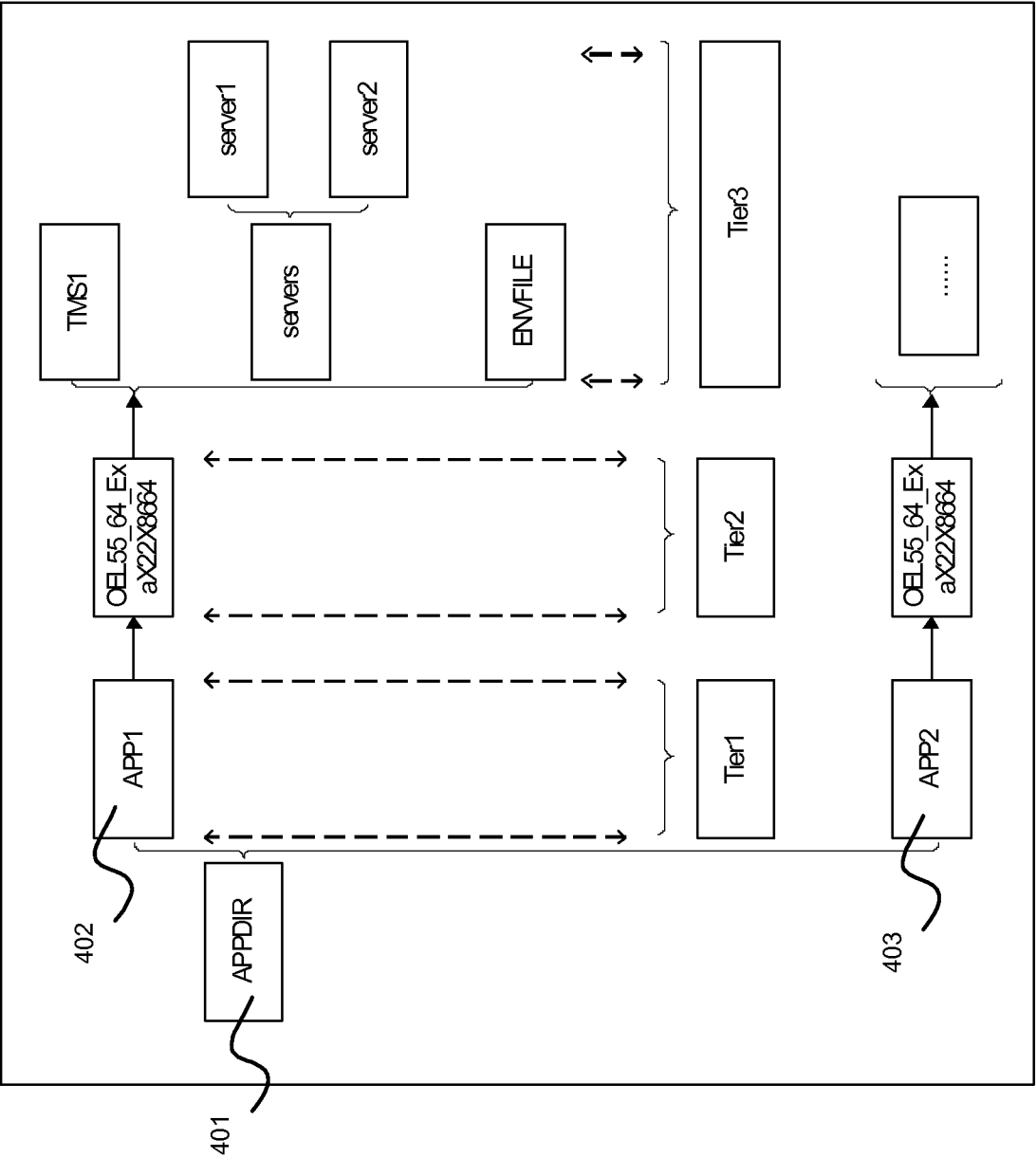


FIGURE 4

500

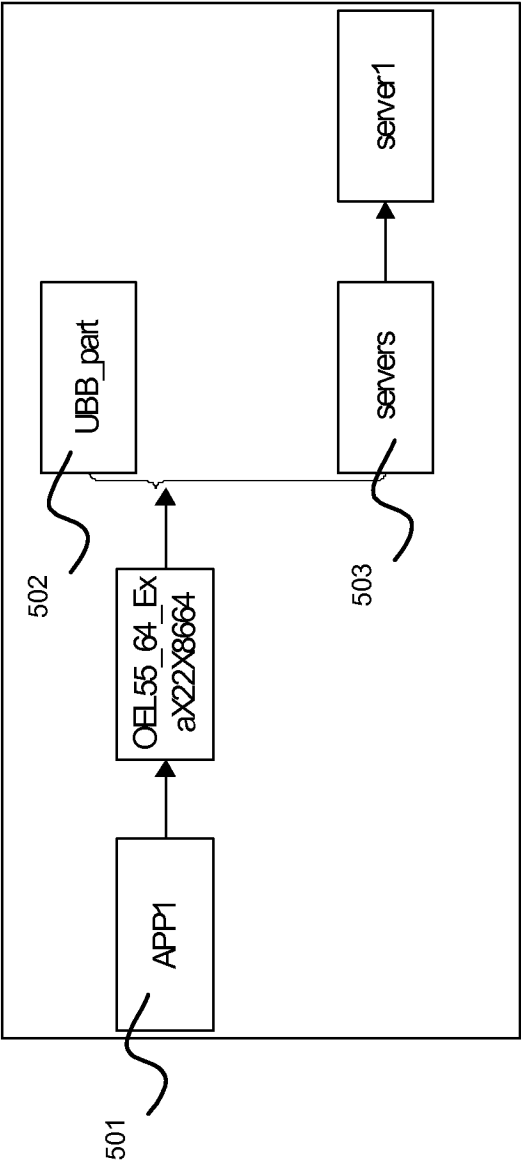



FIGURE 5

6/14

600

Logical Name	ldnx24  601
Physical Name	ldnx24
Operation System	Oracle Enterprise Linux 5.5
Machine Architecture	Exalogic X2-2, X86-64
C/C++ and COBOL compilers	C/C++: gcc/g++ 4.1.2 200807048
Tuxedo installation	Oracle Enterprise Linux 5.5
Tuxedo Word Size	64bit
Tuxedo Version	TUX11gR1
TUXDIR	/nfs/lcfiletc/vol1/TUX11gR1
Data Base Type	
Data Base Version	

**FIGURE 6**

700


Application Package Name	APP1.jar  701
Tuxedo Version	TUX11gR1
Supported Operation System	Oracle Enterprise Linux 5.5
Tuxedo Word Size	64bit
Machine Architecture	Exalogic X2-2, X86-64
Services	TOUPPER1
Dependent Services	
Data Base Type	
Data Base Version	

