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

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
1 December 2005 (01.12.2005)

PCT

(10) International Publication Number
WO 2005/112599 A2

(51) International Patent Classification: Not classified

(21) International Application Number:
PCT/US2005/017860

(22) International Filing Date: 20 May 2005 (20.05.2005)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
60/572,883 20 May 2004 (20.05.2004) US
11/132,883 19 May 2005 (19.05.2005) US

(71) Applicant *(for all designated States except US)*: **BEA SYSTEMS, INC.** [US/US]; 2315 North First Street, San Jose, CA 95131 (US).

(72) Inventor; and

(75) Inventor/Applicant *(for US only)*: **SRINIVASAN, Anantha, Bala** [IN/US]; 1610 Sanchez Street, San Francisco, CA 94131 (US).

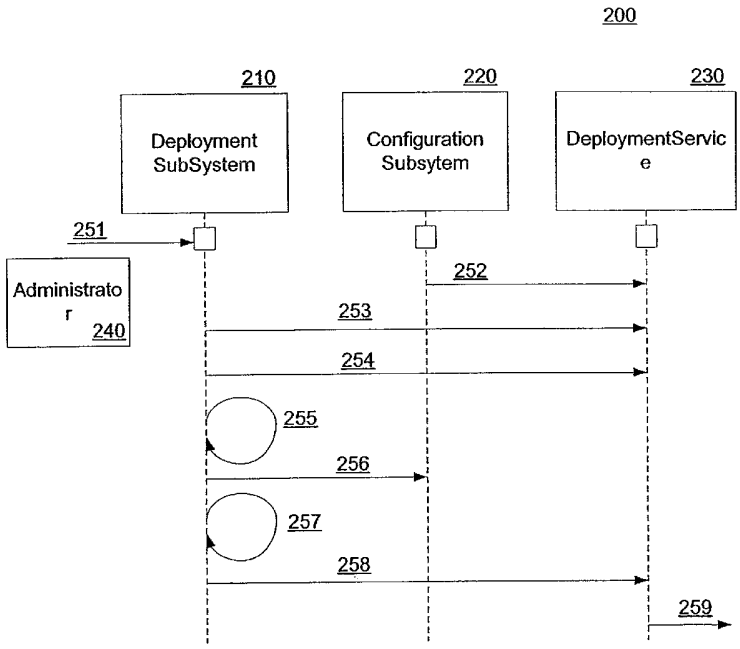
(74) Agents: **MEYER, Sheldon, R.** et al.; Fliesler Meyer LLP, Four Embarcadero Center, Fourth Floor, San Francisco, CA 94111-4156 (US).

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

(84) Designated States *(unless otherwise indicated, for every kind of regional protection available)*: ARIPO (BW, GH, GM, KE, LS, MW, MZ, NA, SD, SL, SZ, TZ, UG, ZM, ZW), Eurasian (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European (AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, HU, IE, IS, IT, LT, LU, MC, NL, PL, PT, RO, SE, SI, SK, TR), OAPI (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, ML, MR, NE, SN, TD, TG).

[Continued on next page]

(54) Title: SYSTEM AND METHOD FOR APPLICATION DEPLOYMENT SERVICE



(57) Abstract: A transactional distribution infrastructure enables multiple participants to work together to deploy changes to an executing context. Embodiments can prepare new changes to executing applications based on a copy of the running context or domain rather than the actual running domain. Performing the changes on a copy of the running domain rather than on the actual running domain allows the changes to be examined, therefore providing an opportunity to detect errors and prevent the changes from causing failures on the domain.

WO 2005/112599 A2

Published:

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

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

SYSTEM AND METHOD FOR APPLICATION DEPLOYMENT SERVICE

COPYRIGHT NOTICE

CLAIM OF PRIORITY

This application claims the benefit of:

U.S. Provisional Patent Application No. 60/572,883, entitled SYSTEM AND METHOD FOR APPLICATION DEPLOYMENT SERVICE by Ananthan Bala Srinivasan, filed on May 20, 2004 (Attorney Docket No. BEAS-01577US0); and

U. S. Patent Application No. XX/XXX,XXX entitled SYSTEM AND METHOD FOR APPLICATION DEPLOYMENT SERVICE, by Ananthan Bala Srinivasan, filed on May 19, 2005 (Attorney Docket No. BEAS-01577US1).

FIELD OF THE INVENTION

The current invention relates generally to managing one or more servers, and more specifically to a system and method for providing an application deployment service.

BACKGROUND

In a clustered or distributed application server environment, some form of data replication is useful in order to distribute configuration and deployment information from an administration (Admin) server to one or more managed servers within its domain. The typical requirements for such a data replication process is to be able to distribute data items over point-to-point connections, i.e., using TCP for example, that provides for a measure of flow control. Data replication allows managed servers to persistently cache data on local disks, and speeds server startup by reducing the amount of data to be transferred. Storing data locally on each server also allows independent server startup and /or restart when the Admin server is unavailable, unreachable, or in a failed condition.

However, updates and changes to a domain configuration need to be distributed to the servers to which the changes are applicable in a manner that maintains the consistency of configuration of the domain. Some changes need to be consistent across the entire domain while others need to be consistent within clusters. Consistency is also crucial to the application deployment and redeployment process because services like availability, failover, load-balancing and in-memory replication are scoped to a cluster and not the entire domain.