FIGURE 7

8/14

DOMAIN LIST	*RESOURCES IPCKEY MASTER MODEL	MACHINE LIST	PACKAGE LIST
DOM1		LCLNX24	APP1
		LCLNX16	APP

**FIGURE 8**



9/14

DOMAIN LIST	*RESOURCES	MACHINE LIST	PACKAGE LIST
<div data-bbox="212 421 427 477">DOM1 <span style="position: absolute; left: 225px; top: 188px;">901</span></div> <div data-bbox="228 622 467 667">M1 <span style="position: absolute; left: 255px; top: 278px;">902</span></div> <div data-bbox="355 701 459 734">MACHINE</div> <div data-bbox="443 779 547 813">LCLNX24</div> <div data-bbox="403 869 467 902">903</div> <div data-bbox="355 925 459 958">PACKAGE</div>	<div data-bbox="571 342 866 409">IPCKEY 33333</div> <div data-bbox="571 432 946 768">           DOMAINID SIMPAPP            MASTER SITE1            MAXACCESSERS 50            MAXSERVERS 40            MAXSERVICES 40            MODEL MP            LDBAL Y            OPTIONS MIGRATE,LAN            SYSTEM_ACCESS PROTECTED         </div> <div data-bbox="571 790 1225 1037"> <div data-bbox="571 790 691 857">*MACHINES</div> <div data-bbox="571 857 675 880">"LCLNX24"</div> <div data-bbox="675 857 1225 1037"> <div data-bbox="675 857 786 880">LMID=SITE</div> <div data-bbox="675 891 1058 913">APPDIR="/NFS/LCFILERC/VOL1/APP_1"</div> <div data-bbox="675 925 1225 947">TUXCONFIG="/NFS/LCFILERC/VOL1/APP_1/TUXCONFIG"</div> <div data-bbox="675 958 1090 981">TUXDIR="/NFS/LCFILERC/VOL1/TUX11GR1"</div> <div data-bbox="675 992 770 1014">TYPE="A"</div> </div> </div> <div data-bbox="571 1059 683 1081">*NETWORK</div> <div data-bbox="571 1104 643 1126">"SITE1"</div> <div data-bbox="675 1137 938 1160">NADDR="//LCLNX24:32333" <span style="position: absolute; left: 605px; top: 488px;">906</span></div> <div data-bbox="675 1171 962 1193">NLSADDR="//LCLNX24:32334"</div>	<div data-bbox="1246 813 1342 835">LCLNX24</div> <div data-bbox="1246 1037 1342 1059">LCLNX16</div>	<div data-bbox="1374 813 1445 835">APP_1</div> <div data-bbox="1374 1037 1445 1059">APP_2</div>

FIGURE 9

10/14

DOMAIN LIST	*RESOURCES	MACHINE LIST	PACKAGE LIST
<div data-bbox="204 409 555 470">DOM1 <span style="float: right;">1001</span></div> <div data-bbox="204 481 555 660"> <div data-bbox="204 481 555 548">└─ M1 <span style="float: right;">1002</span></div> <div data-bbox="204 560 555 817"> <div data-bbox="204 560 555 705">└─ MACHINE</div> <div data-bbox="204 716 555 817">└─ LCLNX24</div> </div> <div data-bbox="204 828 555 985"> <div data-bbox="204 828 555 896">└─ PACKAGE</div> <div data-bbox="204 907 555 985">└─ APP1</div> </div> <div data-bbox="204 996 555 1097"><span style="float: right;">1003</span></div> </div>	<div data-bbox="563 409 1230 470">IPCKEY 33333</div> <div data-bbox="563 481 1230 761">           DOMAINID SIMPAPP            MASTER SITE1            MAXACCESSERS 50            MAXSERVERS 40            MAXSERVICES 40            MODEL MP            LDBAL Y            OPTIONS MIGRATE,LAN            SYSTEM_ACCESS PROTECTED         </div> <div data-bbox="563 772 1230 1030"> <div data-bbox="563 772 1230 840">*MACHINES <span style="float: right;">1005</span></div> <div data-bbox="563 851 1230 1030">           "LCLNX24"            LMD=SITE1            APPDIR="/NFS/LCFILERC/VOL1/APP_1"            TUXCONFIG="/NFS/LCFILERC/VOL1/APP_1/TUXCONFIG"            TUXDIR="/NFS/LCFILERC/VOL1/TUX11GR1"            TYPE="A"         </div> </div> <div data-bbox="563 1041 1230 1164"> <div data-bbox="563 1041 1230 1108">*GROUPS <span style="float: right;">1006</span></div> <div data-bbox="563 1120 1230 1164">           GROUP1            LMID=SITE1GRPNO=1         </div> </div> <div data-bbox="563 1176 1230 1344"> <div data-bbox="563 1176 1230 1243">*SERVERS <span style="float: right;">1007</span></div> <div data-bbox="563 1254 1230 1344">           APP1/OEL55_64_EXAX22X8664/SERVERS/SIMPSERV1            SRVGRP=GROUP1 SRVID=1            CLOPT="-A"         </div> </div> <div data-bbox="563 1355 1230 1433"> <div data-bbox="563 1355 1230 1422">*SERVICES <span style="float: right;">1008</span></div> <div data-bbox="563 1433 1230 1433">TOUPPER1</div> </div> <div data-bbox="563 1444 1230 1601"> <div data-bbox="563 1444 1230 1512">*NETWORK <span style="float: right;">1009</span></div> <div data-bbox="563 1523 1230 1601">           "SITE1"            NADDR="//LCLNX24:32333"            NLSADDR="//LCLNX24:32334"         </div> </div>	<div data-bbox="1238 409 1350 761"></div> <div data-bbox="1238 772 1350 985">LCLNX24</div> <div data-bbox="1238 996 1350 2047">LCLNX16</div>	<div data-bbox="1358 409 1469 761"></div> <div data-bbox="1358 772 1469 985">APP_1</div> <div data-bbox="1358 996 1469 2047">APP_2</div>

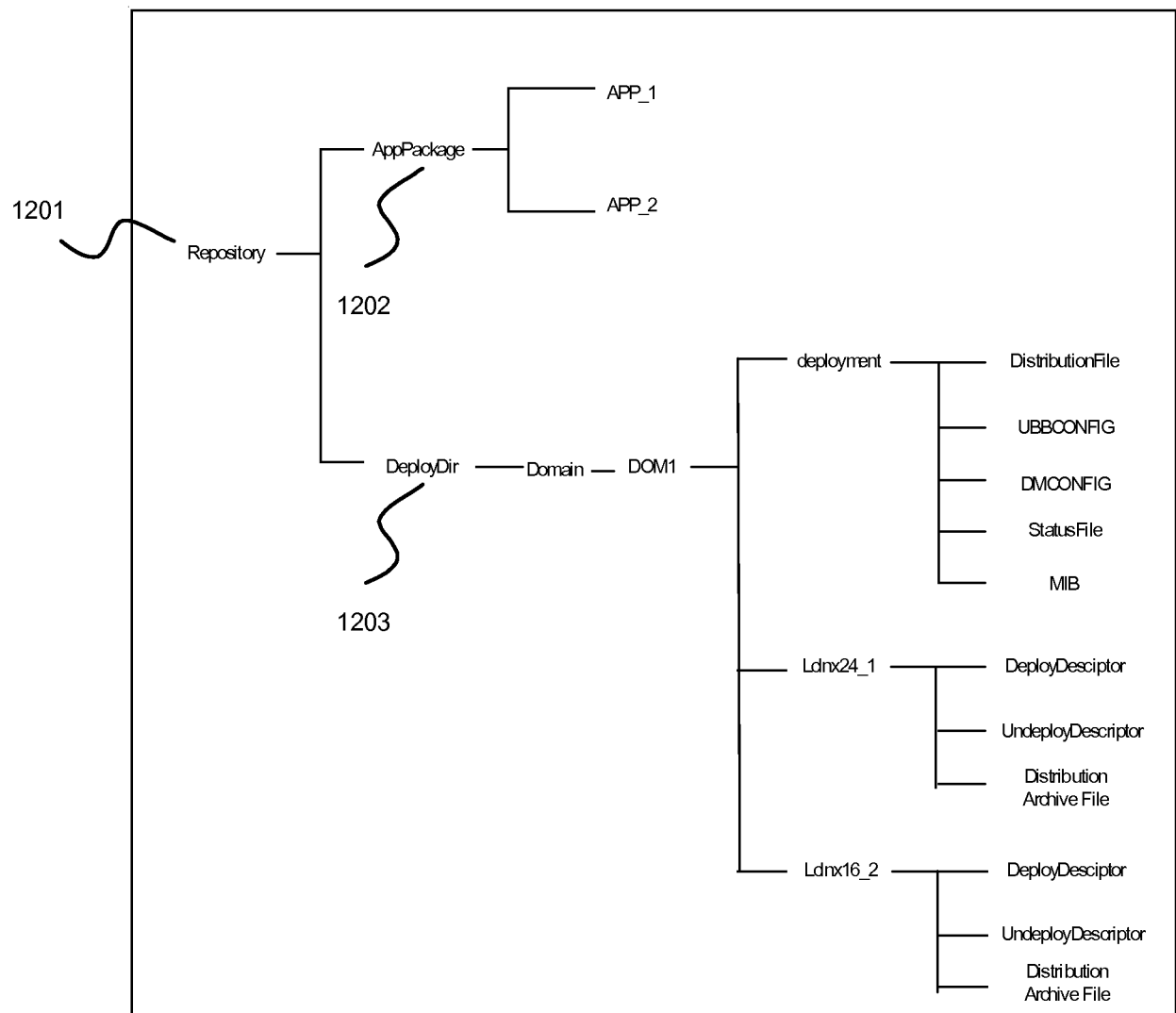
FIGURE 10

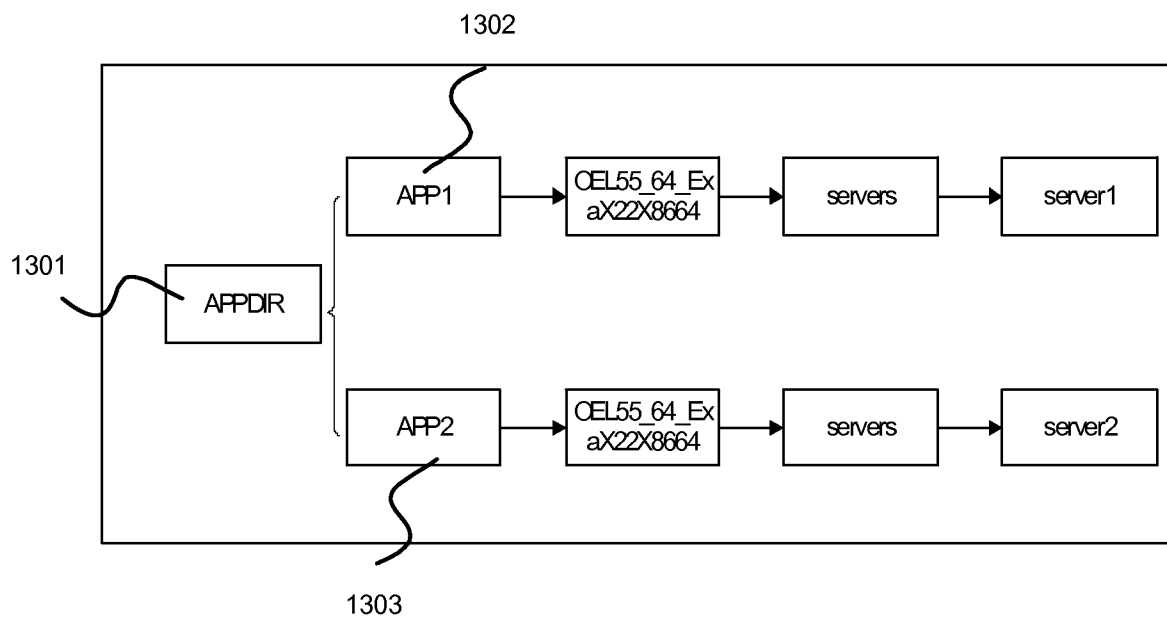
11/14

DOMAIN LIST	*RESOURCES	MACHINE LIST	PACKAGE LIST
DOM1 1101	IPCKEY 33333		
M1 1102	DOMAINID SIMPAPP		
MACHINE	MASTER SITE1		
LCLNX24	MAXACCESSERS 50		
PACKAGE	MAXSERVERS 40		
APP1	MAXSERVICES 40		
	MODEL MP		
	LDBAL Y		
	OPTIONS MIGRATE, LAN		
	SYSTEM_ACCESS PROTECTED		
	*MACHINES		
	"LCLNX24"	LCLNX24	APP1
	LMID=SITE1		
	APPDIR="/NFS/LCFILERC/VOL1/APP_1"		
	TUXCONFIG="/NFS/LCFILERC/VOL1/APP_1/TUXCONFIG"		
	TUXDIR="/NFS/LCFILERC/VOL1/TUX11GR1"		
	TYPE="A"		
	"LCLNX16"	LCLNX16	APP2
	LMID=SITE2		
	APPDIR="/NFS/LCFILERC/VOL2/APP_2"		
	TUXCONFIG="/NFS/LCFILERC/VOL2/APP_2/TUXCONFIG"		
	TUXDIR="/NFS/LCFILERC/VOL2/TUX11GR1"		
	TYPE="B"		
	*GROUPS		
	GROUP1		
	LMID=SITE1 GRPNO=1		
	GROUP2		
	LMID=SITE2 GRPNO=2		
	*SERVERS		
	APP1/OEL55_64_EXAX22X8664/SERVERS/SIMPSEV1		
	SRVGRP=GROUP1 SRVID=1		
	CLOPT="-A"		
	APP2/OEL55_64_EXAX22X8664/SERVERS/SIMPSEV2		
	SRVGRP=GROUP2 SRVID=1		
	CLOPT="-A"		
	*SERVICES		
	TOUPPER1		
	TOUPPER2		
	*NETWORK		
	"SITE1"		
	NADDR="//LCLNX24:32333"		
	NLSADDR="//LCLNX24:32334"		
	"SITE2"		
	NADDR="//LCLNX16:32333"		
	NLSADDR="//LCLNX16:32334"		