BRIEF DESCRIPTION OF THE DRAWINGS

FIGURE 1 is an illustration of a domain in an embodiment.

FIGURE 2 is an illustration of an administration server interaction process during deployment in an embodiment.

FIGURE 3 is an illustration of a managed server interaction process during deployment in an embodiment.

FIGURE 4 is an illustration of an administration server interaction process during deployment in an embodiment.

DETAILED DESCRIPTION

In accordance with embodiments, there are provided transactional distribution mechanisms and methods for deploying changes made to a copy of an application context to the actual executing application context. These mechanisms and methods can enable multiple participants that are part of a domain, comprised of an administration server and one or more managed servers, to coordinate and control deployment of changes to an application context. An embodiment prepares new changes based on a copy of the running context or domain rather than the actual running domain. Performing the changes on a copy of the executing context rather than on the actual executing context allows the changes to be examined prior to deploying the changes in the actual executing context, thereby providing an opportunity to detect errors and prevent the changes from causing failures on the domain. Transactional distribution mechanism embodiments can include a multiple-phase commit process, such as described herein with reference to examples illustrated by FIGURES 2 – 4, between the administration server and the at least one target server that may be used to coordinate and control deployment of the at least one change in the domain.

In an embodiment, a method for deploying changes to a domain is provided. The method includes receiving at least one change to a copy of an application context executing on at least one target server managed by an administration server. The at least one change may be performed on a copy of the application context rather than the application context executing in the domain. Performing changes on a copy of the application context can provide the capability to examine the at least one change to detect errors that would cause failures in the domain prior to deploying the at least one change on the application context executing at least one target server. The at least one change is deployed to the application context executing on the at least one target server

in the domain. A multiple-phase commit process implementing a transactional framework between the administration server and the at least one target server may be used to deploy the at least one change in the domain.

As used herein, the term "topology" is intended to describe a distributed architecture. As used herein, the term "artifacts" includes a topology of the domain and applications. The term domain is intended to mean a management unit comprising an administrative server and at least one managed server. A domain may include servers that may be grouped into clusters as illustrated in domain 100 of FIGURE 1. As used herein, the term application is intended to be broadly construed to include any application capable of executing in the domain, and may include without limitation any application, program or process resident on one or more computing devices, network based applications, web based server resident applications, web portals, search engines, photographic, audio or video information storage applications, e-Commerce applications, backup or other storage applications, sales/revenue planning, marketing, forecasting, accounting, inventory management applications and other business applications and other contemplated computer implemented services.

While the present invention is described with reference to an embodiment in which changes made to a copy of an application context are deployed to the executing application context in a managed server domain, the present invention is not limited to changes in application contexts, but may be practiced by embodiments deploying changes to other contexts, i.e., operating systems, system programs and the like without departing from the scope of the embodiments claimed.

FIGURE 1 illustrates a domain in an embodiment. As shown in FIGURE 1, a Domain 100 includes managed or target servers 110, 120, 130, 150, 160 and 170, administration server 140, and clusters 180 and 190. In domain 100, cluster 180 includes servers 110, 120 and 130 and cluster 190 includes 150, 160 and 170. Though illustrated with three managed servers, a

cluster may contain any number of managed servers. The administration server may exchange information with any of the managed servers. The servers of domain 100 comprise a distributed system architecture with the administration server ensuring the managed servers have the same information regarding domain configuration and applications.

In one embodiment, the domain 100 includes a distributed configuration subsystem and a deployment subsystem. The configuration subsystem manages systems, adds clusters, and adds global resources (such as a database). The deployment subsystem handles applications and associated tasks. Both subsystems collaborate and are consumers of the underlying deployment service. The deployment service distributes configuration artifacts and files from the administration server to managed server.

In one embodiment, the system topology includes a deployment subsystem, configuration subsystem and deployment service distributed over the domain. On an administration server, when changes are made to the domain by means of a deployment request, the deployment request can be received via the deployment subsystem or the configuration subsystem. The appropriate subsystem creates an artifact and will contact other subsystem(s) to determine if anything needs to be added to the artifact. This is done because when either a server, application or some other entity is being deployed into a domain, it requires changes to the configuration and an addition to the domain. Thus, the two subsystems need to work together to provide a system snapshot of the overall domain configuration. Once completed, the administration server transmits a deployment package to the appropriate managed server(s). Once the package is received by the managed server(s), the managed server processes the package. In one embodiment, the managed server process the package in the same order as the administration server did. Thus, the appropriate subsystem receives a signal indicating a package has been received and that the subsystem

should perform any necessary processing to validate this package. The same signal is sent to the other subsystem.

FIGURES 2 - 4 illustrate control sequences that a deployment request may undergo as the deployment request is processed in an embodiment. The process illustrated shows interactions with the configuration subsystem and the deployment subsystem of the administration server and managed servers. The deployment subsystem of FIGURES 2 - 4 is based on a new application deployment request. Thus, the configuration subsystem is contacted first and initiates work at certain points in the deployment process. The scope of the present invention also includes server deployment requests, at which the process begins with the configuration subsystem. Thus, if the deployment involves adding a new server, the deployment request is received through the configuration subsystem. If the deployment involves adding an application, the request is received through the deployment subsystem.