FIGURE 11

12/14

**FIGURE 12**

**FIGURE 13**

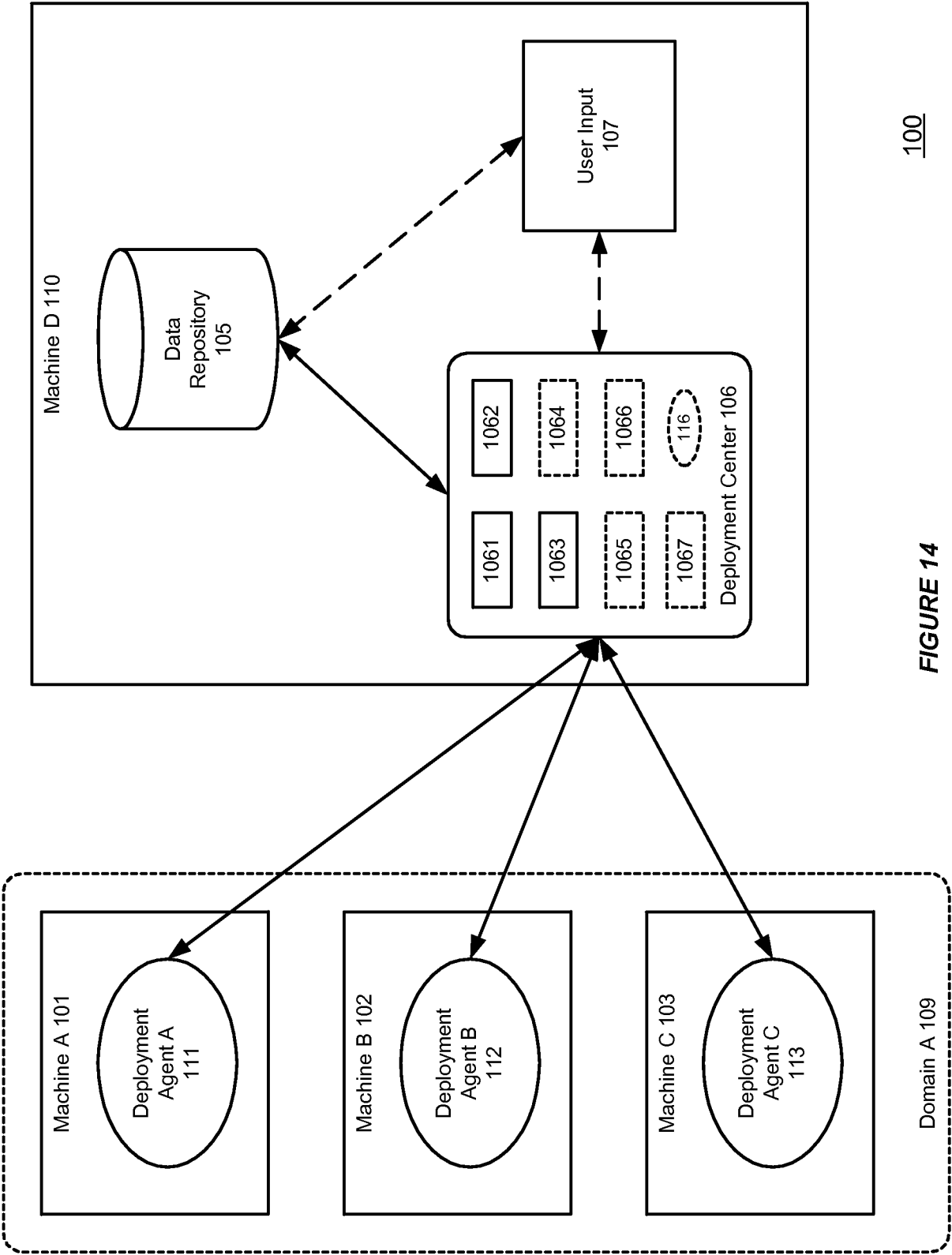


FIGURE 14

100

## INTERNATIONAL SEARCH REPORT

International application No.

PCT/US 12/57136

## A. CLASSIFICATION OF SUBJECT MATTER

IPC(8) - G06F 9/44 (2012.01)

USPC - 717/172

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)

USPC: 717/172

IPC(8): G06F 9/44 (2012.01)

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

USPC: 717/168; 717/172; 717/177; 709/201(Keyword limited; terms below)

IPC(8): G06F 9/44 (2012.01) (Keyword limited; terms below)

Electronic data base consulted during the international search (name of data base and, where practicable, search terms used)

PubWEST (PGPB, USPT, EPAB, JPAB); Google (Scholar, Patents, Web); PatBase (All)

Terms used: transactional middleware application package binary file configuration information data server relationship parameter generate distribution package deploy target platform domain machine create directory automatic metadata repository compressed file tier

## C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X	US 2003/0172135 A1 (Bobick et al.), 11 September 2003 (11.09.2003), entire document, especially Abstract, Claim 18, para [0056], [0068]-[0069], [0075], [0077], [0125], [0156], [0184]-[0185], [0199], [0394], [0694], [0829], [0894], [0952]	1-6, 8-9, 11-16, 18-19
Y		7, 10, 17, 20-22
Y	US 6,640,238 B1 (Bowman-Amuah), 28 October 2003 (28.10.2003), entire document, especially Abstract; col 37, ln 62 to col 38, ln 2	7, 17, 21-22
Y	US 2008/0256531 A1 (Gao et al.), 16 October 2008 (16.10.2008), entire document, especially Abstract, para [0049]	10, 20-22

☐ Further documents are listed in the continuation of Box C.


\* Special categories of cited documents:

"A" document defining the general state of the art which is not considered to be of particular relevance

"E" earlier application or patent but published on or after the international filing date

"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)

"O" document referring to an oral disclosure, use, exhibition or other means

"P" document published prior to the international filing date but later than the priority date claimed

"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

"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

"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

"&amp;" document member of the same patent family

Date of the actual completion of the international search

15 November 2012 (15.11.2012)

Date of mailing of the international search report

29 NOV 2012

Name and mailing address of the ISA/US

Mail Stop PCT, Attn: ISA/US, Commissioner for Patents  
P.O. Box 1450, Alexandria, Virginia 22313-1450

Facsimile No. 571-273-3201

Authorized officer:

Lee W. Young

PCT Helpdesk: 571-272-4300  
PCT OSP: 571-272-7774