FIGURE 2 illustrates a process of an administration server at the start of a deployment request involving deployment of an application through deployment APIs. As shown in FIGURE 2, a Process 200 involves actions by deployment subsystem 210, configuration subsystem 220, deployment service 230 and administrator 240, each of which may be embodied within administration server 140 of FIGURE 1. In one embodiment, the deployment service 230 is configured to distribute configuration artifacts from the administration server to the managed servers.

In one embodiment, the deployment subsystem 210 performs work related to deployment of new applications and implements JSR-88 deployment APIs, BaseDeploymentProvider and DeploymentServiceCallbackHandler. The configuration subsystem 220 performs work relating to global changes, servers, and management control and implements ConfigurationDeploymentProvider and DeploymentServiceCallbackHandler. DeploymentService 230 comprises the infrastructure to perform transactional processing and implements

DeploymentRequestFactory, DataTransferHandlerManager, DeploymentServiceOperations and ChangeDescriptorFactory.

In an embodiment, administrator 240 first initiates a deployment of an application. The admin server 140 receives a deployment request at the deployment subsystem 210 via the deployment API in step (251). In one embodiment, the deployment request may take the form of deploy(app2,targets), wherein app2 represents the application to be deployed and targets are the managed servers at which the application is to be deployed. The deployment subsystem 210 informs the configuration subsystem 220 of the deployment request. Then, both the configuration subsystem 220 and deployment subsystem 210 register deployment service callback handlers with deployment service 230 in step (252) and step (253), respectively. In step (252) the configuration subsystem 220 sends a register command with parameters of "config" and configCallbackHandler. In step (253) the deployment subsystem 210 sends a register command with parameters of "deployment" and deploymentCallbackHandler. Registering their respective DeploymentServiceCallbackHandlers with the deployment service 230 allows deployment subsystem 210 and configuration subsystem 220 to receive any callbacks sent up by the deployment service 230.

After registering, the deployment subsystem 210 requests the creation of a deployment request from the deployment service in step (254). In one embodiment, whichever subsystem receives the deploy request in step (252) will initiate the request to create a deployment request. In one embodiment, the request is a container into which each of subsystems will add work that they need done to complete the deployment. In one embodiment, the deployment service 230 generates the deployment request using via the DeploymentRequestFactory. The deployment subsystem 210 then creates a base deployment job entry in step (255). Once the job entry is generated, the deployment subsystem 210 signals the configuration subsystem 220 to add to

the generated job entry in step (256). If the configuration subsystem 220 has anything to add to the job entry, the deployment subsystem 210 makes the additions in step (257). The deployment subsystem 210 then sends the job entry to the deployment service 230 in step (258). In response, the deployment service 230 encodes the job entry into a package ("the deployment request package") and sends the deployment request package to appropriate target servers in step (259). For example, if the deployment request is to be sent to a cluster of three servers, than three instances of the deployment request package will be generated and distributed.

FIGURE 3 illustrates a processing of a target server that is the subject of a deployment in an embodiment. As shown in FIGURE 3, a Process 300 involves actions by deployment service 310, configuration subsystem 320 and deployment subsystem 330, all located on the target (managed) server. In one embodiment, the deployment subsystem 330 performs work related to deployment of new applications and implements DeploymentReceiver. The configuration subsystem 320 performs work relating to global changes, servers, and management control and implements Deployment Receiver. The DeploymentService 310 contains the infrastructure to do transactional work and implements DeploymentReceiver Coordinator.

In operation, the target server receives the deployment request package from the administration server in step (341). In one embodiment, the deployment request is received from the administration server through a DeploymentReceiversCoordinator. Once the deployment request is received, the deployment service 310 of the target server generates a context ("the deployment context") from the parameters of the request package in step (342). In one embodiment, the deployment context is a view of the environment in the managed server from the point of view of the configuration. The context is a copy of the running system domain, upon which the changes are to be made. The context is not yet active. This allows for the system to determine whether

the deployment will negatively affect the running system by first applying the changes to the copy, or context version. Thus, the context running system is examined, or validated, before it is actually deployed. Validation of the changed domain may involve, after looking at the configuration, determining whether the deployment can be performed successfully, are the required external resources available, have all required services been set-up, and whether the prepare phase can be executed successfully. In one embodiment, the validation includes each managed server retrieving information and content from the deployment package and providing the information and content to the managed server modules. The modules then perform the validation of the package contents. The modules may include messaging subsystems, web server subsystems, and other subsystems that determine how functionality is delivered to the application.

Once the context is prepared, the deployment service 310 instructs the configuration subsystem 320 to prepare for the deployment in step (343). Upon receipt of the prepare instruction, the configuration subsystem 320 performs prepare related operations and updates to the context as necessary in step (344). The configuration subsystem 320 then responds to the target server deployment service 310 to indicate that the prepare request is acknowledged in step (345). Upon receiving the acknowledgement signal from the configuration subsystem 320, the deployment service 310 repeats this process with the deployment subsystem 330. Thus, the deployment service 310 instructs the deployment subsystem 330 to prepare for the deployment in step (346). Upon receipt of the prepare instruction, the deployment subsystem 330 performs prepare related operations and updates the context as necessary in step (347). The deployment subsystem 330 then responds to the target server deployment service 310 to indicate that the prepare request is acknowledged in step (348). After the systems of the target server have acknowledged the "prepare for deployment" signals, the target server replies to the administration server in step (349). The

target server then waits to receive a reply from the administration server before continuing. Operation of the administration server between the time of step (349) and step (350) of process 300 is illustrated and discussed with reference to FIGURE 4. The target server eventually receives a reply containing a commit signal from the administration server in step (350). Once the commit signal is received, the deployment service 310 of the target server sends commit commands to the configuration subsystem 320 in step (351) and deployment subsystem 330 in step (352). In one embodiment, if in step (345) or step (348) one or more of the managed server subsystems return a failed acknowledge signal, the managed server sends a fail signal back to the administration server in step (349). The deployment process for the particular deployment request then completes.

FIGURE 4 illustrates processing of an administration server from the point that prepare acknowledge signals are received from target servers in an embodiment. As shown in FIGURE 4, a Process 400 involves actions by deployment subsystem 430, configuration subsystem 420, deployment service 430, all within an administration server. In one embodiment, the deployment subsystem 430, configuration subsystem 420 and deployment service 410 are similar to those illustrated and described in reference to FIGURE 2.

In operation, the administration server receives prepare acknowledgments from the target servers in step (441). Once all target servers have responded, the administration server's deployment service 410 acts to save a record of the commit in step (442). In one embodiment, if the administration server should fail, the saved record of the commit will allow the deployment to proceed from this point rather than starting the entire process over from step (251) in FIGURE 2. Once the record is saved, the administration server distributes commit signals to all the involved target servers in step (443). In another embodiment, if the administration server receives one or more prepare

failure signals in step (441), it will mark the operation as failed and inform the target servers of the failed operation in step (443).

After distributing the commit signals, the administration server sends a deployment successful signal to the configuration subsystem 420 in step (444). In response, the configuration subsystem 420 saves the new configuration file config.xml in step (445). The configuration subsystem 420 also saves the old configuration file at some well known location. Next, the administration server sends a deployment successful signal to the deployment subsystem 430 in step (446). Upon receiving the deployment successful signal, the deployment subsystem 430 then saves any state managed by the deployment subsystem 430 for the particular deployment in step (447). The deployment subsystem 430 then returns a signal indicating the deployment was successful to the administrator 240.

In one embodiment, the present invention includes a transactional distribution infrastructure that enables multiple participants that are part of a deployment change to work together to create a deployment context. The infrastructure prepares new changes based on a copy of the running context or domain rather than the actual running domain by employing a two phase commit process implemented using a transactional framework between components of an administration server and one or more managed servers in a domain. Performing the changes on a copy of the running domain rather than on the actual running domain allows the changes to be examined, therefore providing an opportunity to detect errors and prevent the changes from causing failures on the domain.

Other features, aspects and objects of the invention can be obtained from a review of the figures and the claims. It is to be understood that other embodiments of the invention can be developed and fall within the spirit and scope of the invention and claims.

The foregoing description of preferred embodiments 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. Obviously, 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.

In addition to an embodiment consisting of specifically designed integrated circuits or other electronics, the present invention may be conveniently implemented using a conventional general purpose or a specialized digital computer or microprocessor programmed according to the teachings of the present disclosure, as will be apparent to those skilled in the computer art.

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. The invention may also be implemented by the preparation of application specific integrated circuits or by interconnecting an appropriate network of conventional component circuits, as will be readily apparent to those skilled in the art.

The present invention includes a computer program product which is a storage 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.

Stored on any one of the computer readable medium (media), the present invention includes software for controlling both the hardware of the general purpose/specialized computer or microprocessor, and for enabling the computer or microprocessor to interact with a human user or other mechanism utilizing the results of the present invention. Such software may include, but is not limited to, device drivers, operating systems, and user applications.

Included in the programming (software) of the general/specialized computer or microprocessor are software modules for implementing the teachings of the present invention, including, but not limited to, implementing a deployment service for a domain.

CLAIMS

IN THE CLAIMS:

1. A method for deploying changes to a domain, the method comprising:

receiving at least one change to a copy of an application context executing on at least one target server managed by an administration server; and

deploying the at least one change to the application context executing on the at least one target server in the domain, thereby enabling the at least one change to be examined to detect errors that would cause failures in the domain prior to deploying the at least one change on the application context executing at least one target server.

2. The method of claim 1, wherein receiving at least one change to a copy of an application context executing on at least one target server managed by an administration server further comprises:

performing the at least one change on a copy of the application context rather than the application context executing in the domain.

3. The method of claim 1, wherein deploying the at least one change to the application context executing on the at least one target server in the domain further comprises:

using a multiple-phase commit process implementing a transactional framework between the administration server and the at least one target server.

4. The method of claim 3, wherein using a multiple-phase commit process implementing a transactional framework between the administration server and the at least one target server further comprises:

receiving an acknowledgement from each component of the at least one target server indicating that the component is ready to accept the change, prior to sending a command to commit the at least one change to each component of the at least one target server.

5. The method of claim 4, further comprising:

discontinuing deployment of the at least one change if a failed acknowledgement is received from any component of the at least one target server.

6. The method of claim 1, further comprising:

testing the at least one change prior to deploying the at least one change on the application context executing at least one target server.

7. A machine-readable storage medium carrying one or more sequences of instructions for deploying changes to a domain, which instructions, when executed by one or more processors, cause the one or more processors to carry out the steps of:

receiving at least one change to a copy of an application context executing on at least one target server managed by an administration server; and

deploying the at least one change to the application context executing on the at least one target server in the domain, thereby enabling the at least one change to be examined to detect errors that would cause failures in the domain prior to deploying the at least one

change on the application context executing at least one target server.

8. The machine-readable medium of claim 7, wherein the instructions for receiving at least one change to a copy of an application context executing on at least one target server managed by an administration server include instructions for carrying out the step of:

performing the at least one change on a copy of the application context rather than the application context executing in the domain.

9. The machine-readable medium of claim 7, wherein the instructions for deploying the at least one change to the application context executing on the at least one target server in the domain include instructions for carrying out the step of:

using a multiple-phase commit process implementing a transactional framework between the administration server and the at least one target server.

10. The machine-readable medium of claim 9, wherein the instructions for using a multiple-phase commit process implementing a transactional framework between the administration server and the at least one target server include instructions for carrying out the step of:

receiving an acknowledgement from each component of the at least one target server indicating that the component is ready to accept the change, prior to sending a command to commit the at least one change to each component of the at least one target server.

11. The machine-readable medium of claim 10, further comprising instructions for carrying out the step of:

discontinuing deployment of the at least one change if a failed acknowledgement is received from any component of the at least one target server.

12. The machine-readable medium of claim 7, further comprising instructions for carrying out the step of:

testing the at least one change prior to deploying the at least one change on the application context executing at least one target server.

13. An administration server for deploying changes to a domain, the administration server comprising:

a processor; and

one or more stored sequences of instructions which, when executed by the processor, cause the processor to carry out the steps of:

receiving at least one change to a copy of an application context executing on at least one target server; and

deploying the at least one change to the application context executing on the at least one target server in the domain, thereby enabling the at least one change to be examined to detect errors that would cause failures in the domain prior to deploying the at least one change on the application context executing at least one target server.

14. A method for receiving changes to be deployed in a domain, the method comprising:

receiving at least one change to an application context being executed;

determining whether deployment of the at least one change to the application context being executed would produce any negative affects;

signaling an administration server whether the deployment of the at least one change to the application context being executed would produce any negative affects; and

committing the at least one change to the application context being executed, if a commit signal is received from the administration server responsive to the signaling.

15. The method of claim 14, further comprising:

discontinuing deployment of the at least one change to the application context if a fail signal is received from the administration server.

16. A machine-readable medium carrying one or more sequences of instructions for receiving changes to be deployed in a domain, which instructions, when executed by one or more processors, cause the one or more processors to carry out the steps of:

receiving at least one change to an application context being executed;

determining whether deployment of the at least one change to the application context being executed would produce any negative affects;

signaling an administration server whether the deployment of the at least one change to the application context being executed would produce any negative affects; and

committing the at least one change to the application context being executed, if a commit signal is received from the administration server responsive to the signaling.

17. The machine-readable medium of claim 16, further comprising instructions for carrying out the step of:

discontinuing deployment of the at least one change to the application context if a fail signal is received from the administration server.

18. A target server for receiving changes to be deployed in a domain, the target server comprising:

a processor; and

one or more stored sequences of instructions which, when executed by the processor, cause the processor to carry out the steps of:

receiving at least one change to an application context being executed;

determining whether deployment of the at least one change to the application context being executed would produce any negative affects;

signaling an administration server whether the deployment of the at least one change to the application context being executed would produce any negative affects; and

committing the at least one change to the application context being executed, if a commit signal is received from the administration server responsive to the signaling.

19. A method for performing a deployment change, comprising:

receiving a deployment request;

registering a first deployment subsystem and a first configuration subsystem with a first deployment module, the a first deployment subsystem, first configuration subsystem and first deployment module located at an administration server;

generating a deployment job entry at an administration server;

forwarding the deployment job entry to a managed server;

acknowledging the deployment request by a second deployment subsystem and a second configuration subsystem, the second deployment subsystem and the second configuration subsystem located at the managed server; and

performing the deployment request at the managed server.

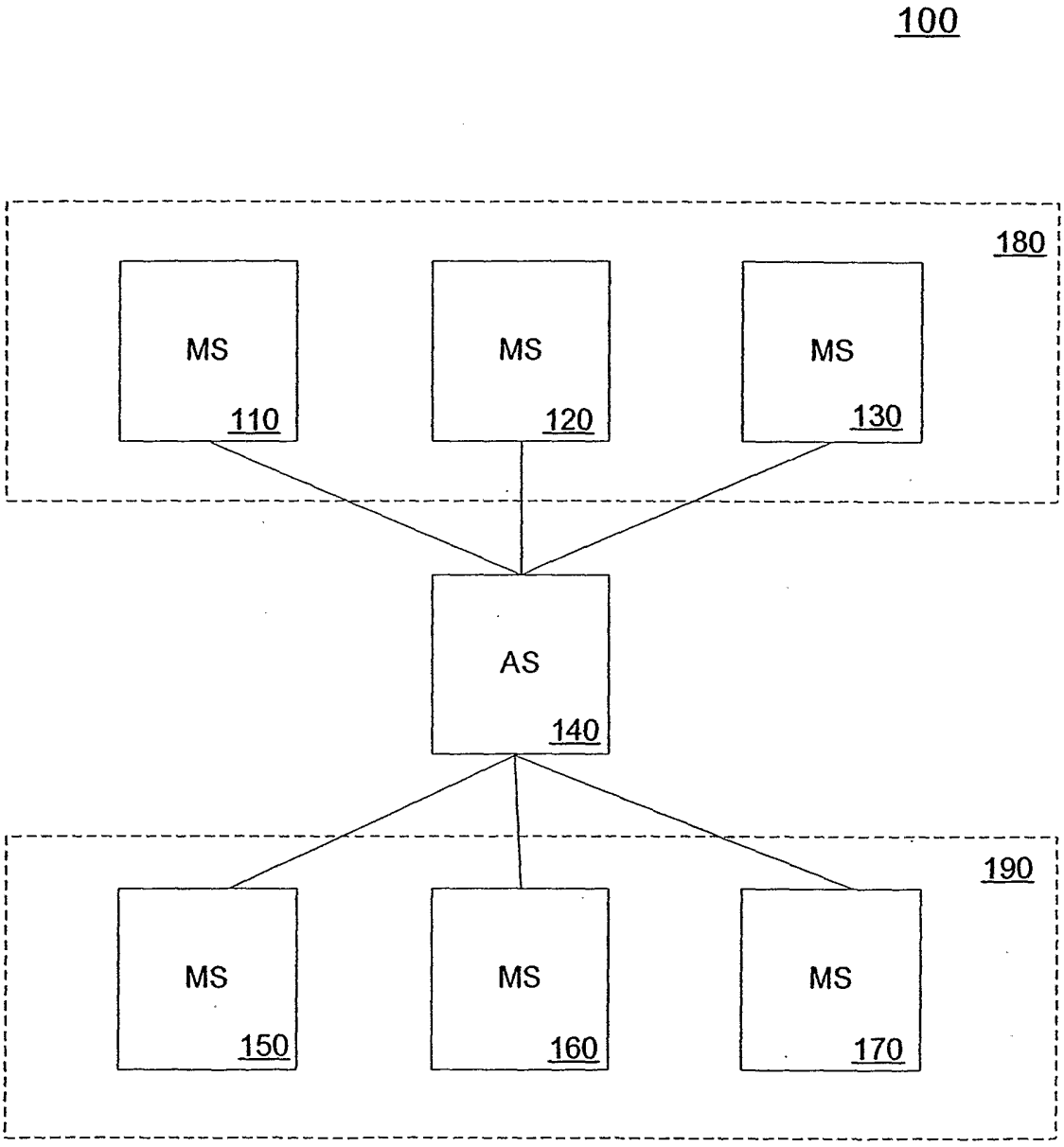


Figure 1

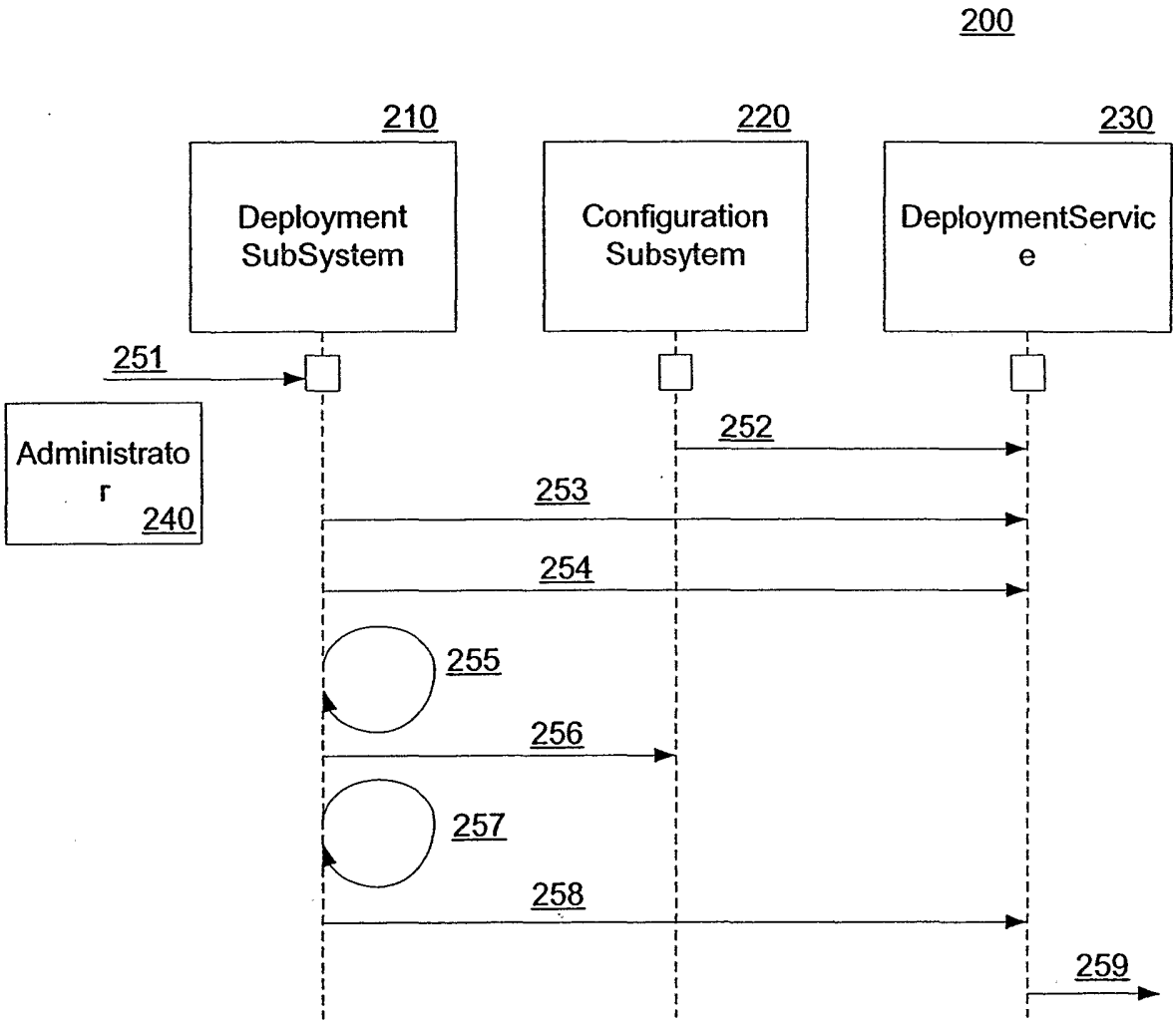

200
210
220
230
Deployment SubSystem
Configuration Subsytem
DeploymentServic e
251
Administrato r
240
252
253
254
255
256
257
258
259

Figure 2

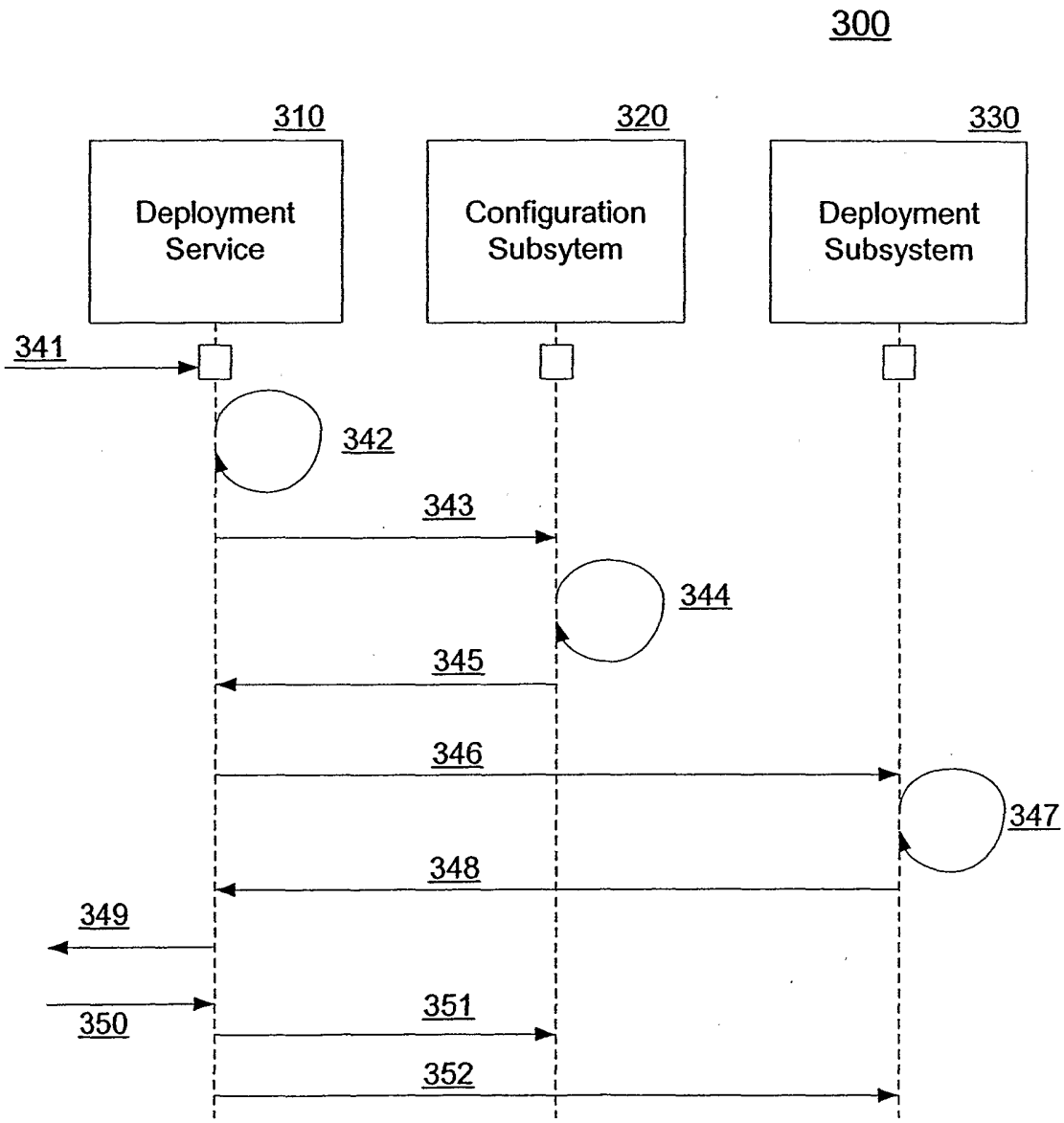

300
310
320
330
Deployment Service
Configuration Subsytem
Deployment Subsystem
341
342
343
344
345
346
347
348
349
350
351
352

Figure 3

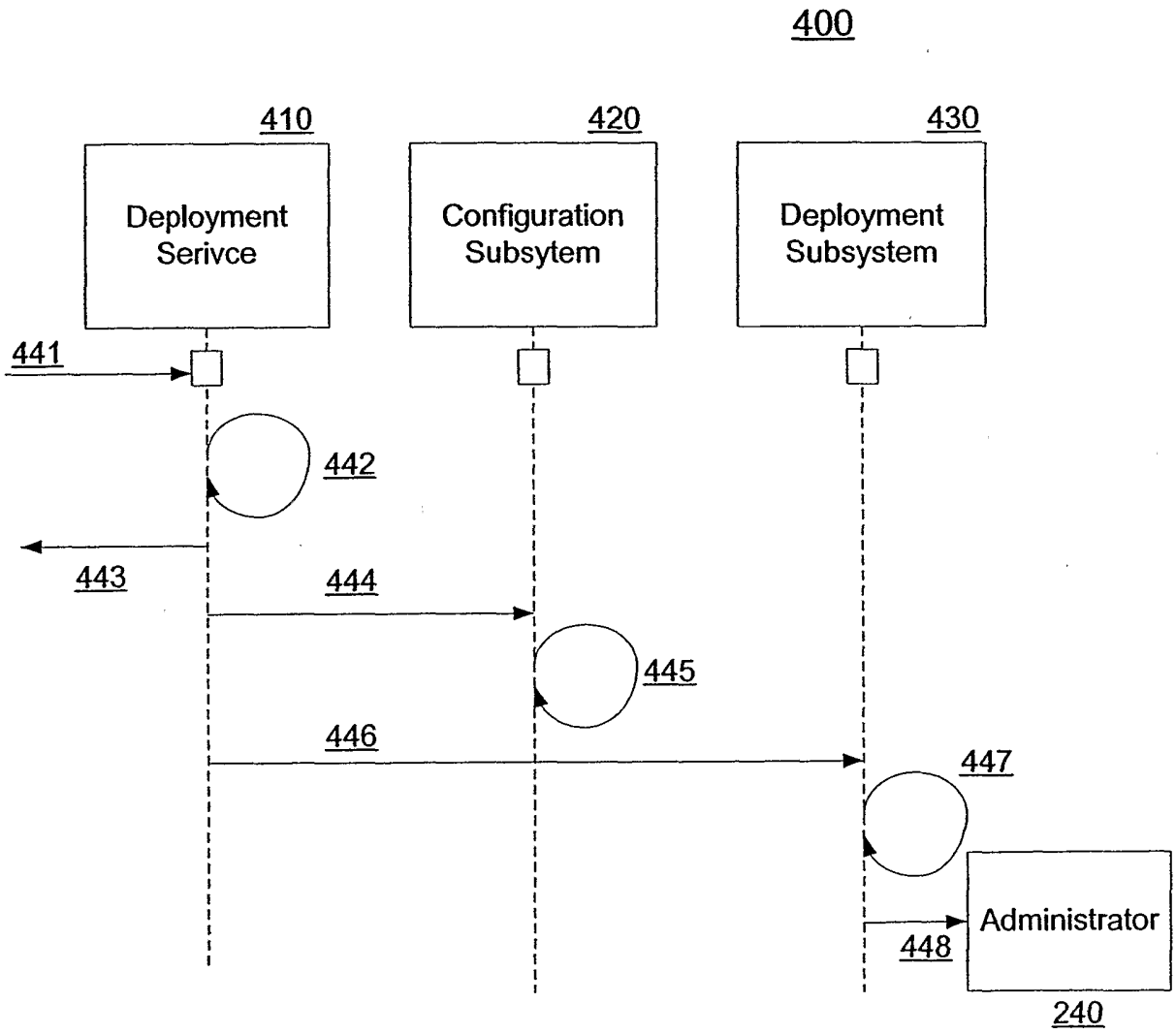

400
410
420
430
Deployment Serivce
Configuration Subsytem
Deployment Subsystem
441
442
443
444
445
446
447
448
Administrator
240

Figure 4